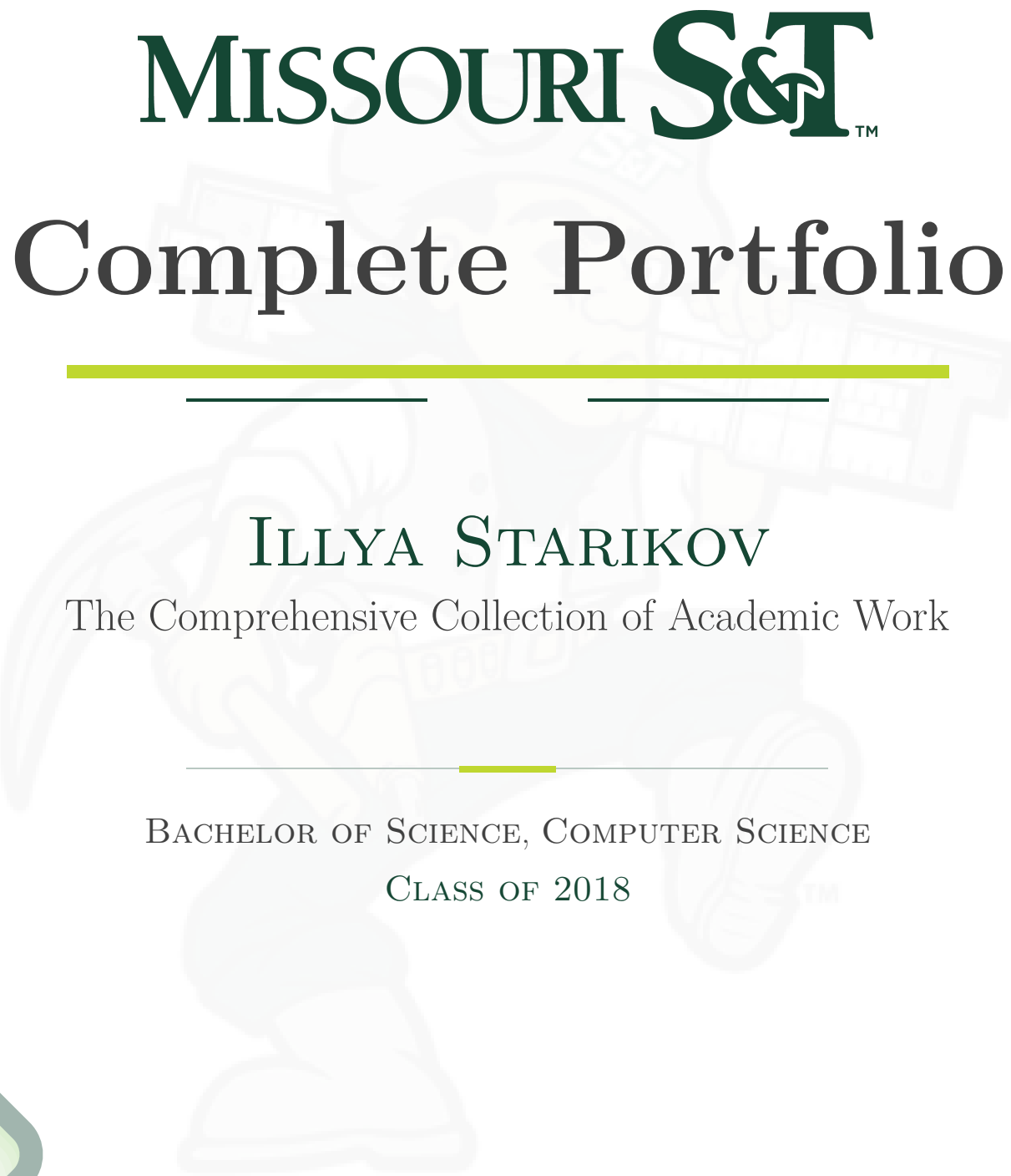
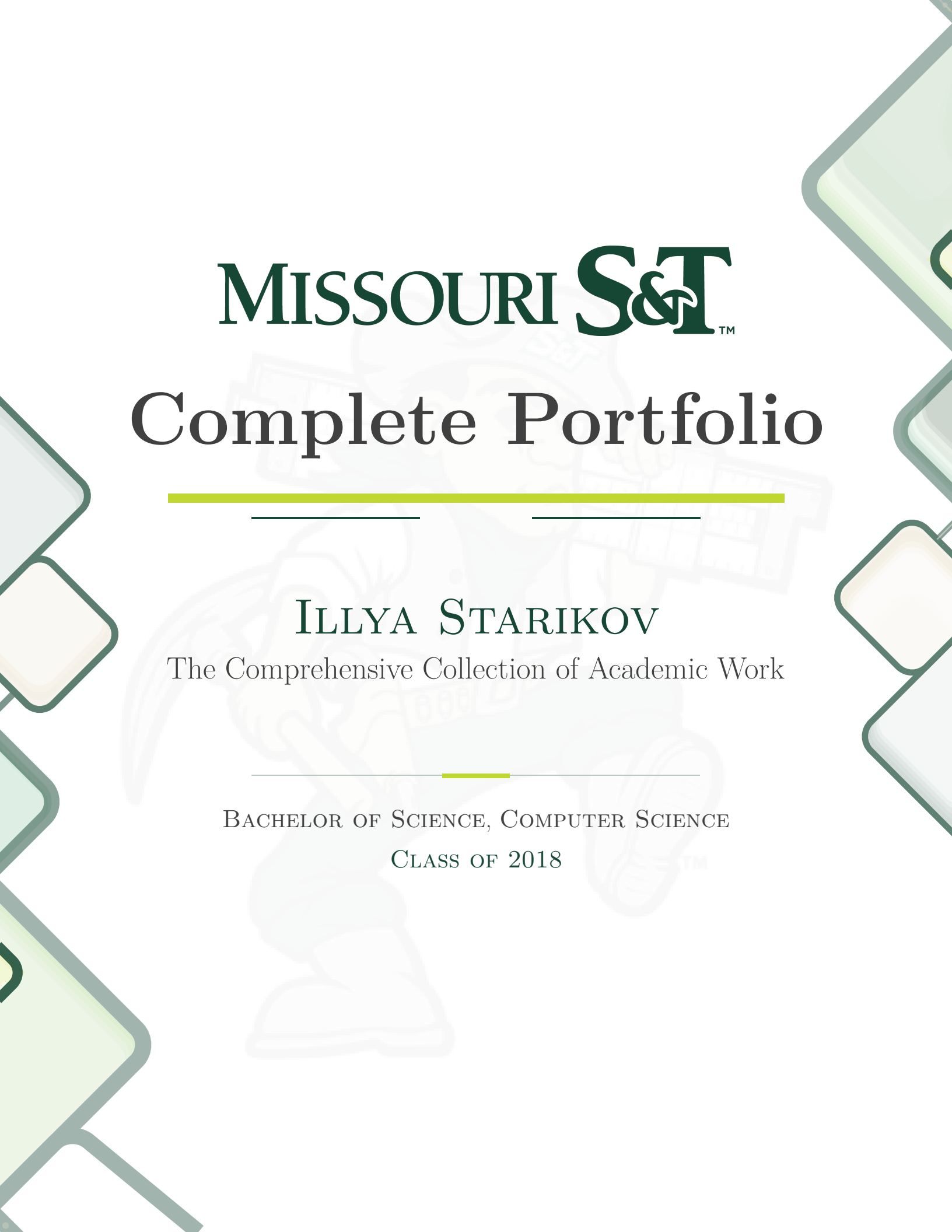




MISSOURI S&T™

Complete Portfolio

ILLYA STARIKOV

The Comprehensive Collection of Academic Work

BACHELOR OF SCIENCE, COMPUTER SCIENCE

CLASS OF 2018

Preface

This comprehensive portfolio represents the entirety of my academic journey at Missouri University of Science and Technology, documenting four years of rigorous coursework across computer science, mathematics, and engineering disciplines. Each page within this collection reflects the depth and breadth of a technical education that challenged me to think critically, solve complex problems, and develop expertise across multiple domains of computing.

From foundational courses in data structures and discrete mathematics to advanced topics in artificial intelligence and numerical methods, this portfolio captures not only the knowledge acquired but also the evolution of my analytical and technical capabilities. The materials presented here—including lecture notes, homework assignments, laboratory reports, and projects—demonstrate the comprehensive nature of the Missouri S&T computer science curriculum and my dedication to academic excellence throughout my undergraduate studies.

Illya Starikov

Bachelor of Science, Computer Science
Missouri University of Science and Technology

Academic Advisors: Dr. Jennifer Leopold, Dr. A. Ricardo Morales,
Dr. Simone Silvestri, Professor Clayton Price

Class of 2018
starikov@mst.edu

“Miners Dig Deeper”



Table of Contents

Contents

Table of Contents	1
[CPE2210] Introduction To Computer Engineering	7
Exam I	8
Exam II	9
Lab 1	10
Lab 2	12
[CPE3150] Micro Embedded Design	14
Lecture 3	15
Lecture 4	18
Lecture 6	19
Lecture 7	22
Lecture 9	23
Project 1 Report	24
Project 3 Report	31
[CS1001] Data Structures Lab	40
Lab 1	41
Lab 5	43
Lab 8	44
Lab 9	45
Lab 10	46
Lab 14 Assignment	47

Final Project	49
[CS1200] Discrete Math	53
Chapter 5.1	54
Chapter 5.1 Continued	55
Chapter 6.1	56
Chapter 7.1-7.2	58
Chapter 8.1-8.2	59
Chapter 8.3	60
Chapter 8.5 Partial Order Relations	63
Chapter 9.1-9.2	65
Chapter 9.3-9.4	66
Chapter 9.3-9.6 Continued	68
Chapter 9.5-9.6	70
Chapter 10.1-10.2 Part I	72
Chapter 10.1-10.2 Part II	74
Chapter 10.2	77
Chapter 10.3-10.5	79
Test II Study Guide	81
Test III Review	91
Homework 4	103
Homework 5	104
[CS1510] Data Structures	105
Graph	106
Heap	107
Recursion	109
Tree	114
[CS2300] Databases	118
Quiz I	119
Homework 1	120
Homework 2	124
Homework 4	126
Group Meeting I	128
Conceptual Database Design	130
ERR Diagram	133
Logical and Physical Database Design	134
Wireframe	142
Final Report	146
End of Semester	171
[CS2500] Algorithms	173
1. Algorithms Introduction	174
2. Asymptotic Analysis	175
3. Divide And Conquer	177
4. Recurrence	178
5. Correctness	179

6. Quick Sort	180
7. Heapsort	181
8. Counting Sort	182
9. Example	184
10. Dynamic Programming	186
11. LCS	187
Exam I Study Guide	188
Exam II Study Guide	193
Exam III Study Guide	194
Homework 1	197
Homework 2	201
Homework 3	206
Homework 4	208
Project Report I	213
Project Report II	220
Project Graphs	226
[CS3001] Skills Development	228
Assignment Grade	229
[CS3100] Software Engineering I	230
Requirements	231
Protocol Description	245
Progress Presentation	252
Final Presentation	275
Final Summary	281
[CS3200] Numerical Methods	284
Lecture 1	285
Lecture 2	286
Lecture 4	287
Cribsheet	288
Numerical Methods Equations	289
Homework 9	291
Homework 10	297
[CS3800] Operating Systems	298
Lecture 1	299
Lecture 2	300
Exam 1 Cribsheet	301
Exam 2 Cribsheet	302
Homework 2 Report	303
[CS4096] Software Systems Development	305
Progress Presentation	306
Communication Protocol	318
Feature List	320
Ethics	326

Reflection Essay	329
[CS5200] Analysis Of Algorithms	331
Homework 1	332
Homework 2	337
Homework 3	348
Homework 4	354
Homework 5	359
Homework 6	362
Homework 7	369
Homework 8	393
Final Exam	413
[CS5400] Artificial Intelligence	429
Exercise 1	430
Exercise 2	433
Exercise 3	439
Exercise 4	441
Exercise 5	442
Exercise 6	444
Exercise 7	446
Exercise 8	448
Bonus 1	449
[CS5402] Data Mining	450
Test 1	451
Test 2	457
Homework 1	463
Homework 2	468
Homework 3	471
Homework 5	474
Homework 6	480
[MATH1214] Calculus I	486
Complete Notes	487
[MATH1215] Calculus II	500
Chapter 6	501
Chapter 7	504
Chapter 8	515
Chapter 9	525
Chapter 10	538
Chapter 11	545
11.2 Polar Coordinates	556
[MATH2100] Foundations Of Mathematics	564
Course Notes	565
[MATH2222] Calculus III	566

Chapter 12	567
Chapter 13	584
Chapter 14	599
Chapter 15	607
[MATH3304] Differential Equations	622
Sample Output	623
[PHIL3225] Business Ethics	625
Portfolio	626
[MATH1214] Calculus I	631
Complete Notes	632
[MATH1215] Calculus II	645
Complete Notes	646
[MATH2222] Calculus III	695
Complete Notes	696
[STAT3117] Statistics	749
Exam III Cribsheet	750
Final Project Report	752
[PHYS1135] Physics I	758
First Law of Thermodynamics	759
[PHYS2135] Physics II	761
Lecture 1	762
Lecture 2	763
Lecture 3	764
Lecture 4	766
Lecture 5	768
Lecture 6	770
Lecture 7	771
Lecture 9	773
Lecture 10	774
Lecture 11	775
Lecture 12	776
Lecture 13	777
Exam 1	778
Exam 2	779
[PHYS2305] Modern Physics	783
Review	784
[PSYC1101] Psychology 101	790
A Beautiful Mind	791
Famous People	792

[TEACH] Teaching Materials	793
CS1570 Lecture: Modern C++	794
CS1570 Lab 8	801
CS1570 Lab 15	805
Grading Rubric	813
iOS Development With Swift 4	814

CPE2210

Introduction To Computer Engineering



1 Test I Review

1.1 Redundancy Law

$$A + AB = A \quad (1)$$

$$A(A + B) = A \quad (2)$$

1 Digital Design

- nFET = Not Negated. Turns on by 1 or High.
- pFET = Negated. Turns on by 0 or Low.
- Xor = $\bar{A}B + A\bar{B}$
- Fan-out refers to the number of additional gates that can be physically connected to the output of a gate.
 - I_{ILMax} = The max current draw of a gate input for a low input value
 - I_{OLMax} = The max current output of a gate for a low output
 - I_{IHMax} = The max current draw of a gate input for a high input value
 - I_{OHMax} = The max current output of a gate for a high output
 - Max Fanout = $Min(\frac{I_{OLMax}}{I_{ILMax}}, \frac{I_{OHMax}}{I_{IHMax}})$
- We don't do POS for Variable Entered K-Maps. Turn into SOP.
- Inside Each Comparator
 - $out\_gt = in\_gt + (in\_eq * a * \bar{b})$
 - $out\_lt = in\_lt + (in\_eq * \bar{a} * b)$
 - $out\_eq = in\_eq * (a \oplus b)$

Lab #1: Laboratory Equipment

Illya Starikov

January 25, 2016

1 Objective

To familiarize self with lab equipment, namely the DC Power Supply, Function Generator, Frequency Counter/Timer, Digital Multimeter, and Mixed Signal Oscilloscope.

2 Equipment Used

- DC Power Supply
- Function Generator
- Frequency Counter/Timer
- Digital Multimeter
- Mixed Signal Oscilloscope

3 Procedures

1. Generate a square wave with an amplitude of 3 V and frequency of 900 Hz.
 - Offset DC by 2 V.
2. Generate 3 signals:
 - 1 kHz
 - 10 kHz
 - 50 kHz

3. Generate 7 V DC, measuring with the Digital Multimeter.
4. Follow the steps of category 5.

4 Conclusion and Results

4.1 Function Generator

4.2 Frequency Counter/Timer

Function Generator	Function Counter
1.017 kHz	1.1 kHz
10.02 kHz	10.1 kHz
49.99 kHz	50.1 kHz

4.3 Digital Multimeter (DMM)

Multimeter:	7.015 V
Power Supply:	7.1 V

4.4 Mixed Signal Oscilloscope

At approximately 2.430 MHz the square wave starts becoming indistinguishable, taking the shape of a sinusoidal wave.

5 Notes And Comments

- $v_{rms} = \frac{VP}{\sqrt{2}}$
- $p - p$ means peak to peak.

Lab # 2

Illya Starikov

February 01, 2016

1 Objective

To familiarize self with Quartus and ModelSim software.

2 Equipment Used

- Computer
 - Quartus Software
 - ModelSim Software

3 Procedures

1. Launch Quartus and go through initial project startup
2. Create a new Block Diagram / Schematic File
3. Add logic gates (as specified)
4. Compile the logic gates.
 - If any compiler errors, fix.
 - Rinse and repeat.
5. Create VHDL from schematic.
6. Launch ModelSim, go through initial setup.
7. Add VHDL to the project.
8. Simulate the schematic built and print the results.

4 Conclusion and Results

ModelSim and Quartus is quite useful software that makes the simulation of circuits a trivial task.

5 Notes And Comments

- Every project must have a file named the same as the project.
- Do **not** save into the C directory.

CPE3150

Micro Embedded Design



3 Control Transfer Instructions and Time Delay Generation

- There are 5 types of instructions
 - One of which is MOV, MOVC, MOVX
- No one uses JMP.
 - Because it's a bit difficult to work with — as apposed to a long jump.
- NOP is literally do nothing
 - It's to kill time.
- Let's multiply 5×5

JZ Multiply By 5

```
ORG 0

MOV R0, #5
MOV R1, #0

MOV A, #0

LOOP:  MOV A, R1
        ADD A, #5

        MOV R1, A
        MOV A, R0

        DEC A
        JNZ A, LOOP

END
```

DJNZ Multiply By 5

```
ORG 0

MOV R0, #5
MOV A, #0

LOOP:  ADD A, #5
       DJNZ R0, Loop

END
```

Nested Loop

Load A with 5, add 10 for 300 times.

```
ORG 0

MOV A, #5

MOV R0, #3

LOOP2: MOV R1, #100    ; Runs 3 times
LOOP1: ADD A, #10      ; Runs 100 times
                     ; 3x100 = 300 times

       DJNZ R1, LOOP1
       DJNS R0, LOOP2
```

- $1 \text{ MC} = 1.0852\mu s$ — **KNOW THIS!**
- For slides, the values (in order) of the delay is

Machine Cycle	Times
1	1
1	∞
2	∞
1	∞
2	∞
1	∞
2	$200 \times \infty$
2	$1 \times \infty$

Delay Example

DELAY: MOV R0, #251

DL1: MOV R1, #182

DL2: DJNZ R1, DL2

 DJNZ R0, DL1

 RET

4 I/O Port Programming

- Port 0 is weird, no pull up transistor
- Port 0 also has bit addressability
- Only when ALE is set will the data be treated as an address P0
- Port 1 general input/output port. Boring port.
- Port 2 is date for input/output and addressing. Meaning it does not need pull up resistors.
- Port 3 has special functions.
- For duty cycle.

$$f = 5\text{kHz} \implies T = 200\mu s \quad (1)$$

$$\frac{200}{1.085} = 184\text{MC} \implies 92 \text{ iterations} \quad (2)$$

$$\frac{92}{5} = 18.4 \quad (3)$$

- I/O Example — 2
 - We'll use CJNE A, # or CJNE A, dir
 - Also, SUBB

6 Arithmetic and Logic Instructions

- Unpacked = 4 0s out front
- Packed = no 0s out front, more efficient.
 - However, there are reasons for having 0000s out front
 - Could process nibble, instead of byte by byte
 - Makes processing easier
- BCD addition **does not** work.. kind of.
- DA takes into account the AC as well.
 - If AC = 1 or > 9, add 06H
 - If CY = 1 or > 9, add 60H.
- The extra B in SUBB mean subtracted with borrow.
 - Make sure to set clear CY before using SUBB
- One special case, divide by zero: 0V = 1, values remain the same.
 - Division example: A = 9, B = 5
- XRL only works for 8 bits.
 - Same addressing modes as for ANL
- Complement works for A, C or *anything* that is bit addressable.
- CJNE changes the CY flag
- Serial Communication example (we use RLC because we want to use the Carry flag for transmitting data)

ORG 0

MOV A, #35H

MOV P2, #0

MOV R0, #8

```
SETB P2.1
SETB P2.1
```

```
TX: RLC A
    MOV P2.1, C
    DJNZ R0, TX
```

```
SETB P2.1
SETB P2.1
```

```
END
```

- Same example, backwards

```
ORG 0
```

```
MOV R0, #8
```

```
RX: MOV C, P2.5
    RRC A
    DJNZ R0, RX
```

```
MOV R2, A
```

```
END
```

Number of '1's example

```
ORG 0
MOV R0, #0 ; Counter for 1s
MOV R1, #8 ; Counter for loop
```

```
MOV A, P2
```

```
LP: RRC A
```

```
DJNE
```

$$n\text{bit} \cdot n\text{bit} = 2n\text{bit} \quad (1)$$

```

ORG 0
MOV R0, #30H
MOV @R0, #0

XCHD A, @R0      ; M[30] = 07, A = 30H
SWAP A           ; A = 30H
ORL A, #30H

```

9 Timers and the 8051

- The time delay is $14 * 1.085 = 15.19 \mu\text{s}$
- $65536 - \frac{7500}{1.085} = 58624_{10} = E500_{16}$
- $200 \cdot (65536 - 264) \cdot 1.085 = 14.16 \mu\text{s}$

9 Timers

- TF is overflow flag, tells the timer is expired
 - TR starts/stops timer
- Math for programming timer in Mode 2: $\frac{1}{(241 \cdot 1.085 \cdot 10^{-6} \cdot 2)}$ slide 27

Project #2

Michael Schoen, Abdirahman Osman, Illya Starikov

Due Date: November 8th, 2016

For our project, we have decided to implement a memory matching game. The game will start off by enabling a single light. If the user correctly presses said light, the user will move onto the next level and enable a light sequence. For every successive level, an additional light have to be kept in mind (i.e. for the n^{th} level, there will be n lights that will have to be pressed).

This will continue for 15 games, where upon a successful finish, a special light sequence and a song will asynchronously play. From here, the user can choose to play another game if they so choose.

1 Explanation

Below we will take a more in-depth look into the implementation of our project.

1.1 Random Number Generation

Because we want our game to be different every round, we have to have some form of random number generation. Initially, we had tried to implement a *linear congruential generator*; unfortunately, we were unsuccessful. We instead went with a different implementation.

Both implementations require a seed value (and to have different results, we need a different seed value) — we “cheat” to get this. During our input, we not only check to see if the user initiated a game, but we also increment the register the seed is stored in. This way, there is only a $1/255$ chance the user will get the same game¹.

¹Although this is a significant value (0.39%), it serves well for demo purposes.

Failed Attempt

The basic summary is as follows: beginning with a seed value (named X_n), a new linear, psuedo-random number can be calculated via:

$$X_{n+1} = (aX_n + c) \mod m \quad (1)$$

where the follow conditions hold

1. $a, X_n, c, m \in \mathbb{Z}^+$
2. $0 \leq X_n, c < m$
3. $0 < a < m$
4. $m \neq 0$

The issue we face is it is *highly* recommended that m be quite large (most popular implementations range from 2^{31} to 2^{48}); unfortunately, we only have 2^8 available to us. Because of this, we either get a cycle roughly every 10 iterations or the same number produced. As one might imagine, this is a game-breaking bug; we decided to go with something different.

Our failed attempt for a linear congruential generator is as follows.

```
; Check writeup for how this works
; For now, the formula is

; X_n+1 = aX_n + c mod m

; For our purposes, a = 7, c = incrementor, m = 72
; for our purposes, incrementor can't be a multiple of 7
; X_n is stored in R7, incrementor in R6

; Result will be stored in A
RNG:
    MOV A, R7
    MOV B, #7D
    DIV AB
    MOV A, B
```

```

                                JNZ SKIP1
                                INC R6
SKIP1:                          MOV A,  #7D
                                MOV B,  R7
                                MUL AB                                ; A = aX_n

                                ADD A,  R6
                                MOV B,  #72D
                                DIV AB

                                MOV R7,  B

                                MOV A,  B
                                MOV B,  #10D
                                DIV AB

                                MOV A,  B
                                RET

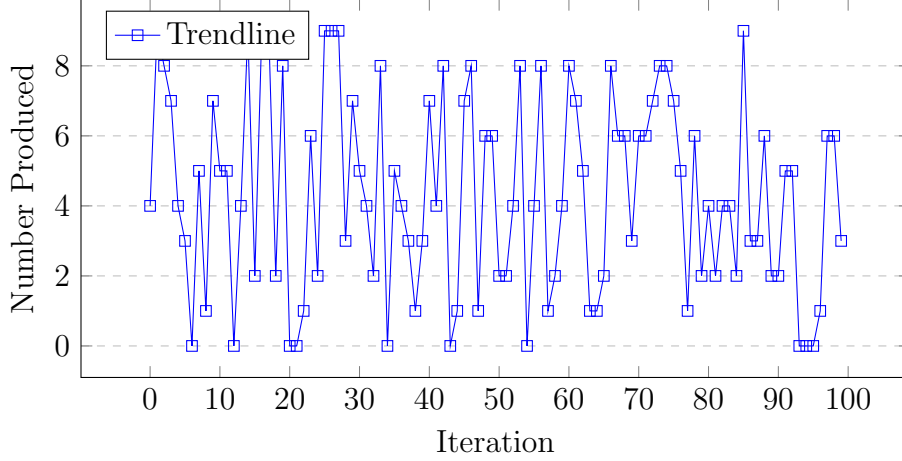
```

Successful Attempt

For our successful attempt, we ported a random number generating from an open source code base². It similar to the linear congruential generator in the sense that it linearly produces a psuedo-random number; however, it is a different formula (one that can't be eloquently described in an equation). Below is a graph that shows the distribution of numbers up to 100 iterations.

²<https://www.pjrc.com/tech/8051/>

Random Numbers Generated (Seed: 7)



1.2 Light Show

When programming the lights, we notice there to be 9 lights; which just so happens to be the exact size of register *A* and the carry bit *C* in the `sfr`. So we decided to map every bit in *A* and *C* to a particular light. For simplicity, we used a 1_2 to represent an ‘on’ state and a 0_2 for the ‘off’ state.

Our mapping is as follows: we start the MSB of *A*, which represents the top-left light. The proceeding two bits make up the top-most row of the lighting board. The fourth bit represents the middle row, *first* light. We continue with this pattern until we come to the last light. The last light is represented by the carry bit *C*. Fig. 1 summarizes this lighting schematic.

As an example, `111000111 1` would enable only the top and bottom lights, leaving the middle row blank. `10110110 1` enables the left and right columns. `10101010 1` makes an X.

As mentioned, to represent an ‘on’ state, we used a 1_2 ; however, the lights are active low. So, the first thing we do is `CPL A` and `CPL C` to ensure we don’t have opposite affect. From there, we rotate through *C* to enable individual light. Finally, we `CPL A` and *C* again to bring the data back to its original state.

The individual lighting effects were done through trial and error of what looked aesthetically pleasing. There are three light patterns, one for losing, advancing to the next level, and the winning light sequence. The winning light sequence has five parts, eloquently named:

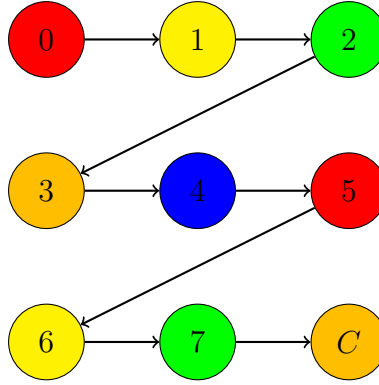


Figure 1 – The lighting schematic for “lights” subroutine.

1. Light Spiral Part I
2. Light Spiral Part II
3. Full Light Spiral
4. Os
5. Bring the Thundaaa

1.3 Music

We know the frequency of the Philips P89LPC932A1 to be 7.373 MHz, with 2 cycles per machine cycle. Therefore,

$$\frac{2 \text{ cycles}}{\text{machine cycle}} \cdot \frac{1 \text{ Period}}{7.373 \text{ MHz}} = 0.271 \text{ } 26 \text{ } \mu\text{s}/_{\text{mc}} \quad (2)$$

We use this calculation as the base of our music.

E5	$f = 659.255 \text{ Hz} \implies T = 1516 \text{ }\mu\text{s}$ $1516 \text{ }\mu\text{s} \div 0.27126 \text{ }\mu\text{s}/\text{mc} = 5589 \text{ mc}$ $5589 \text{ mc} \div 4 = 1398 \text{ mc}$ $1398 \text{ mc} \implies 699 \text{ iterations (with DJNZ)}$
F5	$f = 698.456 \text{ Hz} \implies T = 1431 \text{ }\mu\text{s}$ $1431 \text{ }\mu\text{s} \div 0.27126 \text{ }\mu\text{s}/\text{mc} = 5275 \text{ mc}$ $5589 \text{ mc} \div 4 = 1318 \text{ mc}$ $1318 \text{ mc} \implies 569 \text{ iterations (with DJNZ)}$
G5	$f = 783.991 \text{ Hz} \implies T = 1275.5 \text{ }\mu\text{s}$ $1275.5 \text{ }\mu\text{s} \div 0.27126 \text{ }\mu\text{s}/\text{mc} = 4702 \text{ mc}$ $4702 \text{ mc} \div 4 = 1176 \text{ mc}$ $1176 \text{ mc} \implies 588 \text{ iterations (with DJNZ)}$
D5	$f = 587.330 \text{ Hz} \implies T = 1702.6 \text{ }\mu\text{s}$ $1702.6 \text{ }\mu\text{s} \div 0.27126 \text{ }\mu\text{s}/\text{mc} = 6277 \text{ mc}$ $7045 \text{ mc} \div 4 = 1570 \text{ mc}$ $1570 \text{ mc} \implies 785 \text{ iterations (with DJNZ)}$
C5	$f = 523.251 \text{ Hz} \implies T = 1911.1 \text{ }\mu\text{s}$ $1911.1 \text{ }\mu\text{s} \div 0.27126 \text{ }\mu\text{s}/\text{mc} = 7045 \text{ mc}$ $7045 \text{ mc} \div 4 = 1760 \text{ mc}$ $1760 \text{ mc} \implies 880 \text{ iterations (with DJNZ)}$
Flat D5	$f = 554.365 \text{ Hz} \implies T = 1803.8 \text{ }\mu\text{s}$ $1803.8 \text{ }\mu\text{s} \div 0.27126 \text{ }\mu\text{s}/\text{mc} = 6650 \text{ mc}$ $6650 \text{ mc} \div 4 = 1662 \text{ mc}$ $1662 \text{ mc} \implies 831 \text{ iterations (with DJNZ)}$

2 Future Work

3 Work Effort

- Michael Schoen
 - Programmed binary counter.

- Programmed game logic.
- Osman Abdirahman
 - Programmed initial beep.
 - Programmed song implementation.
- Illya Starikov
 - Programmed random number renerator.
 - Programmed light sequence.

Project #3

Michael Schoen, Abdirahman Osman, Illya Starikov

Due Date: Tuesday, December 6th, 2016

For the second project, we were delighted to be able to use a higher level programming language; we¹ decided to apply this new-found excitement to implement a retro game from the 70s: Space Invaders.

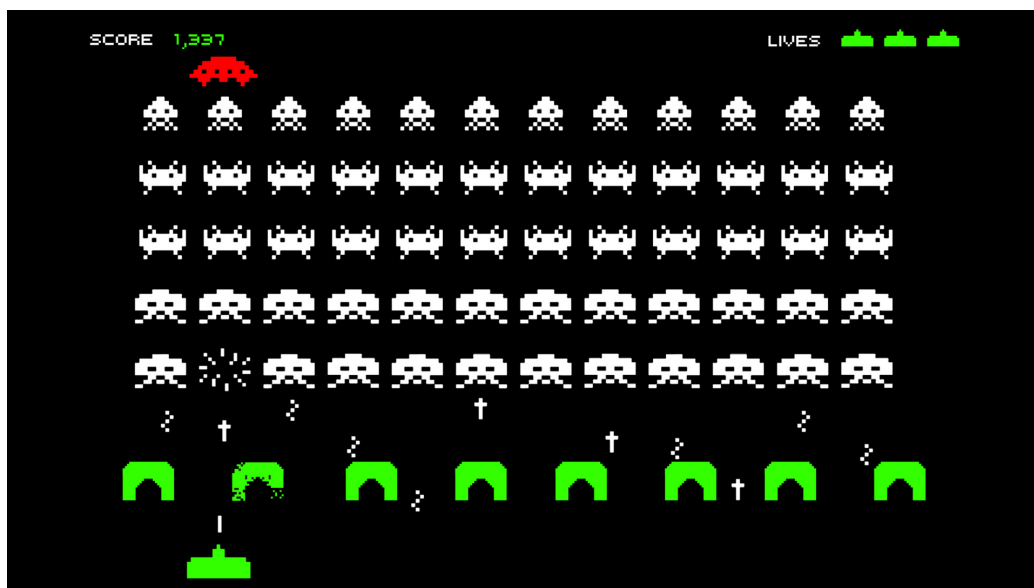


Figure 1 – The original space invaders.

When this idea came crashing and burning down to the ground, we decided to combat this with a additional music, an elegant user interface, and a keyboard (with some other miscellanies).

¹Illya.

1 Project Description

Below we will go into more detail about each individual parts of our project.

1.1 The Game

The game we initially decided to go with was space invaders. We had intention of doing the game logic on the microcontroller through serial communication; however, we learned early that this was likely not be possible². We describe in *Problems Encountered* how we absolved this. From here, we decided our game would exclusively be in the terminal.

Our game essentially creates a two dimensional array (in three segments — the header to show score and level, the aliens, and the shooter). Then we loop through depending on the input:

- Move the shooter left.
- Move the shooter left.
- Quits the game
- Shoots with the gunner.
- No input meant refresh game.

We achieved the drawing through `curses`³. Ultimately, we did not get our game fully done, but a good majority of it. See Figure. 2 for a look at the UI of the game.

1.2 Music

The formula for each note is

$$\frac{1/4 \text{ Oscillator Frequency}}{\text{Note Frequency}}$$

So for supposing we want to produce the note C4,

²Our hex file with very basic functionality was 25 kB.

³Can be read about here [https://en.wikipedia.org/wiki/Curses_\(programming_library\)](https://en.wikipedia.org/wiki/Curses_(programming_library))

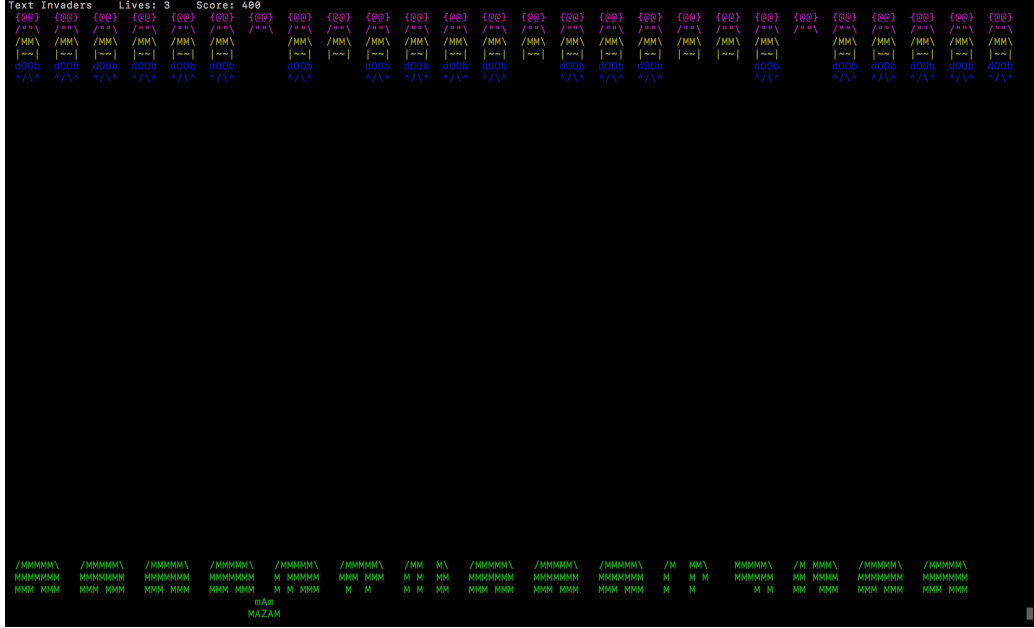


Figure 2 – Text Invaders, our version of space invaders..

$$\frac{1/4 \text{ Oscillator Frequency}}{\text{Note Frequency}} = \frac{1/4 \cdot 7.373 \text{ Hz}}{261.63 \text{ Hz}} = 7.045 \text{ ms} \quad (1)$$

To produce the music, we had a function that took in which note and the type of note (16th, 8th, 4th, half or whole note). The function multiplied the base length of a note by a constant depending on the type of note. A 16th note would be multiplied by 1, an 8th by 2, a quarter note by 4, a half-note by 8, and a whole note by 16. We used timers and interrupts for the period of the note and the length of the note.

We also used sheet music to get the notes. For speeding and slowing down the music, the type of note was just multiplied by 2 or divided by 2 respectively.

1.3 Interactive User Interface

The menu goes through three stages; printing the pen-rose triangle background, printing options, then allowing the selection of items. The following are printed

1. Counter

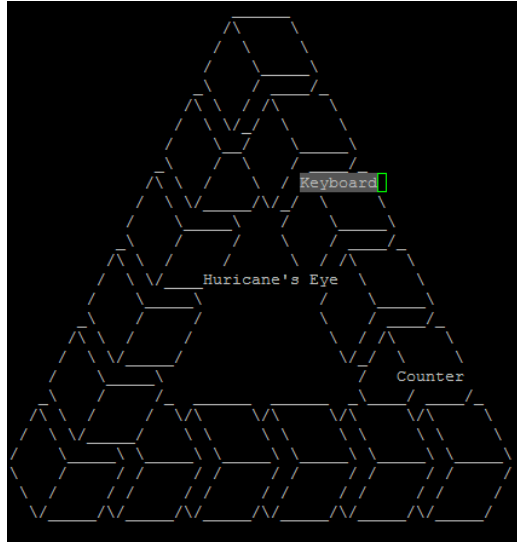


Figure 3 – The pen-rose style menu.

2. Hurricane's Eye
3. Keyboard

respectively. From there the user has the option to cycle upwards using P_2 (see Figure. 4), cycle down using P_8 , or select using P_5 . The cycle starts with 1, highlighting Counter if P_2 is pressed, then Hurricane's eye will be highlighted, and if printed once again Counter will be selected. If the menu user presses P_1 once at the 3rd option (Keyboard) then the menu will cycle back to option 1 and vice versa if P_8 were clicked at the 1st option. Selecting an menu option with P_5 while highlighted will start its respective function. Also each menu option has a corresponding number displayed on a seven segment display.

1.4 Seven Segment Display

For the seven segment display, we had to set P_2 to be push/pull so that the display would receive enough power. We then looked up the correct values to output to the pins to make each number show up on the display and used those values in the increment/decremented function as well as the simple function that lets you display any number. We used the data sheet for the

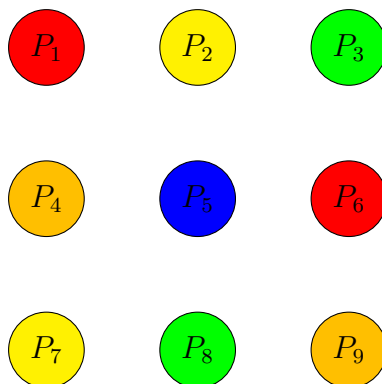


Figure 4 – The Main Menu.

seven segment display to see how to connect it on the breadboard. See Figure. 5 for additional information.

1.5 Keyboard

Our keyboard has a C scale set to switches $P_1 \dots P_8$ with P_9 escaping to the menu. Each switch uses the corresponding light associated with its position save P_9 which is the escape button.

1.6 Curses

In order to print to any part of the screen we needed to implement a structure to send ANSI control sequences and characters to the terminal using `uart_get()` as a base. To do this we implemented an 8051 port of curses that takes in (y, x) coordinates as arguments for printing `chars` and `strings` at any defined (x, y) location. This allows for the implementation of the menu since it requires printing of options in sometimes nonlinear fashion.

We found a whole host of others issues associated with printing characters to a terminal, printing to an arbitrary location, control sequences, fluctuation in crystal timings, and requesting information from a terminal. We accounted for these individually.

PIN NO.			
1	Cathode E	6	Cathode B
2	Cathode D	7	Cathode A
3	Common Anode DIG. 1	8	Common Anode DIG. 2
4	Cathode C	9	Cathode F
5	Cathode DP	10	Cathode G

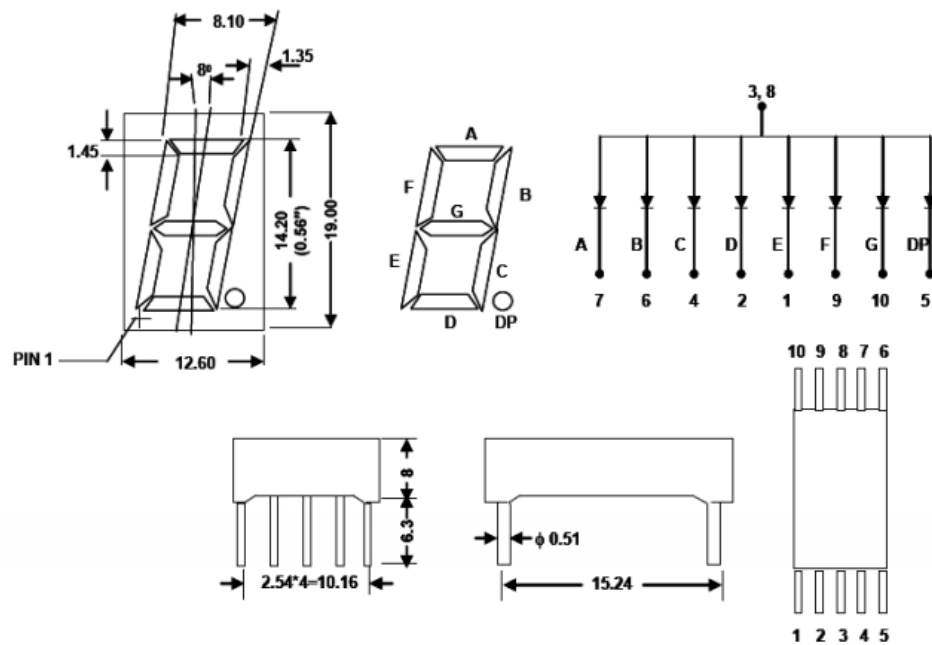


Figure 5 – The schematic for the seven segment display.

2 Problems Encountered

Below we will list some of the “several” problems we ran into.

2.1 Memory Constraints

By far, the biggest problem we encountered was the memory constraint. The 8051 has 8 kB of memory; our code base, with the exclusion of all the `malloc`s⁴, it is roughly 25 kB — a size a bit larger than the 8051 allows. Our workaround was to fight fire with fire.

Instead of “dumbing” down our game⁵ to get it to fit, we decided to have an external interface; specifically, a port sniffer. It would listen for input from the 8051, and if there is a signal on the serial port, use that as input. If not, default to the keyboard input.

Ultimately, we were unsuccessful with the port sniffer. Originally, we had tried to use a Linux port sniffer so we can just embed it into our program, `slnif`⁶. Unfortunately this port sniffer does not support “legacy” ports, and the 8051 falls in this category. So we moved onto a Windows port sniffer, `Serial Input For Windows`⁷. This too did not work, because we could only have one interface use the serial port, so we would need a dedicated socket to intercept the `COM1` port’s input — something we were not familiar with.

Ultimately, we were simply unsuccessful and had to scrap this.

2.2 Printing Lines and Columns.

We had an issue with the timing crystals of the Simon board where the board heating up would cause the serial port to start printing unrecognizable characters. This lead us to think we were causing a segfault when we were printing string literals. This caused hours of wasted time since nothing we could change would cause the board to start printing what we wanted, but one afternoon after hours of exhaustive research and help at the lead sessions the conclusion was reached that heat changes were causing the board to lose its timing and print out nonsense. The fix for this was to let the board cool off while writing code. After this change garbage outputs dropped significantly.

⁴Since `unsigned char` = 1 B, the terminal window will roughly be 20 height × 80 width, we can roughly expect 1.6 kB to be allocated on the heap; a non-insignificant amount compared to 8 kB.

⁵According to back of the hand calculations, a space-optimized version of the game would still be 7 kB. This was likely to be impossible.

⁶Can be found at <https://sourceforge.net/projects/slnif/>.

⁷Can be found at <http://www.randomnoun.com/wp/2013/02/03/serial-input-for-windows/>.

2.3 Seven Segment Display Buttons

One problem we ran into was some of the buttons use the same pins as we used for the seven segment display. The display used push/pull which makes the buttons not work, so we had to carefully chose buttons that didn't use the same pins.

3 8051 Architecture

For this project, we made sure to utilize the additional functionality we learned in the later parts of the semester — specifically, timers, serialization, and interrupts.

We spent a great deal of time with the serial communication aspect of the board; we took advantage of this the most (with respect to the 8051 architecture). We have already discussed the things that made our board unique; however, we will show an example of how our implimination of `curses`.

```
1 void move(unsigned char y, unsigned char x) {
2     unsigned char i; // counter
3     zero_cursor(); // moves the code to (0, 0)
4
5     for(i = 0; i < x; i++) {
6         cursor_jump(1, 'R'); // move cursor to the x position
7     }
8     for(i=0; i < y; i++) {
9         cursor_jump(1, 'D'); // now to the y
10    }
11 }
```

Now combining that with `addch(char)`,

```
1 void addch(unsigned char value) {
2     uart_isr();
3     uart_transmit(value);
4     while(TI==0);
5 }
```

we now have a fundamental idea to `curses`, printing anywhere!

Next, we implemented all delays with timer interrupts. For example,

```
1
2 void delay1ms() {
3     TH0=-0x0E;
```

```

4     TLO=-0x66;
5     TR0 = 1; // start
6     while(TF0 == 0); // poll to finish
7     TR0 = 0; // stop
8     TF0 = 0; // clear the finish
9 }

```

delays for 1 ms, because $(FFFF_{16} - 0E66_{16} + 1) \times 1.085 \mu s \approx 1 \text{ ms}$. As it turns out, we mostly needed millisecond precision, so we used this as the foundation for the rest of the program.

4 Individual Features

- Michael Schoen — 33% Contribution
 - Song
 - Seven Segment Display
- Abdirahman Osman — 33% Contribution
 - Port Serialization
 - Text User Interface
 - Menu
- Illya Starikov — 33% Contribution
 - Space Invaders Game
 - Keyboard

Happy Holidays
From Michael, Abdirahman, and Illya!

CS1001

Data Structures Lab



Lab #1

Illya Starikov

July 27, 2025

1 Output Redirection

1. foo
2. bar. I conclude that it is echoing the variable (foo and bar) into the file.txt.
3. `bar`
`baz`
4. `»`is prepending a newline.

2 ls and Friends

1. Outputs all directories (hidden and public) as with the subdirectories. `*` matches all directories and subdirectories.
2. It outputs all super directories, `.*` matches anything with that in the name.
3. It will annihilate all super directories, including your home directory.

3 Intermediate man usage

1. Shows the manual for the `passwd` file.
2. `man 5 passwd`, a lot of tinkering.
3. `$Home`, `$SHELL`, `$PATH`, `$LOGNAME`, `$Mail`

4 A series of tubes

1. Done.
2. `cat My Cow Story | foo` (The Build File)
3. Bash error.

5 Submitting your work

1. Done.
2. `zip -r Lab\ #2 Lab\ #2/`
3. Done.

1 That filter problem, AGAIN

1. `grep \# -v story-plain.txt`
2. `grep -v "[\#]" story-space.txt`

2 Lk m, n vwls!

1. `sed s/[aeuio]//gi story-plain.txt`

3 Counting Lines

1. `grep -P "\d{4}-01" numbers.log | wc -l`

4 Phone numbers

1. `grep -Po "([]?d{3}[]\s-]?d{3}[\s-]?d{4}" phonebook.txt`

Debuggers

Illya Starikov

March 14, 2016

1 Segfaults

1. `List::append`
2. Passed a `nullptr`. It's fixed!

2 Loopy

1. `Print`
2. They repeat. It's a looped List!
3. Fixed.

3 Math is hard

1. Fixed!

Lab #9

Illya Starikov

March 21, 2016

1 Drip, Drip, Drip, Drip

1. Memory leak, indirectly lost: 80 bytes in 5 blocks and definitely lost: 16 bytes in 1 blocks
2. Because a pointer to a list is being created, but not freed.
3. Done!

2 Imagine the possibilities!

1. possibly lost, still reachable, suppressed.
2. I was.. I knew it.
3. There is a destructor in `dtor` and `plain` only has the default (which does not take care of manual memory allocation).

3 Use wisely

1. Memory Leak.
2. You have a memory address assigned to an memory address.
3. Done.

Lab 10

Problem 1: Massif

- Bad cumulative: 2,304
- Fixed
- Good cumulative: 1,152

Problem 2: gprof

- Functions:
 - Vector::length() const
 - Vector::operator
 - int average, (Vector<int>&)
 - Vector::push_back(int)
- Because you're *not* passing by reference (by value).
- Yes.

Problem 3:

- Lines:
 - /usr/local/home/isgx2/SDRIVE/Comp Sci/CS1001/2016sp-b-lab10-isgx2/vector.h:Vector::Vector() (1x)
 - ???::std::istream::getline(char*, long) (11x)
 - /usr/local/home/isgx2/SDRIVE/Comp Sci/CS1001/2016sp-b-lab10-isgx2/vector.h:Vector::operator
- 62,627
- 9,974

Lab 14: L^AT_EX

Nathan Jarus

May 4, 2016

1 Directions

Your lab assignment today is to write the code to produce this document.

1. First, write some code.
2. Then, compile it with `pdflatex`.
3. Check the pdf to see if it looks right.
4. When you're done, submit your tex file.
5. Good luck on finals!

2 Text Formatting

Sometimes, you *may* want to emphasize some text.

3 Math

Let $i^2 = -1$, or as you may recognize it, $i = \sqrt{-1}$.

Claim: $e^{\pi i} + 1 = 0$. Why on earth would this be true?

First, what does it mean for us to raise e to an imaginary power? Well, the Maclaurin series for e^x might help us:

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots \quad (1)$$

(Hint: `\cdots` makes a centered ellipsis.)

So, plug in ix :

$$e^{ix} = \sum_{k=0}^{\infty} \frac{i^k x^k}{k!} = 1 + ix + \frac{i^2 x^2}{2} + \frac{i^3 x^3}{3!} + \frac{i^4 x^4}{4!} + \cdots \quad (2)$$

Now we get to use our nice rule that says $i^2 = -1$:

$$e^{ix} = 1 + ix - \frac{x^2}{2} - \frac{ix^3}{3!} + \frac{x^4}{4!} + \frac{ix^5}{5!} - \dots \quad (3)$$

Let's rearrange a little bit:

$$e^{ix} = \left(1 - \frac{x^2}{2} + \frac{x^4}{4} - \dots\right) + i\left(x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots\right) \quad (4)$$

If you put on your Maclaurin series glasses and squint a bit, it turns out that all this hootenanny reduces to

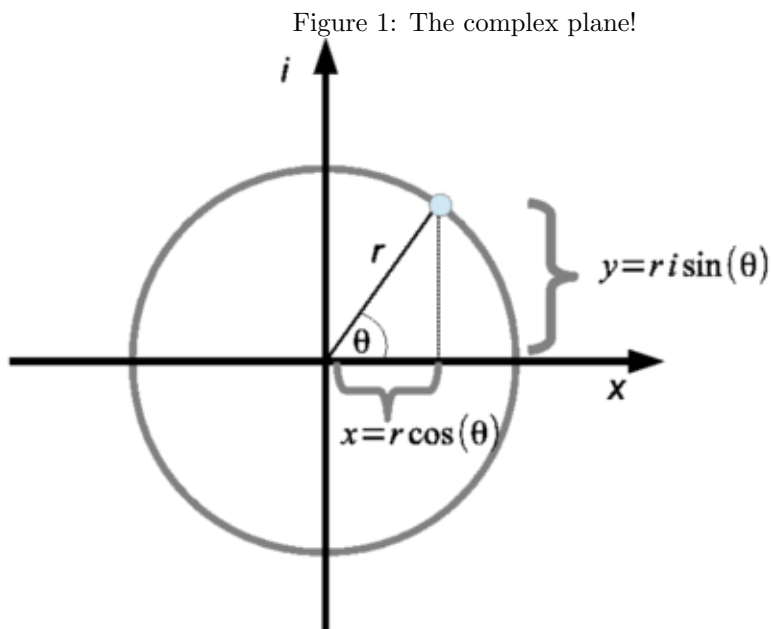
$$e^{ix} = \cos(x) + i \sin(x) \quad (5)$$

And since $\cos(\pi) = -1$ and $\sin(\pi) = 0$, $e^{\pi i} = -1$.

4 Figures

If you like drawing pretty figures, you should look at TikZ.

If you want to read more about complex numbers and fractals, read <http://acko.net/blog/how-to-fold-a-julia-fractal/>.



Project #1

Illya Starikov

Due Date: March 25, 2016

0 Introduction

- Install 2 editor plugins
- Document 5 editor features (with vim, this'll be a cinch)
- Do the job control section of project B (I use this all the time)
- What does the -exec option on find do?
- What does xargs do?
- Look up and use tmux.

1 Two Editor Plugins

For two text editors plugins, I chose (facetiously) pathogen and powerline.

1.1 pathogen.vim

Naturally when I knew I was going to install packages, my first thought was to use a package manager. Easily manipulating and deleting new packages was going to be critical, so Pathogen looked to be the proper choice.

1.1.1 What It Does

In its simplest terms, pathogen makes package management as easy as `rm -r`. My main use has been mostly to easily locate and delete new plugins that I do not use in my `bundle` directory.

1.1.2 How To Use It

After **initial setup**, pathogen actually does the rest.

1.2 Powerline

While on my quest to find text editor plugins, I stumbled upon and installed powerline. While trying to think of my second plugin to write in this report, I couldn't help but notice how much I was using Powerline.

1.2.1 What It Does

Powerline is an alternative to the standard Vim statusline.

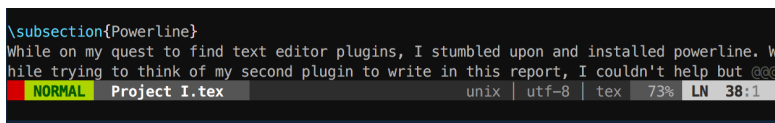


Figure 1: The Powerline In Action

It changes the configuration based on the mode that you are.

1.2.2 How To Use It

Use of powerline is automatic.

2 Five Editor Features

For my five editor features, I decided to do:

External commands (`(r ! Command)`) To run an external command outside of vim, and pipe the input back into the file. Makes inserting the date into LaTeX documents much, much faster.

Searching (`(/ ?)`) `Command F` for Vim.

Go to line (`:number`) Jumping to a specified line number.

Autocomplete (`(control+n)`) Attempts to finish the work being typed (whether it be a command or variable)

Text folding (`(z+f)a`)

3 Job Control

1. It terminates.
2. It terminates the command from the background.
3. Waits until all previous processes have finished until continueing.

4 Find `exec` ution

According to `Man`,

```
-exec utility [argument ...] ;
    True if the program named utility returns a zero value as its exit
    status. Optional arguments may be passed to the utility. The
    expression must be terminated by a semicolon (';'). If you
    invoke find from a shell you may need to quote the semicolon if the
    shell would otherwise treat it as a control operator. If the
    string '{}' appears anywhere in the utility name or the arguments
    it is replaced by the pathname of the current file. Utility will
    be executed from the directory from which find was executed.
    Utility and arguments are not subject to the further expansion of
    shell patterns and constructs.
```

Essentially `exec` allows the `exec` ution of certain programs during a command. For example, to make the use relevant to `find`, searching for an instance of a keyword in open files would `find . -exec grep KEYWORD +.`

5 *Terminal Multiplexor*

Finished, used for the job controls section (SSH to tmux to logout).

6 Xargs

Back to `Man`,

The xargs utility reads space, tab, newline and end-of-file delimited strings from the standard input and executes utility with the strings as arguments.

Any arguments specified on the command line are given to utility upon each invocation, followed by some number of the arguments read from the standard input of xargs. The utility is repeatedly executed until standard input is exhausted.

Spaces, tabs and newlines may be embedded in arguments using single (' ' ') or double (" ") quotes or backslashes ('\'). Single quotes escape all non-single quote characters, excluding newlines, up to the matching single quote. Double quotes escape all non-double quote characters, excluding newlines, up to the matching double quote. Any single character, including newlines, may be escaped by a backslash.

`xargs` allow for the manipulation of standard input, becoming quite useful with commands such as `find` or `ls`.

7 Conclusion

In conclusion, I have gained more familiarity with the command line. To take the assignment another step up, the entirety of this assignment has been created using the command line — Vim, `pdflatex`, `rm` for removing of $\text{\LaTeX}$ files, etc.

CS1200

Discrete Math



Theorem 5.3.1 For all integers

$$n, 2^{2n} - 1$$

is divisible by 3.

Proof Let the property

$$P(n)$$

be the sentence

$$2^{2n} - 1 \text{ is divisible by } 3.$$

$$2^{2n} - 1 \text{ is divisible by } 3.$$

Show that

$$P(0)$$

is true: To establish

$$P(0)$$

, we must show that

$$2^{2 \cdot 0} - 1$$

is divisible 3.

But

$$2^{2 \cdot 0} - 1 = 2^0 - 1 = 1 - 1 = 0$$

and 0 is divisible by 3 because

$$0 = 3 \cdot 0$$

. Hence

$$P(0)$$

is true.

Show that for all integers k , if $P(k)$ is true then $P(k + 1)$ is also true.

Problem: Prove by mathematical induction that $\forall n \in \mathbb{Z}^+ \cup \{0\}, 6|7^n - 1$

Solution: Proof [*By Mathematical Induction*]. Let the predicate $P(k)$ be the statement $6|7^n - 1$. We will prove that $P(n)$ is true $\forall n \in \mathbb{Z}^+ \cup \{0\}$.

Basis Step: Show that $P(0)$ is true. $P(0)$ is true because $6|7^0 - 1 = 6|1 - 1 = 6|0$

Inductive Step: Show that $\forall n \in \mathbb{Z}^+ \cup \{0\}, P(k) \rightarrow P(k+1)$. Let $k \in \mathbb{Z}^+ \cup \{0\}$ and suppose that $6|7^k - 1$ (inductive hypothesis.) Show that $6|7^{k+1} - 1$. By definition of divisibility, the inductive hypothesis is equivalent to $\exists r \in \mathbb{Z}, 7^k - 1 = 6r$

$$\begin{aligned}\text{Then } 7^{k+1} - 1 &= 7 * 7^k - 1 \\ &= 7 * 7^k - 1 \\ &= (6 + 1)7^k - 1 \\ &= 6 * 7^k + (7^k - 1) \\ &= 6 * 7^k + 6r \\ &= 6(7^k + r)\end{aligned}$$

Now $7^k + r \in \mathbb{Z}$ because the products and sums of integers are integers. Thus by divisibility, $6|7^{k+1} - 1$.

Problem: A sequence d_1, d_2, d_3 is defined by letting $d_1 = 2$ or $\forall k \in \mathbb{Z}, k \geq 2, d_k = \frac{d_{k-1}}{k}$. Prove that $\forall n \in \mathbb{Z}^+, d_n = \frac{2}{n!}$

Solution: Proof (By mathematical induction) According to the definition of $d_1, d_2, d_3, \dots$, we have $d_1 = 2$ or $\forall k \in \mathbb{Z}, k \geq 2, d_k = \frac{d_{k-1}}{k}$. Let the predicate be the equality $d_n = \frac{2}{n!}$. We will prove that $P(n)$ is true $\forall n \in \mathbb{Z}^+$

Basis Step: Show that $P(1)$ is true. Left hand side of $P(1)$ is $d_1 = 2$, right hand side of $P(1)$ is $\frac{2}{1!} = 2$, so left hand side and the right hand side of $P(1)$ are equal, so $P(1)$ is true.

Inductive Step: Show $\forall k \in \mathbb{Z}^+, P(k) \rightarrow P(k+1)$. Let $k \in \mathbb{Z}^+$ and suppose that $d_k = \frac{2}{k!}$. We must show that $d_{k+1} = \frac{2}{(k+1)!}$

$$\begin{aligned}\text{Left hand side of } P(k+1) \text{ is } d_{k+1} &= \frac{d_k}{k+1} \\ &= \frac{\frac{2}{k!}}{k+1} \\ &= \frac{2}{k!(k+1)} \\ &= \frac{2}{(k+1)!}\end{aligned}$$

Problem: Prove by Mathematical Induction $\forall n \in \mathbb{Z}, n \geq 2, 5^n + 9 < 6^n$ **Answer:** Proof by Mathematical Induction. Let the predicate $P(n)$ be the statement $5^n + 9 < 6^n$. We will prove that $P(n)$ is true $\forall n \in \mathbb{Z}, n \geq 2$.

Basis Step: Show that $P(2)$ is true because the left hand side is $5^2 + 9 = 25 + 9 = 34$ and the right hand side is $6^2 = 36$ and $34 < 36$.

Inductive Step: Show that $\forall k \in \mathbb{Z}, k \geq 2, P(k) \rightarrow P(k+1)$. Let $k \in \mathbb{Z}, k \geq 2$ and suppose $5^k + 9 < 6^k$ (The inductive hypothesis) We must show that $5^{k+1} + 9 < 6^{k+1}$.

Multiply both the sides of the inductive hypothesis with 5 yields $5(5^k + 9) < 5 * 6^k$.

Note that $5^{k+1} + 9 < 5^{k+1} + 45 = 5(5^k + 9)$ and $5 * 6^k < 6^{k+1}$.

Thus: $5^{k+1} + 9 < 5(5^k + 9)$ and $5(5^k + 9) < 5 * 6^k$ and $5 * 6^k < 6 * 6^k = 6^{k+1}$

Thus, by transitivity, $5^{k+1} + 9 < 6^{k+1}$

1 Chapter 6.1: Set Theory

Problem 5.13: Prove by Mathematical induction $\forall n \in \mathbb{Z}^+ \cup \{0\}, x, y \in \mathbb{Z}, x \neq y, x - y | x^n - y^n$

Proof Suppose $x, y \in \mathbb{Z} \vee x \neq y$. Let the predicate $P(n)$ be the statement $x - y | x^n - y^n$.

Basis Step Show that $P(0)$ is true. $P(0)$ is true because $x^0 - y^0 = 1 - 1$ and $x - y | 0$.

Inductive Step Show that $\forall k \in \mathbb{Z}^+ \cup \{0\}, P(k) \rightarrow P(k + 1)$

Let $k \in \mathbb{Z}^+ \cup \{0\}$ and suppose $x - y | x^k - y^k$. We must show that

$$x - y | x^{k+1} - y^{k+1}$$

. By rewriting the inductive hypothesis using the definition of divisibility, we have:

$$\exists r \in \mathbb{Z}, x^k - y^k = (x - y)r$$

Then:

$$\begin{aligned} & x^{k+1} - y^{k+1} \\ = & \\ & x^{k+1} - xy^k + xy^k - y^{k+1} \\ = & \\ & x(x^k - y^k) + y^k(x - y) \\ = & \\ & x(x - y)r + y^k(x - y) \\ = & \\ & (x - y)[xr + y^k] \end{aligned}$$

Now

$$x(r + y^k) \in \mathbb{Z}$$

because products and sums of integers are integers. Thus,

$$x - y | x^{k+1} - y^{k+1}$$

.

Let the set

$$A = \{x \in \mathbb{Z} | x = 5a + 2, a \in \mathbb{Z}\}$$

$$B = \{y \in \mathbb{Z} | y = 10b - 3, b \in \mathbb{Z}\}$$

$$C = \{z \in \mathbb{Z} | z = 10c + 7, z \in \mathbb{Z}\}$$

A. Is $A \subset B$? . No, because $2 = 5 * 0 + 2$, but $2 \notin B$ because if $2 \in B$ then $\exists b \in \mathbb{Z}$ such that $2 = 10b - 3$. Then $10b = 5, b = \frac{1}{2}$ which is not an integer. Counterexample found.

B. Is $B \subset A$? . Yes. Proof: Suppose $y \in B$. [*We must show that $y \in A$. By definition of A, this means that we must show $y = 5(\text{some int}) + 2$]. By definition of B, $\exists b \in \mathbb{Z}, y = 10b - 3$. Let $a = 2b - 1$. Then $a \in \mathbb{Z}$ and $5a + 2 =$

$$5(2b - 1) + 2 = 10b - 5 + 2 = 10b - 3 = y$$

To prove $A = B$, prove both are subsets of each other Let the universal set be $\mathbb{R}$

$$A \cup B = \{x \in \mathbb{R} | 6 < x \leq 8\}$$

$$A \cap B = \{x \in \mathbb{R} | -1 < x \leq 0\}$$

$$A^c = \{x \in \mathbb{R} | -3 > x \text{ or } x > 0\}$$

$$A \cap C = \emptyset$$

*Draw the venn diagram to describe $A \cap B \neq \emptyset, B \cap C = \emptyset, A \cap C = \emptyset, A \not\subset B, C \not\subset B$ *

Questions: Is $0 \in \emptyset$? No $\emptyset \neq \{\emptyset\}$? Yes. $\emptyset \in \{\emptyset\}$ $\emptyset \notin \emptyset$

EXAM QUESTION

$$\forall i \in \mathbb{Z}^+, R_i = \{x \in \mathbb{R} | 1 \leq x \leq 1 + \frac{1}{i}\} = [1, 1 + \frac{1}{i}]$$

$$\cup_{i=1}^4 R_i = R_1 \cup R_2 \cup R_3 \cup R_4 = \cup [1, 1 + \frac{1}{1}] \cup [1, 1 + \frac{1}{2}] \cup [1, 1 + \frac{1}{3}] \cup [1, 1 + \frac{1}{4}] = [1, 2]$$

$$\cap_{i=1}^4 R_i = R_1 \cap R_2 \cap R_3 \cap R_4 = \cap [1, 1 + \frac{1}{1}] \cap [1, 1 + \frac{1}{2}] \cap [1, 1 + \frac{1}{3}] \cap [1, 1 + \frac{1}{4}] = [1, 1 + \frac{1}{4}]$$

Is $\{\{w, x, v\}, \{w, y, q\}, \{p, z\}\}$ a partition of $\{p, q, u, v, w, x, y, z\}$? Yes.

Suppose $A = \{1,2\}$, $B = \{2,3\}$

b. $P(A) = P(\{1,2\}) = \{\emptyset, \{1\}, \{2\}, \{1,2\}\}$

d. $P(A \times B) = P(\{(1,2), (1,3), (2,2), (2,3)\})$

$$= \{\emptyset, \{(1,2)\}, \{(1,3)\}, \{(2,2)\}, \{(2,3)\}, \quad (1)$$

$$\{(1,2), (1,3)\}, \{(1,2), (2,2)\}, \{(1,2), (2,3)\}, \dots\} \quad (2)$$

you get the idea.

a. $P(\emptyset) = \{\emptyset\}$

b. $P(P(\emptyset)) = P(\{\emptyset\}) = \{\emptyset, \{\emptyset\}\}$

c. $P(P(P(\emptyset))) = P(\{\emptyset, \{\emptyset\}\}) = \{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}, \{\emptyset, \{\emptyset\}\}\}$

Let $A = \{a,b\}$, $B = \{1,2\}$, $C = \{2,3\}$ $A \times (B \cap C) = \{a,b\} \times \{2\} = \{(a,2), (b,2)\}$

$x = 1,3,5$, $y = a, b, c$, d $g: X \rightarrow Y$ defined by $1 = b$; $3 = b$; $5 = b$;

a. domain of g : $1, 3, 5$ co-domain of g :

b. $g(1) = b$ $g(2) = b$ $g(3) = b$

range of g : is 3 an inverse image of a ? Nah is 3 an inverse image of b ?

Yes

What is the inverse image of b ? $1,3,5$ What is the inverse image of c ? $\emptyset$

Represent g as a set of ordered pairs: $(1,b), (3,b), (5,b)$

Find all functions from $X = a,b,c$ to $Y = u$; ATTACH SKETCH

Find all the functions from $X = a,b,c$ to $Y = u, v$

Let F and G be the functions from $\mathbb{R}$ to $\mathbb{R}$ Define new functions $F - G: \mathbb{R} \rightarrow \mathbb{R}$ and $G - F: \mathbb{R} \rightarrow \mathbb{R}$ as: $\forall x \in \mathbb{R}, (F - G)(x) = F(x) - G(x)$
 $(G - F)(x) = G(x) - F(x)$

Does $F - G = G - F$ No.

Let $x = a, b, c$. Define relation **J** on $P(x)$ as follows: $\forall A, B \in P(x), A\mathbf{J}B \Leftrightarrow A \cap B \neq \emptyset$

- a) Is $\{A\}\mathbf{J}\{C\}$? No, because $\{A\} \cap \{C\} = \emptyset$
- b) Is $\{a, b\}\mathbf{J}\{b, c\}$? Yes, because $\{a, b\} \cap \{b, c\} = \{b\} \neq \emptyset$
- c) Is $\{a, b\}\mathbf{J}\{a, b, c\}$? Yes, because $\{a, b\} \cap \{a, b, c\} = \{a, b\} \neq \emptyset$

Let $A = \{3, 4, 5\}, B = \{4, 5, 6\}$ and let **S** be the “Divides” relation: $\forall (x, y) \in A \times B, x\mathbf{S}y \Leftrightarrow x|y$

- a) $S = \{(3, 6), (4, 4), (5, 5)\}$
- b) $S^{-1} = \{(6, 3), (4, 4), (5, 5)\}$

Suppose a function $F : X \rightarrow Y$ is onto but not one-to-one. Is F^{-1} a function? No, because if $F : X \rightarrow Y$ is not one-to-one, then $\exists (x_1, x_2) \in X \wedge \exists y \in Y, x_1 \neq x_2 \wedge (x_1, y) \in F \wedge (x_2, y) \in F$. But this implies $\exists x_1, x_2 \in X \wedge \exists y \in Y, x_1 \neq x_2 \wedge (y, x_1) \in F^{-1} \wedge (y, x_2) \in F^{-1}$. Consequently, F^{-1} does not satisfy property (2) of the definition of function.

Define relations **R** and **S** on $\mathbb{R}$ as follows:

R = $\{x, y\} \in \mathbb{R} \times \mathbb{R} | x^2 + y^2 = 4$ Square!

S = $\{x, y\} \in \mathbb{R} \times \mathbb{R} | x = y$ Line!

Graph $R, S, R \cup S, R \cap S$ in the Cartesian plane. *R, Circle. S, Line. R and S, both of them. R or S, just the intersection points.*

Let $A = \{2, 3, 4, 6, 7, 9\}$ and define a relation **R** on A as follows: $\forall x, y \in A, x\mathbf{R}y \Leftrightarrow 3|(x - y)$.

1. Each element has a loop to itself.
2. If element a is related to element b , then b is also related to a .
3. In each case where there is an arrow from one point to another point, and that to a third, there is an arrow going from the first to the third.

1. Reflexive? Yes.

2. Symmetric? HELL YES.

3. Transitive? HELL TO THE HELL TO THE YES

R is reflexive iff $\forall x \in A, x\mathbf{R}x$ **R** is symmetric iff $\forall x, y \in A, x\mathbf{R}y \rightarrow y\mathbf{R}x$ **R** is transitive iff $\forall x, y, z \in A, x\mathbf{R}y \wedge y\mathbf{R}z \rightarrow x\mathbf{R}z$

Let $A = \{0, 1, 2, 3\}$ and define relations **R, S, T** as follows:

R: $\{(0, 0), (0, 1), (0, 3), (1, 0), (1, 1), (2, 2), (3, 0), (3, 3)\}$

S: $\{(0, 0), (0, 2), (0, 3), (2, 3)\}$

T: $\{(0, 1), (2, 3)\}$

- a) Is **R** reflexive, symmetric, transitive? Yes, Yes, Nah.
- b) Is **S** reflexive, symmetric, transitive? No, no, yasss.
- c) Is **T** reflexive, symmetric, transitive? No, no, yasss.

Reflexivity, Symmetry, Transitivity

Properties of Congruence Modulo 3

Define relation T on $\mathbb{Z}$ as follows: $\forall m, n \in \mathbb{Z}, m T n \Leftrightarrow 3 \mid (m - n)$

1. **Reflexive?** Yes. Proof: Let $m \in \mathbb{Z}$. We must show $m T m$. Now $m - m = 0$. But $3 \mid 0$ since $0 = 3 \times 0$. Hence $3 \mid (m - m)$. Thus by definition of T , $m T m$.
2. **Symmetric?** Yes. Proof: Let $m, n \in \mathbb{Z}, m T n$. We must show that $n T m$. By definition of T . Since $m T n$, then $3 \mid (m - n)$. By definition of divides, $\exists k \in \mathbb{Z}, m - n = 3k$. Multiplication on both sides by -1 gives $n - m = 3(-k)$. Since $-k \in \mathbb{Z}, 3 \mid (n - m)$. Hence, by definition of T , $n T m$.
3. **Transitive?** Hell to the yes. Proof: Let $m, n, p \in \mathbb{Z}, m T n \wedge n T p$. We must show $m T p$. By definition of T , since $m T n \wedge n T p$, then $3 \mid (m - n) \wedge 3 \mid (n - p)$. By definition of divides, $\exists r \in \mathbb{Z}, m - n = 3r \wedge \exists s \in \mathbb{Z}, n - p = 3s$. Adding the two equations gives $(m - n) + (n - p) = 3r + 3s$ and simplifying gives $m - p = 3(r + s)$. Since $r + s \in \mathbb{Z}, 3 \mid (m - p)$. Hence, by definition of T , $m T p$.

Transitive Closure

Let A be a set and R a relation on A . The Transitive Closure of R is the relation R^t on A that satisfies the following three properties:

1. R^t is transitive.
2. $R \subset R^t$.
3. If S is any other transitive relation that contains R , then $R^t \subset S$

Example

Let $A = \{0, 1, 2, 3\}$ and Relation R on A is $R = \{(0, 1), (1, 2), (2, 3)\}$. Find the transitive closure of R .

Answer: $\{(0, 1), (1, 2), (2, 3)\} \subset R^t. R^t = \{(0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3)\}$

Equivalent Relations

Partition

A partition of set A is a set of mutually disjoint **non-empty** subsets of A whose union is A .

Give a partition of set A , the relation induced by the partition, R , is defined as follows: $\forall x, y \in A, x R y \Leftrightarrow \exists$ subsets A_i of the partition, $x, y \in A_i$.

Example

Let $A = \{0, 1, 2, 3, 4\}$ and consider the following partition of A : $\{0, 3, 4\}, \{1\}, \{2\}$. Find the relation R induced by this partition.

Answer: $R = \{(0, 0), (0, 3), (0, 4), (1, 1), (2, 2)\}$

Theorem

Let A be a set with partition and let R be the relation induced by the partition. Then R is *reflexive, symmetric, and transitive*.

Proof: Suppose A is a set with partition $\{A_1, A_2, \dots, A_n\}$. Without loss of generality, we'll assume a finite partition. Then $\forall i, j \in \{1, 2, \dots, n\}, i \neq j \Rightarrow A_i \cap A_j = \emptyset$. And $\bigcup_{i=1}^n A_i = A$. The relation R induced by the partition is defined as follows: $\forall x, y \in A, x R y \Leftrightarrow \exists A_i$ of the partition, $x, y \in A_i$.

Reflexive Let $x \in A$, Since $A_1, A_2, \dots, A_n$ is a partition of A , $\exists i \in \{1, 2, \dots, n\}, x \in A_i$. But then the statement $\exists A_i$ of the partition, $x \in A_i$ and $x \in A_i$ is true. Thus, by definition of R , $x R x$.

Symmetric Let $x, y \in A, x R y$. Then $\exists A_i$ of the partition, $x \in A_i \wedge y \in A_i$. It follows that the statement $\exists A_i$ of the partition, $y \in A_i$ is true. Hence, by definition of R , $x R y$.

Transitivity Let $x, y, z \in A, x R y \wedge y R z$. By definition of R , $\exists A_i, A_j$ of the partition $x, y \in A_i \wedge y, z \in A_j$. Suppose $A_i \neq A_j$, then $A_i \cap A_j = \emptyset$. Since $\{A_1, A_2, \dots, A_n\}$ is a partition of A . But $y \in A_i \wedge y \in A_j$. Hence $A_i \cap A_j \neq \emptyset$. That's a contradiction, thus $A_i = A_j$. It follows that $x, y, z \in A_i$. Thus, by definition of R , $x R z$.

Let A be a set and R a relation on A , R is an equivalence relation iff R is reflexive, symmetric, and transitive.

1 Partial Order Relations

1.1 Antisymmetry

No symmetry, **at all**. Let R be a relation on a set A . R is **antisymmetric** iff $\forall a, b \in A, a R b \wedge b R a \rightarrow a = b$.

1.2 Partial Order Relations

Let R be a relation defined on a set A . R is a **partial order relation** if, and only if, R is reflexive, antisymmetric, and transitive.

1.2.1 Example

Let P be the set of all people who have ever lived and define a relation R on P as follows: $\forall r, s \in P, r R s \Leftrightarrow r$ is an ancestor of s or $r = s$. Is R a partial order relation?

Yes. Proof.

R is reflexive Suppose $r \in P$. Then $r = r$, by the definition of R , $r R r$

R is antisymmetric Suppose $r, s \in P, s \in P \wedge r R s \wedge s R r$. We must show that $r = s$. By definition of R , either r is an ancestor of s , or $r = s$, and also, either s is an ancestor of r , or $s = r$. Now it is impossible for both r to be an ancestor of s and for s to be an ancestor of r . Hence one of the conditions must be false, and so $r = s$.

R is transitive Suppose $r, s, t \in P \wedge r R s \wedge s R t$. We must show that $r R t$. By definition of R , either r is an ancestor of s , or $r = s$ and either s is an ancestor of t , or $s = t$.

case 1 In case r is an ancestor of s and s is an ancestor of t , then r is an ancestor of t , and so $r R t$.

case 2 In case r is an ancestor of s and $s = t$, then r is an ancestor of t , and so $r R t$.

$\vdots$

case 3 $\dots$

case 4 $\dots$

1.2.2 Example

Define a relation r on $\mathbb{Z}$ as follows: $\forall m, n \in \mathbb{Z}, m R n \Leftrightarrow$ every prime factor of m is a prime factor of n .

No, because it's not antisymmetric. Counter example: let $m = 2 \wedge n = 4$. Then $m R n$ because every prime factor of 2 is a prime factor of 4, and $n R m$ because 4 is a prime factor of 4. But $m \neq n$, because $2 \neq 4$.

1.3 Dictionary or Lexicographic

Let A be a set with a partial order relation R , and let S be a set of strings over A . Define a relation $\preceq$ on S as follows:

For any two strings in S , $a_1a_2 \cdots a_m$ and $b_1b_2 \cdots b_n$, where m and n are positive integers,

1. $m \leq n$ $a_i = b_i$ for all $i = 1, 2, \dots, m$, then

$$a_1a_2 \cdots a_m \preceq b_1b_2 \cdots b_n.$$

2. If for some integer k with $k \leq m, k \leq n$, and $k \geq 1, a_i = b_i$ for all $i = 1, 2, \dots, k-1$ and $a_k \neq b_k$, but $a_k R b_k$ then.

$$a_1a_2 \cdots a_m \preceq b_1b_2 \cdots b_n.$$

3. If ϵ is the null string and s in any string in S , then $\epsilon \preceq s$.

If no strings are related other than by these three conditions, then $\preceq$ is a partial order relation.

1.3.1 Example

Let $A = \{a, b\}$ and suppose A has the partial order relation $R = \{(a, a), (a, b), (b, a), (b, b)\}$. Let S be the set of all strings in a's and b's and let $\preceq$ be the corresponding lexicographic order on S .

C: $\epsilon \preceq aba$? Yes, property #3.

E: $bbab \preceq bbaa$? No, property #2.

F: $ababa \preceq ababaa$? Yes, property #1.

9 Counting And Probability

9.1 Introduction

Example 9.1.12

- a) List the eight possibilities: $\{BBB, BBG, BGB, BGG, GBB, GBG, GGB, GGG\}$
b) $E_1 = \{BBG, BGB, GBB\}$. Chance of one being girl: $\frac{3}{8}$.
 $E_2 = \{BGG, GBG, GGB, GGG\}$. Chance of being *at least* two girls: $\frac{1}{2}$

Example 9.1.13

The monty hall problem.. Woo.

Theorem 9.1.1

$m, n \in \mathbf{Z} \wedge m \leq n \Rightarrow$ There are $n - m + 1$ integers from m to n inclusive.

Exercise 9.2.5

Creating Trees.

Example 9.2.15

A combination lock requires 3 selections of numbers 1 - 30.

- How many combinations? 30^3
- How many without repeats? $30 * 29 * 28$

Example 9.2.27

```
for i = 5 to 50
  for j = 10 to 20
    myFunction();
```

How many function calls? 506.

Example 9.2.3

- How many ways to rearrange the letters in Algorithm? $\frac{9!}{6!}$

9.3 Counting Elements of Disjoint Sets: The Addition Rule

0.1 The Addition Rule

Let $\{A_1, A_2, \dots, A_k\}$ be a partition of A , then $N(A) = N(A_1) + N(A_2) + \dots, N(A_k)$

0.1.1 Example 9.3.7

In some state all license plates consist of 4-6 symbols chosen from 26 letters and 10 digits.

- How many license plates without repetition? $36^4 + 36^5 + 36^6$
- No repetition? $\frac{36!}{32!} + \frac{36!}{31!} + \frac{36!}{30!}$
- How many license plates have repetition? $36^4 + 36^5 + 36^6 - [\frac{36!}{32!} + \frac{36!}{31!} + \frac{36!}{30!}]$
- What is the probability that a random license plate has a repeat?

$$\frac{36^4 + 36^5 + 36^6 - [\frac{36!}{32!} + \frac{36!}{31!} + \frac{36!}{30!}]}{36^4 + 36^5 + 36^6} \quad (1)$$

0.2 The Difference Rule

If $B \subset A \Rightarrow N(A - B) = N(A) - N(B)$

Example 9.3.9b

```

for i = 0 to 4
    for j = 1 to i
        printf("Hello, world");

```

How many runs? $\frac{n(n+1)}{2}$.

0.2.1 Probability of the Complement of an Event

If S is a finite sample space and A is an event in S , then:

$$P(A^c) = 1 - P(A)$$

Let $A = \text{Everyone Aces Exam III}$,
 $A^c = \text{not everyone aces exam III}$.

$$P(A) = 1\%$$

$$P(A^c) = 1 - 1\% = 99\%$$

Assuming all years have 365 days and all birthdays occur with equal probability, how large must n be so that in any randomly chosen group of n people, the probability that two or more have the same birthday is $\geq \frac{1}{2}$

- Probability that no two people have same birthday = $\frac{365 \times 364 \times \dots \times 365 - n + 1}{365^n}$
- Probability that two or more have same birthday = $1 - \frac{365 \times 364 \times \dots \times 365 - n + 1}{365^n}$

A college conducted a survey to explore academic interests of students. If asked students to place checks besides the following statements if they were true:

1. I was on the honor roll last term.
2. I belong to an academic club.
3. I'm majoring in multiple subjects.

Out of 100 students, 28 checked #1, 26 checked #2, and 14 checked #3, #8 checked both #1 and #2, 4 checked #1 and #3, 3 checked #2 and #3, and 2 checked all three.

- How many checked at least one? $N(H \cup C \cup D) = N(H) + N(C) + N(D) - N(H \cap C) - N(H \cap D) - N(C \cap D) + N(H \cap C \cap D) = 28 + 26 + 14 - 8 - 4 - 3 + 2 = 55$
- How many checked none? $100 - 55 = 45$
- Let H be the set of students who checked #1, C those who checked #2, and D those who checked #3. *Insert Picture*
- How many checked #1 and #2, but not #3. $N(H \cap C) - N(H \cap C \cap D) = 8 - 2 = 6$
- How many checked #2 and #3, but not #1. $N(C \cap D) - N(C \cap D \cap H) = 3 - 2 = 1$
- How many checked #2, but neither of the others? $N(C) - N(C \cap H) - N(C \cap D) + N(C \cap H \cap D) = 26 - 8 - 3 + 2 = 17$

1 8.4 The Pigeonhole Problem

- If 13 cards are selected from a standard 52-card deck, must at least 2 be of the same denomination? No, example: 2, 3, 4, ..., 10, J, Q, K, A.
- If 20 cards are selected from a standard 52-card deck, must at least 2 be of the same denomination? Yes, let X be the set consisting of the 20 selected cards and let Y be the 13 possible denominations. Define a function D from X (pigeons) to Y (Pigeonholes) by specifying $\forall x \in X, D(x) = \text{The denomination of } x$. Now X has 20 elements and Y has 13, and $20 > 13$. So by the pigeonhole principle, D is not one-to-one. Hence, $\exists x_1, x_2, x_1 \neq x_2 \wedge D(x_1) = D(x_2)$. Then x_1 and x_2 are distinct cards that have the same denomination.

Example 4.27

In a group of 2000 people, must at least 5 have the same BD? Yes, let X be the set consisting of the 2000 people (Pigeons) and Y be the set of 366 days (Pigeon Holes). Define a function $B : X \rightarrow Y$ by specifying that $\forall x \in X, B(x) = x$'s birthday. Now $2000 > 4 \times 366 = 1464$ and so by the generalized pigeonhole principle, there must be some birthday $y \in Y$ such that $B^{-1}(y)$ has at least $4 + 1 = 5$ elements. Hence at least 5 people must share the same birthday.

Example 4.30

- 12×1967 Penny
- 7×1968 Penny
- 11×1971 Penny

How many must you pick to have at least 5 from the same year? **13**.

9.5 Counting Subsets of a Set: Combinations

Theorem 9.5.1

The number of subsets of size r , called r - combinations, that can be chosen from a set of n elements, $\binom{n}{r}$ is.

$$\binom{n}{r} = \frac{P(n, r)}{r!} = \frac{n!}{r!(n - r)!} \quad (1)$$

Example 9.5.9

$$\binom{40}{6} = \frac{40!}{6!34!} = 3,838,380$$

Example 9.5.21

How many Morse symbols with 7 or fewer dots or dashes? $2^1 + 2^2 + 2^3 + 2^4 + \dots + 2^7 = 2 \times \frac{2^7 - 1}{2 - 1} = 254$

9.5 R-Combination

Let n and r be nonnegative integers with $r \leq n$. An **r-combination** of a set of n elements is a subset of r of the n elements. As indicated in Section 5.1, the symbol

$$\binom{n}{r}$$

which is read n choose r , denotes the number of subsets of size r (r -combinations) that can be chosen from a set of n elements.

9.5.1 Definition

The number of subsets of size r (or r -combinations) that can be chosen from a set of n elements, n , is given by the formula

$$\binom{n}{r} = \frac{P(n, r)}{r!}$$

or, equivalently,

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}$$

where n and r are nonnegative integers with $r \leq n$.

9.5.2 Permutations with sets of Indistinguishable Objects

Suppose a collection consists of n objects of which

n_1 are of type 1 and are indistinguishable from each other. n_2 are of type 2 and are indistinguishable from each other. $\vdots$ n_k are of type k and are indistinguishable from each other.

and suppose that $n_1 + n_2 + \cdots + n_k = n$. Then the number of distinguishable

$$\frac{\binom{n}{n_1} \binom{n-n_1}{n_2} \binom{n-n_1-n_2}{n_3} \cdots \binom{n-n_1-n_2-\cdots-n_{k-1}}{n_k}}{n_1! n_2! n_3! \cdots n_k!} \quad (1)$$

9.6 r-Combinations with Repetition Allowed

An **r-combination with repetition allowed**, or **multiset of size r** , chosen from a set X of n elements is an unordered selection of elements taken from X with repetition allowed. If $X = \{x_1, x_2, \dots, x_n\}$, we write an r -combination with repetition allowed, or multiset of size r , as $[x_{i1}, x_{i2}, \dots, x_{ir}]$, where each x_{ij} is in X and some of the x_{ij} may equal.

9.6.1 Theorem

The number of r -combinations with repetition allowed (multisets of size r) that can be selected from a set of n elements is

$$\binom{r+n-1}{r}$$

This equals the number of ways r objects can be selected from n categories of objects with repetition allowed.

Example 9.6.3

A bakery produces 6 kinds of pastry, including eclairs. Assume a supply of 20 each.

1. How many selections of 20 pastries? $N(T) = \binom{20+6-1}{20} = \binom{20}{20} = \frac{25!}{20!5!} = 53,130$
2. How many selections of 20 pastries including at least 3 eclairs? $\binom{17+6-1}{17} = \binom{22}{17} = \frac{22!}{17!5!} = 26,334$
3. How many selections of 20 pastries contain at most 2 eclairs? $N(E \leq 2) = N(T) - N(E \geq 3) = 53,130 - 26,334 = 26,796$

Example 9.6.9

```

for k = 1 to n
  for j = k to n
    for i = j to n

```

Number of iterations of inner loops is the same as the number of integer triples (i, j, k) where $1 \leq k \leq j \leq i \leq n$. Triples can be represented as a string of $n - 1$ vertical bars and three crosses indicated which integers from 1 to n are included.

$$\binom{3+n-1}{3} = \binom{n+2}{3} = \frac{n(n+1)(n+3)}{6}$$

Example 9.6.12

$y_1 + y_2 + y_3 + y_4 = 30$. How many selections? $\binom{30+4-1}{30}$

Example 9.6.13

$y_1 + y_2 + y_3 + y_4 = 30, y_1 \geq 2, y_2 \geq 2, y_3 \geq 2, y_4 \geq 2$. $\binom{22+4-1}{22}$

Example 9.6.18

A large pile of coins consist of pennies, nickels, dimes, and quarters.

- How many different collections of 30 coins can be selected? $N(T) = \binom{30+4-1}{30} = \binom{33}{30} = 5,456$.
- If the pile contains only 15 quarters, but 30 of each other kind of coin, how many collections of 30 can be chosen? $N(Q \geq 16) =$

Let T be the set of selections of 30 coins, $Q \leq 15$ the set without most 15 quarters, and $Q \geq 16$, The set without at least 16 quarters. Then:

$$T = Q \leq 15 \cup Q \geq 16$$

$$Q \leq 15 \cap Q \geq 16 = \emptyset$$

$$\begin{aligned}
N(T) &= N(Q \leq 15) + N(Q \geq 16) - N(Q \leq 15 \cap Q \geq 16) \\
N(Q \geq 16) &= \binom{14+4-1}{14} = \binom{17}{14} = 680. \\
N(Q \leq 15) &= 5,456 - 680 = 4,776
\end{aligned}$$

- 20 Dimes, 30 of each of the rest. How many 30? $T = D \leq 20 \cup D \geq 21$
Thus: $N(T) = N(D \leq 20) + N(D \geq 21)$
 $N(D \geq 21) = \binom{9+4-1}{9} = \binom{12}{9} = 220$
 $N(D \leq 20) = N(T) - N(D \geq 21) = 5,456 - 220 = 5,236$
- 15 dimes, 15 quarters, 30 of each of the rest. How many 30? $N(Q \leq 15 \cap D \leq 20) = N(T) - N(Q \geq 16 \cup D \geq 21)$
 $= 5,456 - 900 = 4,556$

$$\begin{aligned}
N(Q \geq 16 \cap D \geq 21) &= N(Q \geq 16) + N(D \geq 21) - N(Q \geq 16 \cup D \geq 21) \\
&= 680 + 220 - 0 = 900
\end{aligned}$$

9.1 Graph Theory

9.2 Graph Theory I

10.1.1 Example 10.1.16

A graph of Vertices of degrees 1, 1, 4, 4 and 6 how many edges does the graph have?

$$\text{Total Edges} = \frac{\text{Add Them Up}}{2} \quad (1)$$

$$= \frac{16}{2} \quad (2)$$

$$= 8 \quad (3)$$

10.1.2 Example 10.1.11g

Draw or explain why doesn't exist: Graph with four vertices of degree 1, 1, 1, and 4.

Doesn't exist, because is of odd total degree.

10.1.3 Example 10.1.20

Draw or explain why doesn't exist: graph with 4 vertices of degrees 1, 2, 3, 4.

10.1.4 Example 10.1.26b

Find all subgraphs of the picture depicted in the book. There's a lot.

10.1.5 Example 10.1.36c

Draw $K_{3,4}$. Three on left, four on right, connect the dots.

10.1.6 Example 10.1.37c

Is the following graph bipartite? If so, redraw.

10.1.7 Example 10.1.3gb

THIS WILL PROBABLY BE ON EXAM.

Find the complement of a graph in the book.

10.1.8 Example 10.1.44

- In a simple graph, must every vertex have degree that is less than the number of vertices in the graph?

Yes, in a simple graph. This is so because if G is a simple graph with n vertices and v is a vertex of G , then since G has no parallel edges, v can be joined by at most a single edge to each of the other $n - 1$ vertices of G and since G has no loops, G cannot be joined to itself. Thus the max degree of v is $n - 1$.

- Can there be a simple graph that has 4 vertices each of different degree?

Nah. Suppose there is a simple graph with 4 vertices each of which has different degree. By the previous proof, no vertex can have a greater degree than 3. And, of course, no vertex can have negative degree. Thus, the only possibility is that they have degree 0, 1, 2, and 3. The vertex of degree 3 has to be connected to all 3 other vertices, but this contradicts that one of the other degrees is 0, thus the assumption was false and no such graph exists.

10.2 Trails, Paths, and Circuits

10.2.1 Example 10.2.2

Look up in book.

- a) Just walk.
- b) Simple Circuit, trail.
- c) Closed Walk.
- d) Circuit.
- e) Trail.
- f) Path.

10.3 Matrix Representation of Graphs

Find the adjacency matrix for: *insert picture*

Column x rows, from x to:

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

10.3.1 Example 3b

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

10.3.2 Example 6b

Given:

$$\begin{bmatrix} 0 & 2 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Determine without drawing whether this graph is connected.

10.3.3 Example 13

Let 0 denote the matrix:

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Find 2×2 matrices A and B such that $A \neq B$, $B \neq 0$, and $AB \neq 0$, but $BA = 0$.

Do this some other time.

10.3 Matrix Representations of Graphs

10.3.1 Example 10.3.19

$$\text{Let } A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix}$$

$$A^2 = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 6 & 3 & 3 \\ 3 & 2 & 2 \\ 3 & 2 & 5 \end{bmatrix}$$

$$A^3 = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 6 & 3 & 3 \\ 3 & 2 & 2 \\ 3 & 2 & 5 \end{bmatrix} = \begin{bmatrix} 15 & 9 & 15 \\ 9 & 5 & 8 \\ 15 & 8 & 8 \end{bmatrix}$$

Assume A is an adjacency matrix. How many walks of length 2 from v_1 to v_3 ? 3.

Assume A is an adjacency matrix. How many walks of length 3 from v_1 to v_3 ? 15.

Examine the calculations you performed to find five walks of length 2 from v_3 to v_3 . Then draw G and find the walks by visual inspection.

$$2 \cdot 2 + 1 \cdot 1 + 0 \cdot 0 \quad (1)$$

Left edges from v_3 to v_1 , right is edges from v_1 to v_3 . $2 \times 2 =$ walk of length 2 from v_3 to v_3 via v_1 .

10.2 Matrix Representations of Graphs

Graph in book.

- How many from a to c ? **4**.
- How many trails from a to c ? **4 + 3 + 2 + 1**
- How many walks from a to c ? ∞

10.2.1 Example 10.3.6b

An edge whose removal disconnects graph it is part of is called an bridge. Find the bridge in:

The graph is in the book

- $\langle v_7, v_8 \rangle$
- $\langle v_3, v_4 \rangle$
- $\langle v_1, v_2 \rangle$

10.2.2 Example 10.2.8d

Find the number of connected components in: *Image in book.* **2.**

10.2.3 Euler Circuit

Let G be a graph. An Euler circuit for G is a circuit that contains every vertex and every every edge of G . That is, an Euler Circuit for G is a sequence of adjacent vertices and edges in G that has at least one edge, starts and ends in the same vertex, uses every vertex of G at least once, and uses every edge exactly edge exactly one.

10.2.4 Theorem 10.2.2

If a graph has an Euler circuit, then every vertex has even degree.

10.2.5 Theorem 10.2.3

if a graph G is connected and the degree of every vertex is even, then G has an Euler circuit.

10.2.6 Example 10.2.6

Insert graph here.

10.3 Matrix Representations of Graphs

An $m \times n$ (read “ m by n ”) **matrix A over a set S** is a rectangular array of elements of S arranged into m rows and n columns.

We write $A = (a_{ij})$

10.3.1 Adjacency Matrix

Let G be a directed graph with ordered vertices $v_1, v_2, \dots, v_n$. The adjacency matrix of G is the $n \times n$ matrix $A = (a_{ij})$ over the set of nonnegative integers such that

$$a_{ij} = \text{the number of arrows from } v_i \text{ to } v_j \text{ for all } i, j = 1, 2, \dots, n. \quad (1)$$

10.3.2 Adjacency Matrix

Let G be an undirected graph with ordered vertices $v_1, v_2, \dots, v_n$. The adjacency matrix of G is the $n \times n$ matrix $A = (a_{ij})$ over the set of nonnegative integers such that

$$\begin{aligned} a_{ij} &= \text{the number of edges connecting } v_i \text{ and } v_j \\ &\text{for all } i, j = 1, 2, \dots, n. \end{aligned} \quad (2)$$

10.3.3 Symmetric

An $n \times n$ square matrix $A = (a_{ij})$ is called **symmetric** if, and only if, for all $i, j = 1, 2, \dots, n$,

$$a_{ij} = a_{ji} \quad (3)$$

10.3.4 Identity Matrix

For each positive integer n , the $n \times n$ **identity matrix**, denoted $I_n = (\delta_{ij})$ or just **I** (if the size of the matrix is obvious from context), is the $n \times n$ matrix in which all the entries in the main diagonal are 1's and all other entries are 0's. In other words,

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad \text{for all } i, j = 1, 2, \dots, n$$

10.3.5 The Number of Walks

If G is a graph with vertices $v_1, v_2, \dots, v_m$ and A is the adjacency matrix of G , then for each positive integer n and for all integers $i, j = 1, 2, \dots, m$,

the ij th entry of A_n = the number of walks of length n from v_i to v_j .

10.5 Trees

10.5.1 Terminology

A graph is said to be **circuit-free** if, and only if, it has no circuits. A graph is called a **tree** if, and only if, it is circuit-free and connected. A **trivial tree** is a graph that consists of a single vertex. A graph is called a **forest** if, and only if, it is circuit-free and not connected.

10.5.2 Vertex-Degree Relation

Any tree that has more than one vertex has at least one vertex of degree 1.

10.5.3 Terminal Vertex

Let T be a tree. If T has only one or two vertices, then each is called a **terminal vertex**. If T has at least three vertices, then a vertex of degree 1 in T is called a **terminal vertex** (or a **leaf**), and a vertex of degree greater than 1 in T is called an **internal vertex** (or a **branch vertex**).

10.5.4 Edges And Vertices

For any positive integer n , any tree with n vertices has $n - 1$ edges.

10.6 Connected, Circuit

If G is any connected graph, C is any circuit in G , and any one of the edges of C is removed from G , then the graph that remains is connected.

Test II Study Guide

Prepared By: Illya Starikov

July 27, 2025

5 Sequences, Mathematical Induction, and Recursion

5.1 Sequences

If $a_m, a_{m+1}, a_{m+1} \dots$ and $b_m, b_{m+1}, b_{m+1} \dots$ are sequences of real numbers and c is any real number, then the following equations hold for any integer $n \geq m$:

1. $\sum_{k=m}^n a_k + \sum_{k=m}^n b_k = \sum_{k=m}^n (a_k + b_k)$
2. $c \times \sum_{k=m}^n a_k = \sum_{k=m}^n c \times a_k$
3. $(\prod_{k=m}^n a_k) \times (\prod_{k=m}^n b_k) = \prod_{k=m}^n a_k \times b_k$

5.2 Mathematical Induction I

n choose r

For all integers n and r with $0 \leq r \leq n$,

$$\binom{n}{k} = \frac{n!}{r!(n-r)!}$$

Sum of the First n Integers

For all integers $n \geq 1$,

$$1 + 2 + \dots + n = \frac{n(n+1)}{2}$$

Sum of Geometric Sequence

For any real number r except 1, and an integer $n \geq 0$,

$$\sum_{i=0}^n r^i = \frac{r^{n+1} - 1}{r - 1}$$

6 Set Theory

6.1 Set Theory: Definitions and the Element Method of Proof

Proper Subset

A is a **proper subset** of $B \Leftrightarrow$

1. $A \subseteq B$, and
2. there is at least one element in B that is not in A .

Element Argument: The Basic Method for Proving That One Set Is a Subset of Another

Let sets X and Y be given. To prove that $X \subseteq Y$,

1. **suppose** that x is a particular but arbitrary chosen element of X ,
2. **show** that x is an element of Y .

Equals

Given sets A and B , A **equals** B , written $\mathbf{A} = \mathbf{B}$, if, and only if, every element of A is in B and every element of B is in A .

Symbolically:

$$A = B \quad \Leftrightarrow \quad A \subseteq B \text{ and } B \subseteq A.$$

Union, Intersection, Difference, Complement

Let A and B be subsets of a universal set U .

1. The **union** of A and B , denoted $A \cup B$, is the set of all elements that are in at least one of A or B .
2. The **intersection** of A and B , denoted $A \cap B$, is the set of all elements that are common to both A and B .
3. The **difference** of B minus A (or **relative complement** of A in B), denoted $B - A$, is the set of all elements that are in B and not A .
4. The **complement** of A , denoted A^c , is the set of all elements in U that are not in A .

Symbolically,

$$\begin{aligned}A \cup B &= \{x \in U \mid x \in A \text{ or } x \in B\}, \\A \cap B &= \{x \in U \mid x \in A \text{ and } x \in B\}, \\B - A &= \{x \in U \mid x \in B \text{ and } x \notin A\}, \\A^c &= \{x \in U \mid x \notin A\}.\end{aligned}$$

Disjoint

Two sets are called **disjoint** if, and only if, they have no elements in common. Symbolically:

$$A \text{ and } B \text{ are disjoint} \quad \Leftrightarrow \quad A \cap B = \emptyset$$

Partition

A finite or infinite collection of nonempty sets $A_1, A_2, A_3 \dots$ is a **partition** of a set A if, and only if,

1. A is the union of all the A_i
2. The sets $A_1, A_2, A_3, \dots$ are mutually disjoint. (Not-overlapping)

Power Set

Given a set A , the **power set** of A , denoted $\mathcal{P}(A)$, is the set of all subsets of A .

Cartesian Product

Given sets $A_1, A_2, \dots, A_n$ the **Cartesian product** of $A_1, A_2, \dots, A_n$, denoted $A_1 \times A_2 \times \dots \times A_n$, is the set of all ordered n -tuples $(a_1, a_2, \dots, a_n)$ where $a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n$.
Symbolically:

$$A_1 \times A_2 \times \dots \times A_n = \{(a_1, a_2, \dots, a_n) \mid a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n\}$$

In particular,

$$A_1 \times A_2 = \{(a_1, a_2) \mid a_1 \in A_1 \text{ and } a_2 \in A_2\}$$

is the Cartesian product of A_1 and A_2

7 Functions

7.1 Functions Defined on General Sets

Function

A **function f from a set X to a set Y** , denoted $f : x \rightarrow Y$, is a relation from X , the **domain**, to Y , the **co-domain**, that satisfies two properties: (1) every element x is related to some element in Y , and (2) no element in X is related to more than one element in Y .

Logarithms And Logarithmic Function

Let b be a positive real number with $b \neq 1$. For each positive real number x , the **logarithm with base b of x** , written $\log_b x$, is the exponent to which b must be raised to obtain x . Symbolically,

$$\log_b x = y \quad \Leftrightarrow \quad b^y = x$$

The **logarithmic function with base b** is the function from $\mathbf{R}^+$ to $\mathbf{R}$ that takes each positive real number x to $\log_b x$

One-to-One

Let F be a function from a set X to a set Y . F is **one-to-one** (or **injective**) if, and only if, for all element x_1 and x_2 in X ,

$$\text{if } F(x_1) = F(x_2), \text{ then } x_1 = x_2,$$

or, equivalently,

$$\text{if } x_1 \neq x_2, \text{ then } F(x_1) \neq F(x_2)$$

Symbolically,

$$F : X \rightarrow Y \text{ is one-to-one} \Leftrightarrow \forall x_1, x_2 \in X, \text{ if } F(x_1) = F(x_2) \text{ then } x_1 = x_2$$

This can be read as: A function $F : X \rightarrow Y$ is *not* one-to-one **if, and only if**, there exist elements $x_1, x_2 \in X$ with $F(x_1) = F(x_2)$ and $x_1 \neq x_2$.

Proof of One-To-One

$$f(x) = 4x - 1$$

Suppose x_1 and x_2 are real numbers such that $f(x_1) = f(x_2)$.

$$4x_1 - 1 = 4x_2 - 1 \tag{1}$$

$$4x_1 = 4x_2 \tag{2}$$

$$x_1 = x_2 \tag{3}$$

$$\tag{4}$$

Onto

Let F be a function from a set X to a set Y . F is **onto** (or **surjective**) if, and only if, given any element y in Y , it is possible to find an element x in X with the property that $y = F(x)$.

Symbolically:

$$F : X \rightarrow Y \text{ is onto} \Leftrightarrow \forall y \in Y, \exists x \in X \text{ such that } F(x) = y.$$

In other words, $F : X \rightarrow Y$ is *not* onto **if, and only if**, $\exists y$ in Y such that $\forall x \in X, F(x) \neq y$.

Onto Proof Example

$$f(x) = 4x - 1$$

Let $y \in \mathbf{R}$. Let $x = (y + 1)/4$. Then x is a real number since sums and quotients (other than by 0) of a real numbers are real numbers. It follows:

$$\begin{aligned} f(x) &= f\left(\frac{y+1}{4}\right) \\ &= 4 \times f\left(\frac{y+1}{4}\right) - 1 \\ &= (y+1) - 1 \\ &= y \end{aligned}$$

Bijection

A **one-to-one correspondence** (or **bijection**) from a set X to a set Y is a function $F: X \rightarrow Y$ that is *both one-to-one and onto*.

Inverse Image

Suppose $F: X \rightarrow Y$ is a one-to-one correspondence; that is, suppose F is one-to-one and onto. Then there is a function $F^{-1}: Y \rightarrow X$ that is defined as follows:

Just take the inverse. It's basic algebra.

8 Relations

8.1 Relations on Sets

Relation

Let $\mathbf{R}$ be a relation from A to B . Define the inverse relation R^{-1} from B to A as follows:

$$R^{-1} = \{(x, y) \in B \times A \mid (y, x) \in R\}$$

This is equivalent to: $\forall x \in A$ and $y \in B, (y, x) \in R^{-1} \Leftrightarrow (x, y) \in R$

Relation on Sets

A **relation on a set A** is a relation from A to A .

n -ary relation

Given sets $A_1, A_2, \dots, A_n$, an **n -ary relation** R on $A_1 \times A_2 \times \dots \times A_n$. The special casts of 2-ary, 3-ary, and 4-ary relations are called **binary**, **ternary**, and **quaternary relations**, respectively.

8.2 Reflexivity, Symmetry, and Transitivity

Reflexivity, Symmetry, and Transitivity

Let R be a relation on a set A .

Reflexive R is reflexive if, and only if, for all $x \in A$, $x R x$

- R is reflexive $\Leftrightarrow$ for all x in A , $(x, x) \in R$.
- **Reflexive:** Each element is related to itself.
- R is **not reflexive** $\Leftrightarrow$ there is an element x in A such that $x \not R x$ [that is, such that $(x, x) \notin R$].

Symmetric R is symmetric if, and only if, for all $x, y \in A$, **if** $x R y$ then $y R x$

- R is symmetric $\Leftrightarrow$ for all x and y in A , **if** $(x, y) \in R$ then $(y, x) \in R$
- **Symmetric:** If any one element is related to any other element, then the second element is related to the first.
- R is **not symmetric** $\Leftrightarrow$ there are elements x and y in A such that $x R y$ but $y \not R x$ [that is, such that $(x, y) \in R$ but $(y, x) \notin R$].

Transitive R is transitive if, and only if, for all $x, y, z \in A$, **if** $x R y$ and $y R z$ then $x R z$

- R is transitive $\Leftrightarrow$ for all x, y , and z in A , **if** $(x, y) \in R$ and $(y, z) \in R$ then $(x, z) \in R$.

- **Transitive** If any one element is related to a second and that second element is related to a third, then the first element is related to the third.
- R is **not transitive** $\Leftrightarrow$ there are elements x, y , and z in A such that $x R y$ and $y R z$ but $x \not R z$ [that is, such that $(x, y) \in R$ and $(y, z) \in R$ but $(x, z) \notin R$]

The Transitive Closure of a Relation

Let A be a set and R a relation on A . the **transitive closure** of R is the relation R^t on A that satisfies the following three properties:

1. R^t is transitive.
2. $R \subset R^t$.
3. If S is any other transitive relation that contains R , then $R^t \subset S$

8.3 Equivalence Relations

Give a partition of a set A , the **relation induced by the partition**, R , is defined on A as follows: For all $x, y \in A$,

$x R y$ there is a subset A_i of the partition such that both x and y are in A_i

Equivalence Relation

Let A be a set and R a relation on A . R is an **equivalence relation** if, and only if, R is reflexive, symmetric, and transitive.

Equivalence Class

Suppose A is a set and R is an equivalence relation on A . For each element a in A , the **equivalence class of a** , denoted $[a]$ and called the **class of a**

Representative

Suppose R is an equivalence relation on a set A and S is an equivalence class of R . A representative of the class S is any element a such that $[a] = S$.

Congruence Modulo

Let m and n be integers and let d be a positive integer. We say that **m is congruent to n modulo d** and write

$$m \equiv n \pmod{d}$$

if, and only if,

$$d \mid (m - n)$$

Symbolically:

$$m \equiv n \pmod{d} \Leftrightarrow d \mid (m - n)$$

8.5 Partial Order Relations

Antisymmetric

Let R be a relation on a set A . R is **antisymmetric** if, and only if,

$$\text{for all } a \text{ and } b \text{ in } A, \quad \text{if } a R b \text{ and } b R a \text{ then } a = b.$$

Partial Order Relation

Let R be a relation defined on a set A . R is a **partial order relation** if, and only if, R is reflexive, antisymmetric, and transitive.

General Partial Order

Because of the special paradigmatic role played by the $\leq$ relation in the study of partial order relations, the symbol $\preceq$ is often used to refer to a general partial order relation, and the notation $x \preceq y$ is read “ x is less than or equal to y ” or “ y is greater than or equal to x .”

Dictionary or Lexicographic

Let A be a set with a partial order relation R , and let S be a set of strings over A . Define a relation $\preceq$ on S as follows:

For any two strings in S , $a_1a_2 \cdots a_m$ and $b_1b_2 \cdots b_n$, where m and n are positive integers,

1. $m \leq n$ $a_i = b_i$ for all $i = 1, 2, \dots, m$, then

$$a_1 a_2 \cdots a_m \preceq b_1 b_2 \cdots b_n.$$

2. If for some integer k with $k \leq m, k \leq n$, and $k \geq 1, a_i = b_i$ for all $i = 1, 2, \dots, k-1$ and $a_k \neq b_k$, but $a_k R b_k$ then

$$a_1 a_2 \cdots a_m \preceq b_1 b_2 \cdots b_n.$$

3. If ε is the null string and s in any string in S , then $\varepsilon \preceq s$.

If no strings are related other than by these three conditions, then $\preceq$ is a partial order relation.

Maximal, Minimal, Greatest, Least

A *maximal element* in a partially ordered set is an element that is greater than or equal to every element to which it is comparable. (There may be many elements to which it is not comparable.) A *greatest element* in a partially ordered set is an element that is greater than or equal to every element in the set (so it is comparable to every element in the set). Minimal and least are defined similarly.

9 Counting And Probability

9.1 Introduction

9.1.1 Sample Space

A **sample space** is the set of all possible outcomes of a random process or experiment. An **event** is a subset of a sample space.

9.1.2 Equally Likely Probability Formula

If S is a finite sample space in which all outcomes are equally likely and E is an event in S , then the **probability of E**, denoted $\mathbf{P(E)}$, is

$$P(E) = \frac{\text{the number of outcomes in } E}{\text{the total number of outcomes in } S} \quad (1)$$

The notation is as follows: For any finite set A , $N(A)$ denotes the number of elements in A .

9.1.3 The Number of Elements in a List

If m and n are integers and $m \leq n$, then there are $n - m + 1$ integers from m to n inclusive.

9.2 Possibility Trees and the Multiplication Rule

9.2.1 The Multiplication Rule

If an operation consists of k steps and

the first step can be performed n_1 ways, the second step can be performed in n_2 ways [*regardless of how the first step was performed*]; the k th step can be performed in n_k ways [*regardless of how the preceding steps were performed*],

then the entire operation can be performed in $n_1 n_2 \cdots n_k$ ways.

9.2.2 Number of Permutations

For any integer n with $n \geq 1$, the number of permutations of a set with n elements is $n!$.

9.2.3 R-Permutation

An **r-permutation** of a set of n elements is an ordered selection of r elements taken from the set of n elements. The number of r-permutations of a set of n elements is denoted $\mathbf{P(n, r)}$.

If n and r are integers and $1 \leq r \leq n$, then the number of r -permutations of a set of n elements is given by the formula

$$P(n, r) = n(n-1)(n-2) \cdots (n-r+1)$$

or, equivalently,

$$P(n, r) = \frac{n!}{(n-r)!}$$

9.3 Counting Elements of Disjoint Sets: The Addition Rule

The Addition Rule

Suppose a finite set A equals the union of k distinct mutually disjoint subsets $A_1, A_2, \dots, A_k$. Then:

$$N(A) = N(A_1) + N(A_2) + \cdots + N(A_k).$$

The Difference Rule

If A is a finite set and B is a subset of A , then:

$$N(A - B) = N(A) - N(B).$$

Formula for the probability of the complement of an Event

If S is a finite sample space and A is an event in S , then:

$$P(A^c) = 1 - P(A)$$

The Inclusion/Exclusion Rule for Two or Three Sets

If A , B and C are any finite sets, then

$$N(A \cup B) = N(A) + N(B) - N(A \cap B)$$

and

$$N(A \cup B \cup C) = N(A) + N(B) + N(C) - N(A \cap B) - N(A \cap C) - N(B \cap C) + N(A \cap B \cap C).$$

Pigeonhole Principle

A function from one finite set to a smaller finite set cannot be one-to-one: There must be at least two elements in the domain that have the same image in the co-domain.

Generalized Pigeonhole Principle

For any function f from a finite set X with n elements to a finite set Y with m elements and for any positive integer k , if $k < n/m$, there is some $y \in Y$ such that y is the image of at least $k + 1$ distinct elements of X .

9.3.1 Generalized Pigeonhole Principle (Contrapositive Form)

For any function f from a finite set X with n elements to a finite set Y with m elements and for any positive integer k , if for each $y \in Y$, $f^{-1}(y)$ has at most k elements, then X has at most km elements; in other words, $n \leq km$.

9.3.2 The Pigeonhole Principle

For any function f from a finite set X with n elements to a finite set Y with m elements, if $n > m$, then f is not one-to-one.

9.3.3 One-to-One and Onto for Finite Sets

Let X and Y be finite sets with the same number of elements and suppose f is a function from X to Y . Then f is one-to-one if, and only if, f is onto.

9.5 R-Combination

Let n and r be nonnegative integers with $r \leq n$. An **r-combination** of a set of n elements is a subset of r of the n elements. As indicated in Section 5.1, the symbol

$$\binom{n}{r}$$

which is read n choose r, denotes the number of subsets of size r (r -combinations) that can be chosen from a set of n elements.

9.5.1 Definition

The number of subsets of size r (or r -combinations) that can be chosen from a set of n elements, n , is given by the formula

$$\binom{n}{r} = \frac{P(n, r)}{r!}$$

or, equivalently,

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}$$

where n and r are nonnegative integers with $r \leq n$.

9.5.2 Permutations with sets of Indistinguishable Objects

Suppose a collection consists of n objects of which

n_1 are of type 1 and are indistinguishable from each other. n_2 are of type 2 and are indistinguishable from each other. $\vdots$ n_k are of type k and are indistinguishable from each other.

and suppose that $n_1 + n_2 + \cdots + n_k = n$. Then the number of distinguishable

$$\frac{\binom{n}{n_1} \binom{n-n_1}{n_2} \binom{n-n_1-n_2}{n_3} \cdots \binom{n-n_1-n_2-\cdots-n_{k-1}}{n_k}}{n_1! n_2! n_3! \cdots n_k!} \quad (2)$$

9.6 r-Combinations with Repetition Allowed

An **r-combination with repetition allowed**, or **multiset of size r** , chosen from a set X of n elements is an unordered selection of elements taken from X with repetition allowed. If $X = \{x_1, x_2, \dots, x_n\}$, we write an r -combination with repetition allowed, or multiset of size r , as $[x_{i1}, x_{i2}, \dots, x_{ir}]$, where each x_{ij} is in X and some of the x_{ij} may equal.

9.6.1 Theorem

The number of r -combinations with repetition allowed (multisets of size r) that can be selected from a set of n elements is

$$\binom{r+n-1}{r}$$

This equals the number of ways r objects can be selected from n categories of objects with repetition allowed.

9.7 Graphs: Definitions and Basic Properties

9.7.1 Graph Terminology

A **graph** G consists of two finite sets: a nonempty set $V(G)$ of **vertices** and a set $E(G)$ of **edges**, where each edge is associated with a set consisting of either one or two vertices called its **endpoints**. The correspondence from edges to endpoints is called the **edge-endpoint function**.

An edge with just one endpoint is called a **loop**, and two or more distinct edges with the same set of endpoints are said to be **parallel**. An edge is said to **connect** its endpoints; two vertices that are connected by an edge are called **adjacent**; and a vertex that is an endpoint of a loop is said to be **adjacent to itself**.

An edge is said to be **incident on** each of its endpoints, and two edges incident on the same endpoint are called **adjacent**. A vertex on which no edges are incident is called **isolated**.

9.7.2 Directed Graph / Digraph

A **directed graph**, or **digraph**, consists of two finite sets: a nonempty set $V(G)$ of vertices and a set $D(G)$ of directed edges, where each is associated with an ordered pair of vertices called its **endpoints**. If edge e is associated with the pair (v, w) of vertices, then e is said to be the **(directed) edge** from v to w .

9.7.3 Simple Graph

A **simple graph** is a graph that does not have any loops or parallel edges. In a simple graph, an edge with endpoints v and w is denoted v, w .

9.7.4 Complete Graph of n Vertices

Let n be a positive integer. A **complete graph on n vertices**, denoted K_n , is a simple graph with n vertices and exactly one edge connecting each pair of distinct vertices.

9.7.5 Complete Bipartite Graph on (m, n) Vertices

Let m and n be positive integers. A complete bipartite graph on (m, n) vertices, denoted $K_{m,n}$, is a simple graph with distinct vertices $v_1, v_2, \dots, v_m$ and $w_1, w_2, \dots, w_n$ that satisfies the following properties: For all $i, k = 1, 2, \dots, m$ and for all $j, l = 1, 2, \dots, n$,

1. There is an edge from each vertex v_i to each vertex w_j .
2. There is no edge from any vertex v_i to any other vertex v_k
3. There is no edge from any vertex w_j to any other vertex w_l

9.7.6 Subgraph

A graph H is said to be a **subgraph** of a graph G if, and only if, every vertex in H is also a vertex in G , every edge in H is also an edge in G , and every edge in H has the same endpoints as it has in G .

9.7.7 Degree

Let G be a graph and v a vertex of G . The **degree of v** , denoted $\deg(v)$, equals the number of edges that are incident on v , with an edge that is a loop counted twice. The **total degree of G** is the sum of the degrees of all the vertices of G .

9.7.8 The Handshake Theorem

If G is any graph, then the sum of the degrees of all the vertices of G equals twice the number of edges of G . Specifically, if the vertices of G are $v_1, v_2, \dots, v_n$, where n is a nonnegative integer, then

$$\text{The total degree of } G = \deg(v_1) + \deg(v_2) + \dots + \deg(v_n) \quad (3)$$

$$= 2 \times (\text{the number of edge of } G) \quad (4)$$

This means that **the total degree of a graph is even.**

9.7.9 Proposition

In any graph there are an even number of vertices of odd degree.

9.7.10 Complement

If G is a simple graph, the **complement of G** , denoted G' , is obtained as follows: The vertex set of G' is identical to the vertex set of G . However, two distinct vertices v and w of G' are connected by an edge if, and only if, v and w are not connected by an edge in G .

9.8 Trails, Paths, and Circuits

9.8.1 Definitions

Let G be a graph, and let v and w be vertices in G .

Walk A **walk from v to w** finite alternating sequence of adjacent vertices and edges of G . Thus a walk has the form

$$v_0 e_1 v_1 e_2 \dots v_{n-1} e_n v_n \quad (5)$$

where the v 's represent vertices, the e 's represent edges, $v_0 = v, v_n = w$, and for all $i = 1, 2, \dots, n$, v_{i-1} and v_i are the endpoints of e_i . The **trivial walk from v to v** consists of the single vertex v .

Trail A **trail** from v to w is a walk from v to w that does not contain a repeated edge.

Path A **path** from v to w is a trail that does not contain a repeated vertex.

Closed Walk A **closed walk** is a walk that starts and ends at the same vertex.

Circuit A **circuit** is a closed walk that contains at least one edge and does not contain a repeated edge.

Simple Circuit A **simple circuit** is a circuit that does not have any other repeated vertex except the first and last.

	Repeated Edge?	Repeated Vertex?	Starts and Ends at Same Point?	Must Contain At Least One Edge?
Walk	allowed	allowed	allowed	no
Trail	no	allowed	allowed	no
Path	no	no	no	no
Closed Walk	allowed	allowed	yes	no
Circuit	no	allowed	yes	yes
Simple Circuit	no	first and last only	allowed	yes

9.8.2 Connected

Let G be a graph. Two **vertices v and w of G are connected** if, and only if, there is a walk from v to w . The **graph G is connected** if, and only if, given *any* two vertices v and w in G , there is a walk from v to w . Symbolically,

$$G \text{ is connected} \Leftrightarrow \forall \text{ vertices } v, w \in V(G), \exists \text{ a walk from } v \text{ to } w. \quad (6)$$

Let G be a graph.

- If G is connected, then any two distinct vertices of G can be connected by a path.
- . If vertices v and w are part of a circuit in G and one edge is removed from the circuit, then there still exists a trail from v to w in G .
- If G is connected and G contains a circuit, then an edge of the circuit can be removed without disconnecting G .

A graph H is a connected component of a graph G if, and only if,

1. H is subgraph of G ;
2. H is connected; and
3. no connected subgraph of G has H as a subgraph and contains vertices or edges that are not in H .

9.8.3 Euler Circuit

Let G be a graph. An **Euler circuit** for G is a circuit that contains every vertex and every edge of G . That is, an Euler circuit for G is a sequence of adjacent vertices and edges in G that has at least one edge, starts and ends at the same vertex, uses every vertex of G at least once, and uses every edge of G exactly once.

If a graph has an Euler circuit, then every vertex of the graph has positive even degree. If some vertex of a graph has odd degree, then the graph does not have an Euler circuit.

9.8.4 Euler Algorithm

If a graph G is connected and the degree of every vertex of G is a positive even integer, then G has an Euler circuit.

Proof: Suppose that G is any connected graph and suppose that every vertex of G is a positive even integer. *[We must find an Euler circuit for G .]* Construct a circuit C by the following algorithm:

Step 1: Pick any vertex v of G at which to start. *[This step can be accomplished because the vertex set of G is nonempty by assumption.]*

Step 2: Pick any sequence of adjacent vertices and edges, starting and ending at v and never repeating an edge. Call the resulting circuit C . *[This step can be performed for the following reasons: Since the degree of each vertex of G is a positive even integer, as each vertex of G is entered by traveling on one edge, either the vertex is v itself and there is no other unused edge adjacent to v , or the vertex can be exited by traveling on another previously unused edge. Since the number of edges of the graph is finite (by definition of graph), the sequence of distinct edges cannot go on forever. The sequence can eventually return to v because the degree of v is a positive even integer, and so if an edge connects v to another vertex, there must be a different edge that connects back to v .]*

Step 3: Check whether C contains every edge and vertex of G . If so, C is an Euler circuit, and we are finished. If not, perform the following steps.

Step 3a: Remove all edges of C from G and also any vertices that become isolated when the edges of C are removed. Call the resulting subgraph G . *[Note that G may not be connected (as illustrated in Figure 10.2.4), but every vertex of G has positive, even degree (since removing the edges of C removes an even number of edges from each vertex, the difference of two even integers is even, and isolated vertices with degree 0 were removed.)]*

Step 3b: Pick any vertex w common to both C and G . *[There must be at least one such vertex since G is connected. (See exercise 44.) (In Figure 10.2.4 there are two such vertices: u and w .)]*

Step 3c: Pick any sequence of adjacent vertices and edges of G , starting and ending at w and never repeating an edge. Call the resulting circuit C . *[This can be done since each vertex of G has positive, even degree and G is finite. See the justification for step 2.]*

Step 3d: Patch C and C together to create a new circuit C as follows: Start at v and follow C all the way to w . Then follow C all the way

back to w . After that, continue along the untraveled portion of C to return to v . [The effect of executing steps 3c and 3d for the graph of Figure 10.2.4 is shown in Figure 10.2.5.]

Step 3e: Let $C = C \cup E$ and go back to step 3.

Since the graph G is finite, execution of the steps outlined in this algorithm must eventually terminate. At that point an Euler circuit for G will have been constructed. (Note that because of the element of choice in steps 1, 2, 3b, and 3c, a variety of different Euler circuits can be produced by using this algorithm.)

9.8.5 Theorem

A graph G has an Euler circuit if, and only if, G is connected and every vertex of G has positive even degree.

9.8.6 Corollary

Let G be a graph, and let v and w be two distinct vertices of G . There is an Euler path from v to w if, and only if, G is connected, v and w have odd degree, and all other vertices of G have positive even degree.

9.8.7 Hamiltonian Circuit

Given a graph G , a Hamiltonian circuit for G is a simple circuit that includes every vertex of G . That is, a Hamiltonian circuit for G is a sequence of adjacent vertices and distinct edges in which every vertex of G appears exactly once, except for the first and the last, which are the same.

If a graph G has a Hamiltonian circuit, then G has a subgraph H with the following properties:

1. H contains every vertex of G .
2. H is connected.
3. H has the same number of edges as vertices.
4. Every vertex of H has degree 2.

9.9 Matrix Representations of Graphs

An $m \times n$ (read “ m by n ”) **matrix A over a set S** is a rectangular array of elements of S arranged into m rows and n columns.

We write $A = (a_{ij})$

9.9.1 Adjacency Matrix

Let G be a directed graph with ordered vertices $v_1, v_2, \dots, v_n$. The adjacency matrix of G is the $n \times n$ matrix $\mathbf{A} = (a_{ij})$ over the set of nonnegative integers such that

$$a_{ij} = \text{the number of arrows from } v_i \text{ to } v_j \text{ for all } i, j = 1, 2, \dots, n. \quad (7)$$

9.9.2 Adjacency Matrix

Let G be an undirected graph with ordered vertices $v_1, v_2, \dots, v_n$. The adjacency matrix of G is the $n \times n$ matrix $A = (a_{ij})$ over the set of nonnegative integers such that

$$\begin{aligned} a_{ij} &= \text{the number of edges connecting } v_i \text{ and } v_j \\ &\text{for all } i, j = 1, 2, \dots, n. \end{aligned} \quad (8)$$

9.9.3 Symmetric

An $n \times n$ square matrix $A = (a_{ij})$ is called **symmetric** if, and only if, for all $i, j = 1, 2, \dots, n$,

$$a_{ij} = a_{ji} \quad (9)$$

9.9.4 Identity Matrix

For each positive integer n , the $n \times n$ **identity matrix**, denoted $I_n = (\delta_{ij})$ or just $\mathbf{I}$ (if the size of the matrix is obvious from context), is the $n \times n$ matrix in which all the entries in the main diagonal are 1's and all other entries are 0's. In other words,

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad \text{for all } i, j = 1, 2, \dots, n$$

,

9.9.5 The Number of Walks

If G is a graph with vertices $v_1, v_2, \dots, v_m$ and A is the adjacency matrix of G , then for each positive integer n and for all integers $i, j = 1, 2, \dots, m$,

the ij th entry of A^n = the number of walks of length n from v_i to v_j .

9.5 Trees

9.5.1 Terminology

A graph is said to be **circuit-free** if, and only if, it has no circuits. A graph is called a **tree** if, and only if, it is circuit-free and connected. A **trivial tree** is a graph that consists of a single vertex. A graph is called a **forest** if, and only if, it is circuit-free and not connected.

9.5.2 Vertex-Degree Relation

Any tree that has more than one vertex has at least one vertex of degree 1.

9.5.3 Terminal Vertex

Let T be a tree. If T has only one or two vertices, then each is called a **terminal vertex**. If T has at least three vertices, then a vertex of degree 1 in T is called a **terminal vertex** (or a **leaf**), and a vertex of degree greater than 1 in T is called an **internal vertex** (or a **branch vertex**).

9.5.4 Edges And Vertices

For any positive integer n , any tree with n vertices has $n - 1$ edges.

9.6 Connected, Circuit

If G is any connected graph, C is any circuit in G , and any one of the edges of C is removed from G , then the graph that remains is connected.

10.6 Rooted Trees

A **rooted tree** is a tree in which there is one vertex that is distinguished from the others and is called the **root**. The **level** of a vertex is the number of edges along the unique path between it and the root. The **height** of a rooted tree is the maximum level of any vertex of the tree. Given the root or any internal vertex v of a rooted tree, the **children** of v are all those vertices that are adjacent to v and are one level farther away from the root than v . If w is a child of v , then v is called the **parent** of w , and two distinct vertices that are both children of the same parent are called **siblings**. Given two distinct vertices v and w , if v lies on the unique path between w and the root, then v is an **ancestor** of w and w is a **descendant** of v .

10.7 Binary Tree

A **binary tree** is a rooted tree in which every parent has at most two children. Each child in a binary tree is designated either a **left child** or a **right child** (but not both), and every parent has at most one left child and one right child. A **full binary tree** is a binary tree in which each parent has exactly two children.

Given any parent v in a binary tree T , if v has a left child, then the **left subtree** of v is the binary tree whose root is the left child of v , whose vertices consist of the left child of v and all its descendants, and whose edges consist of all those edges of T that connect the vertices of the left subtree. The **right subtree** of v is defined analogously.

10.7.1 Theorem

If k is a positive integer and T is a full binary tree with k internal vertices, then T has a total of $2k + 1$ vertices and has $k + 1$ terminal vertices.

10.7.2 Height And Terminal Vertices

For all integers $h \geq 0$, if T is any binary tree with of height h and t terminal vertices, then

$$t \leq 2^h. \quad (10)$$

Equivalently,

$$\log_2 t \leq h \quad (11)$$

10.8 Spanning Trees and Shortest Paths

A **spanning tree** for a graph G is a subgraph of G that contains every vertex of G and is a tree.

10.8.1 Proposition

- Every connected graph has a spanning tree.
- Any two spanning trees for a graph have the same number of edges.

10.8.2 Terminology

A **weighted graph** is a graph for which each edge has an associated positive real number **weight**. The sum of the weights of all the edges is the **total weight** of the graph. A **minimum spanning tree** for a connected weighted graph is a spanning tree that has the least possible total weight compared to all other spanning trees for the graph. If G is a weighed graph and e is an edge of G , then $\mathbf{w}(e)$ denotes the weight of e and $\mathbf{w}(G)$ denotes the total weight of G .

10.8.3 Kruskal's Algorithm

Iteration Number	Edge Considered	Weight	Action Take

COMP SCI 1200 Section B FS2015 Assignment 4

Illya Starikov

November 8, 2015

To compile: Run a standard makefile:

```
fg++ *.cpp -o foo
```

Notes:

- Any two int are acceptable (within the scope of the primitive int).
- Negative integers are also accepted, but they will be converted to positive.

COMP SCI 1200 Section B FS2015 Assignment 5

Illya Starikov

November 17, 2015

To compile: Run a standard makefile:

```
fg++ *.cpp -o foo
```

Notes:

- Any two non-negative int are acceptable (within the size of the primitive int).

CS1510

Data Structures



1 Graph

Is there a path from root a to node b ? Let's algorithmize!

```
// Recursive
pathSearch(Graph G, start, goal, visited nodes) {
    if start == goal { return true; }

    add started to visited nodes
    for every neighbor x of start not in visited nodes
        solved = pathSearch(G, x, goal, visited nodes)
        if solved {
            return true
        }

    return false;
}

// Not Recursive
pathSearch(Graph G, start, goal, Visited) {
    stack of nodes S
    push(S, start)

    while (S is not empty) {
        X = top(S)
        pop(S);

        if (X == goal)
            return true

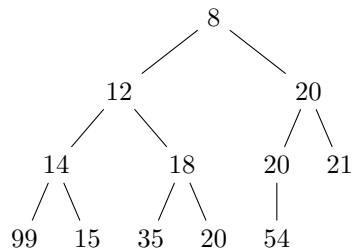
        add X to visited

        for every neighbor y of X, not in visited
            push (S, y)
    }

    return false;
}
```


Heap

- Binary Search Tree
- All but the bottom level is complete.
- From every node x , x is less than its two children.
- Member functions.
 - $\text{top}(T) = 8$
 - $\text{insert}(T, x)$
 - $\text{remove}(T) = T = \text{the top of the heap}$
- Maintenance Functions
 - Percolate Up, during insertion.
 - * Let the item bubble up.
 - Percolate Down, during removal.
 - * Like a stone sinking through a viscous liquid.



D.S. ArrayHeap

```
class ArrayHeap {
    T *data;
    int m_max, m_size;

public:
    const T& top() {
        if (m_size != 0) {
            return m_data[0];
        } else {
            cerr << "Shit."
        }
    }

    void insert(const T& x) {
```

```

        if (m_max == m_size) {
            grow();
        }

        int hole = m_size;
        m_size++;

        while (hole > 0 && x < m_data[(hole- 1) / 2]) {
            m_data[hole] = m_data[(hole - 1) / 2];
            hole = (hole - 1)/2;
        }

        m_data[hole] = x;
    }

    void remove() {
        if (m_size == 0) { return; }
        int hole = 0;
        m_size--;

        We can continue this over break.
    }
};

```

- Homework 5 is on stacks
- Go to megaminer.

1 Recursive Object

An object which potentially consists or defined in terms of itself. Recursion is used to define things such as:

- Sets
- Functions
- Other objects

Power of Recursion:

- Describe an infinite object
- Through finite means.

Recursive Definition

- Base Case
- Recursive Case

Example: Set of all strings of balanced parenthesis.

Base Case: $()$ is in the set.

Recursive Case: if s is in the set, then $s()$, $()s$, and (s) are also in the set.

$$Fibonacci = \begin{cases} fib(1) = 1 \\ fib(2) = 1 \\ fib(n) = fib(n-1) + fib(n-2) \end{cases} \quad (1)$$

$$Factorial = \begin{cases} 1! = 1 \\ n! = n \times (n-1) \end{cases} \quad (2)$$

1.1 Recursive Algorithms

- Base Case
 - Direct solution to a small problem instance.
- Recursive Case
 - Decompose problem into smaller instances.
 - Solve smaller instances.
 - Construct solution from smaller solutions.

1.1.1 Triomino Problem

Suppose we have four possible tiles made of three squares.

Problem: Cover $2^n \times 2^n$ board, where one tile is a hole with triominoes.

- $n = 4, 2^n = 16$

Morales gives example.

- split board in 4 equal parts.
- Place triomino across 3 split parts without a hole.
- Solve each subpart.

```
void foo() {  
    int x;  
  
    foo();  
}
```

```
quicksort(array, left, right) { // assuming left < or = right  
    if (left = right) {  
        return; // Base Case  
    }  
}
```

```
pivot = a[(left + right) / 2];
```

```

int i = left;
int j = right;

repeat
    while (a[i] < pivot) { i++; }
    while (a[j] > pivot) { j--; }
    if (i < j) {
        swap(a[i], a[j]);
        i++;
        j--;
    }
while (j > i);

quicksort(a, i + 1, r);
quicksort(a, l, i - 1)

}

```

2 Recursive Backtracking

```

try
    initialize choices
do
    select choice
    if choice is valid
        record choice
        if solution complete
            success!
        else
            try next step
            if next step succeeds
                success!
            else
                cancel record
    while !success & more choices available

path_find(grid, int row, int col) {

```

```

    for choice c in {N, NE, E}
        nrow = row after c;
        ncol = col after c;

        if (grid[nrow][ncol] != obstacle && nrow, ncol is in bounds)
            record nrow, ncol;

        if (grid[row][column] == cake!) {
            return true;
        } else {
            solve = path_find(grid, nrow, ncol)
            if (solve) {
                return true;
            } else {
                record C;
            }
        }
    }

    return false
}

bool valid(grid, int r, int c) {
    if (c < 0 || r >= N) {
        return false;
    }
    if (c < 0 || c >= N) {
        return false;
    }
    if (grid[r, c] == obstacle) {
        return false;
    }

    return true;
}

for (int i = 0; i < 3; c++) {
    nrow = col + dir[c][0];
    ncol = col + dir[c][1];

```

```
    if (valid(grid, nrow, ncol, n)
}
```

1 Tree

- A tree is a collection of elements with a hierarchical relation over such elements.
- The actual definition is as follows: The empty collection is a tree (the empty tree). A single element is usually called a node. Node is a tree.
- If n is a node and $T_1, T_2, T_3, \dots, T_n$ are trees, then n related to $T_1, T_2, \dots, T_n$:
 - n is called the root
 - $T_1 \dots T_n$ are called the subtrees of n .
- A *path* is a sequence of nodes. $\langle a_1, a_2, a_3, \dots, a_n \rangle$ when n_{i+1} is a parent of n_i $0 \leq i < n$.
- The *depth* of a node a is the number of nodes in the path from a to the root.
- the *height* of a tree is the greatest depth of a node in the tree.

Note Every tree has only one root.

Child The root of each subtree T_1, T_n are called the children of n

n Is called the parent of the root of each subtree T_1, T_n

Siblings If two roots have the same parent they are called siblings.

Leaf A leaf is a node with no children.

Degree The degree of a node a is the number of children of a .

- The degree of a tree is the highest degree of a node in the tree.

decendent/ancestor If there is a path from node a to node z then a is called a decendent of z .
 z is called an ancestor of a . The root is every node's ancestor.

1.1 Binary Search Tree

- Binary: Degree 2
- Search Conditions
 - find(T, x)
 - getMin(T)
 - getMax(T)
 - insert(T)
 - remove(T, x)


```

// Data Structure BinaryTree

template <classname T>
class TreeNode {
    T m_data;
    TreeNode *m_right;
    TreeNode *m_left;
};

// To use recursion, functions cannot be a method of TreeNode
const T& getMin(TreeNode *t) {
    if (t == nullptr) { /* error */ }

    if (t -> m_left == nullptr) {
        return t-> m_data;
    } else {
        return getMin(t -> m_left);
    }
}

const T& getMax(TreeNode *t) {
    if (t == nullptr) { /* error */ }
    TreeNode *p = t;

    while (p -> m_right != nullptr) {
        p = p -> m_right;
    }

    return p -> m_data;
}

bool T& find(TreeNode *t, const T& x) {
    if (t == nullptr) { return false; }
    if (t -> m_data == x) { return true; }

    if (x < t -> m_data) {
        return find(t -> m_left, x);
    } else if (x > t -> m_data) {
        return find(t -> m_right, x);
    }
}

void insert(TreeNode * &t, const T& x) {
    if (t == nullptr) {
        t = new TreeNode;
        t -> m_right = nullptr;
    }
}

```

```

        t -> m_left = nullptr;
    } else (x < t -> m_data) {
        insert(t -> m_left);
    } else if (x > t -> m_data) {
        insert(t -> m_right);
    } else {
        return; // This is a duplicate. No duplicates allowed.
    }
}

void remove(TreeNode * &t, const T& x) {
/*
3 Cases:
- No Children
- One Child
- Two Children

To remove, you have choice. Max of Left or Min of right.
*/

    if (t == nullptr) {
        return;
    }
    if (x < t -> m_data) {
        remove(t -> left, x);
    } else if (x > t -> m_data) {
        remove(t -> right x);
    } else {
        // FOUND X!
        if (t -> m_right == nullptr && t -> m_left == nullptr) {
            // No children
            delete t;
            t = nullptr;
        } else if (t -> m_right == nullptr ||
                    t -> m_left == nullptr) {
            TreeNode *temporary = t -> m_right;
            if (temporary == nullptr) {
                temporary = t -> left;
            }

            // Now, temporary points to the child
            delete x;
            t = temporary;
        } else {
            // X has two children
            t -> m_data = getMin(t -> m_right);

```

```

        remove(t -> m_right, t -> m_data);
    }
}

```

- collection of objects
- repetition is not allowed
 - SETS!
- Why? Who cares? WHY NOT VECTORS?
 - find()
 - * Find of size 500. $\log_2 500 = 8.9$
 - * 5000. $\log_2 5000 = 12.2$
 - * 5 Million. $\log_2 5\text{Million} = 22.25$
 - * Most important operations.
 - insert()
 - * $\log_2 n$
 - remove()
 - * $\log_2 n$

index at i , $\text{right}(i) = 2i + 1$, $\text{left}(i) = 2i + 2$. The parent of is $\frac{i-2}{2}$

CS2300

Databases



1 Categories of Data Models

- Conceptual data model
- Implementation data model
- Physical data model

2 Three Schema Architecture

Internal Schema Physical data model.

- Physical storage structure
- Data storage details, access path

Conceptual Schema Conceptual or implementation data model.

- Structure of the entire database
- Entities, data types, relationships

External Schema Describe the various user views.

- Usually uses the same data model as the conceptual schema.

3 Data Independence

3.1 Logical Data Independence

The capacity to change the conceptual schema without having to change the external schemas and their associated application programs.

3.2 Physical Data Independence

The capacity to change the internal schema without having to change the conceptual schema.

Homework #1

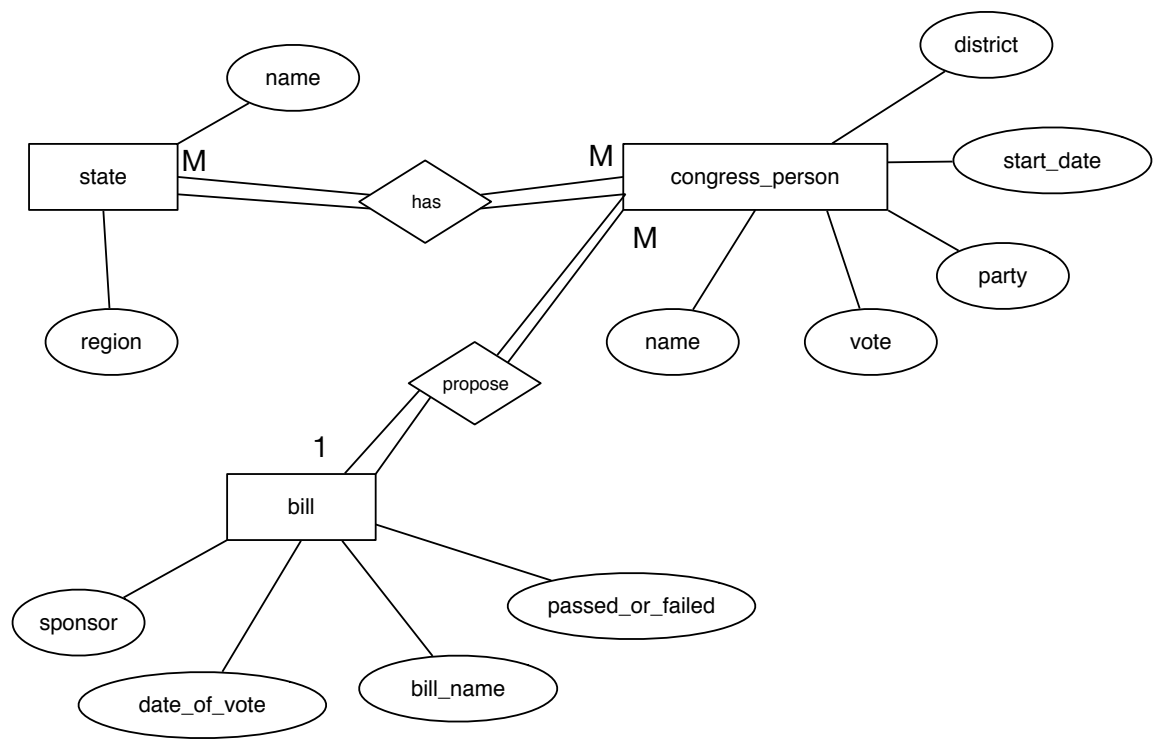
Claire Trebing, Jason Young, Illya Starikov

Due Date: February 04, 2016

Question 1

- a Section's CRoom (Building and Room Number).
- b Section's DayTime and Year.
- c Section's SecID for every Year.

Question 2



Question 3

a Strong Entity Types

- Bank
- Accounts
- Loans
- Customer

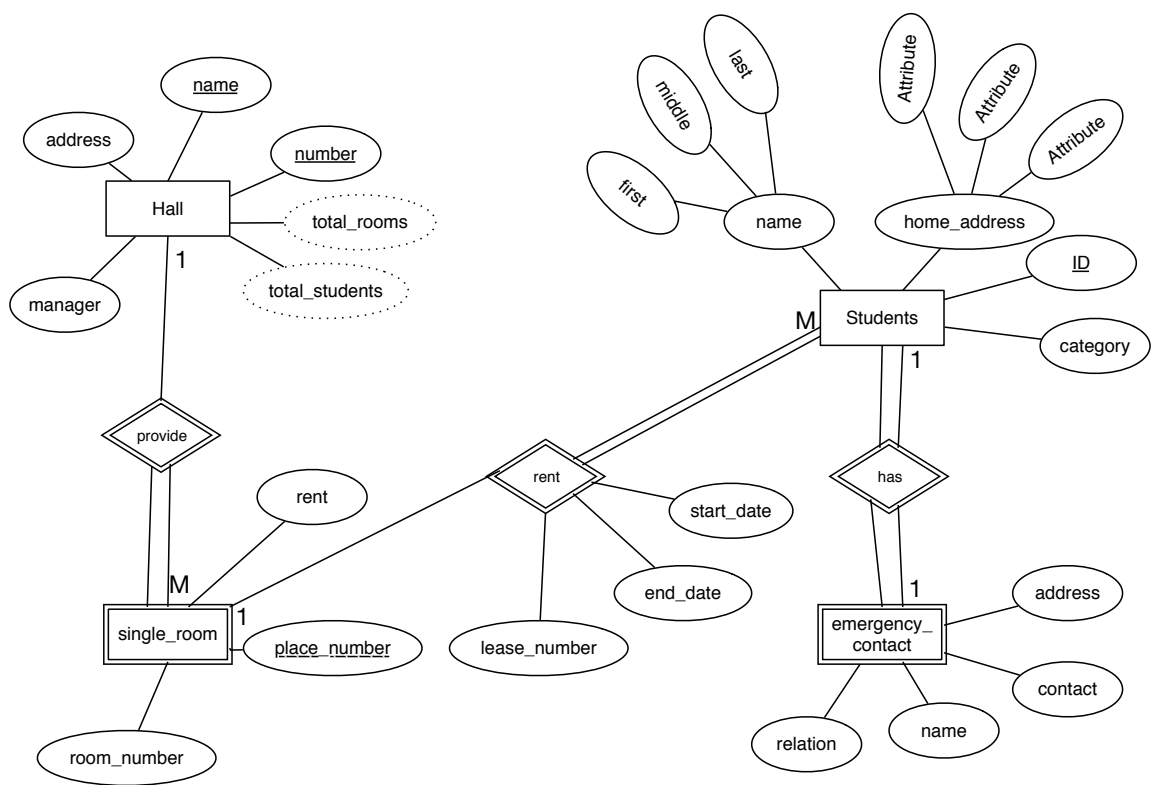
b Weak Entity Type

- Yes, the **Bank Branch** is a weak entity type. The partial key is the **Branch Number** and the identifying relationship is the **Branches** relationship.

c Constraints

- This information specifies that the branch number of a bank branch is only unique within a bank. For example, a bank branch in Bank A may have the same branch number as a bank branch in Bank B, but a bank branch in Bank A would not have the same branch number as any other bank branch in Bank A.

Question 4



Homework #2

Claire Trebing, Jason Young, Illya Starikov

February 25, 2016

Question #1

- Valid Operation
- Valid Operation
- Key Constraint (Key #4 is already taken)
- Key Constraint, Not Null Constraint (Key cannot be null)
- Valid Operation
- Valid Operation
- Integrity Constraint (Did not update WORKS_ON or DEPENDANT)
- Integrity Constraint (Did not update WORK_ON)
- Valid Operation
- Integrity Constraint (Did not update WORKS_ON)
- Valid Operation

Question #2

Grader note: the rho (ρ) is too small in L^AT_EX to be distinguished from a subscript, so instead P is used in it's place.

- $\Pi_{\text{Fname, Minit, Lname}}(\sigma_{\text{Dno} = '5' \wedge \text{hours} = > 10}(\text{project} \bowtie_{\text{Dname} = 'rojectX'} \text{on}))$
- $\Pi_{\text{Fname} = \text{Dependent\_Name}}(\text{mployee} \bowtie_{\text{Dependent}})$
- $\Pi_{\text{Fname, LName}}(\text{mployee} \div \Pi_{\text{Super\_SSN}})(\sigma_{\text{Fname} = 'Frankli'})$
- $\Pi_{\text{Lname, Sum\_Hours}}(s_{\text{name}} \mathcal{F}_{\text{sum hours}}(\text{ROJECT} \bowtie_{\text{ESSN} = \text{SSN}}) \begin{matrix} \text{ } \end{matrix})$
- $\text{mployee1} \rightarrow \text{mployee} \bowtie_{\text{ESSN} = \text{SSN} \text{ork\_On}}$

- f. $\Pi_{Lname}(\Pi_{Lname, SSN} \text{employee} \setminus \Pi_{\_ \{\text{Lname}, SSN\}} (\begin{array}{l} employee \\ \rho_{ESSN = SSN} \text{orks\_On} \end{array}))$
- g. $\Pi_{Fname, Average Salary} (DName \mathcal{F}_{average salary} P_{Dno = Dnumber} \text{employee} \setminus \bowtie (\begin{array}{l} Department \\ Depar \end{array}))$
- h. $\text{Sex} \mathcal{F}_{average salary} (\sigma_{sex = 'female'} \text{employee} \setminus)$
- i. $\text{epartment1} \setminus \leftarrow \rho_{\_ \{\text{Number} = DNorject\}} \setminus \bowtie \rho_{\_}$
- j. $\Pi_{Lname}(\text{employee} \setminus \bowtie \_ \{\text{SSN} = MGR\_SSN\}) \setminus \begin{array}{l} Department \\ SSN) - \Pi_{Lname} \text{employee} \setminus \end{array}$

Question #3

- `ust# \end{verb} (\begin{verb} ORDER \end{verb})`
- `rder \end{verb} (\begin{verb} RDERTEM,HIPMENT \end{verb})`
- `tem# \end{verb} (\begin{verb} ORDER \end{verb}, \begin{verb} TEM
)`
- `arehouse# \end{verb} (\begin{verb} SHIPMENT \end{verb})`

Assumptions

- Table headers with the same name are not unique.

Homework #4

Databases and File Structures

Illya Starikov, Jason Young, Claire Trebing

Due Date: April 21, 2016

1 Closures, Candidate Keys

- $A^+ = \{A\}$
- $AB^+ = \{A, B, C, D, E\}$ **Candidate Key**
- $AC^+ = \{A, B, D, C, E\}$ **Candidate Key**
- $ACD^+ = \{A, B, C, D, E\}$ **Candidate Key**
- $B^+ = \{B, D\}$
- $BC^+ = \{A, B, C, D, E\}$ **Candidate Key**
- $C^+ = \{C\}$
- $CD^+ = \{A, B, C, D, E\}$ **Candidate Key**
- $E^+ = \{A, B, C, D, E\}$ **Candidate Key**
- $DE^+ = \{A, B, C, D, E\}$ **Candidate Key**

2 Functional Dependence Equivalence

No, because there is no occurrence of $C \rightarrow D$ in G , but there is one in F .

3 Minimal Cover

Step 1

- $A \rightarrow B, A \rightarrow C, A \rightarrow D$
- $E \rightarrow G, E \rightarrow H$
- $C \rightarrow D$
- $AE \rightarrow J$

Step 2

- $A \rightarrow B, A \rightarrow C, A \rightarrow D$
- $E \rightarrow G, E \rightarrow H$
- $C \rightarrow D$
- $AE \rightarrow J$

Step 3

- $A \rightarrow B, A \rightarrow C$
- $E \rightarrow G, E \rightarrow H$
- $C \rightarrow D$
- $AE \rightarrow J$

4 3NF and BCNF

1. $F = \{AB \rightarrow C, C \rightarrow D, C \rightarrow A\}$, *Candidate Keys*: $(AB), (BC)$
 - No, because in the functional dependency $C \rightarrow D$, D is not a part of any candidate key and A is not a super key.
 - No, because $\{C \rightarrow D\}$ is neither trivial nor a superkey.
2. $F = ACE \rightarrow BD, B \rightarrow C$, *Candidate Key*: (ACE)
 - Yes, because ACE is a super key and C is part of the candidate key.
 - No, because $\{B \rightarrow C\}$ is neither trivial nor a superkey.

1 Itinerary

1. Introduction
2. Schedules
3. Ideas
 - Store Front
 - Publishing Medium
 - Social Network
4. Expectations
 - Good Grade?
 - Minimal Effort?
 - Learning?
5. Technology
 - Client Side (Front End)
 - HTML5/CSS3
 - JavaScript
 - * Ajax
 - * React
 - * Angular.js
 - * CoffeeScript
 - * Ember.js
 - * Backbone.js
 - * Knockout.js
 - Server Side (Back End)
 - Node.JS
 - Google Go
 - Scala
 - Clojure
 - Rust
 - Perl
 - Ruby
 - PHP
 - C++
 - C#
 - Database
 - SQL
6. Wrap Up

2 Notes

- Everyone is in the Google Calendar, just add your schedule so we know when to meet.

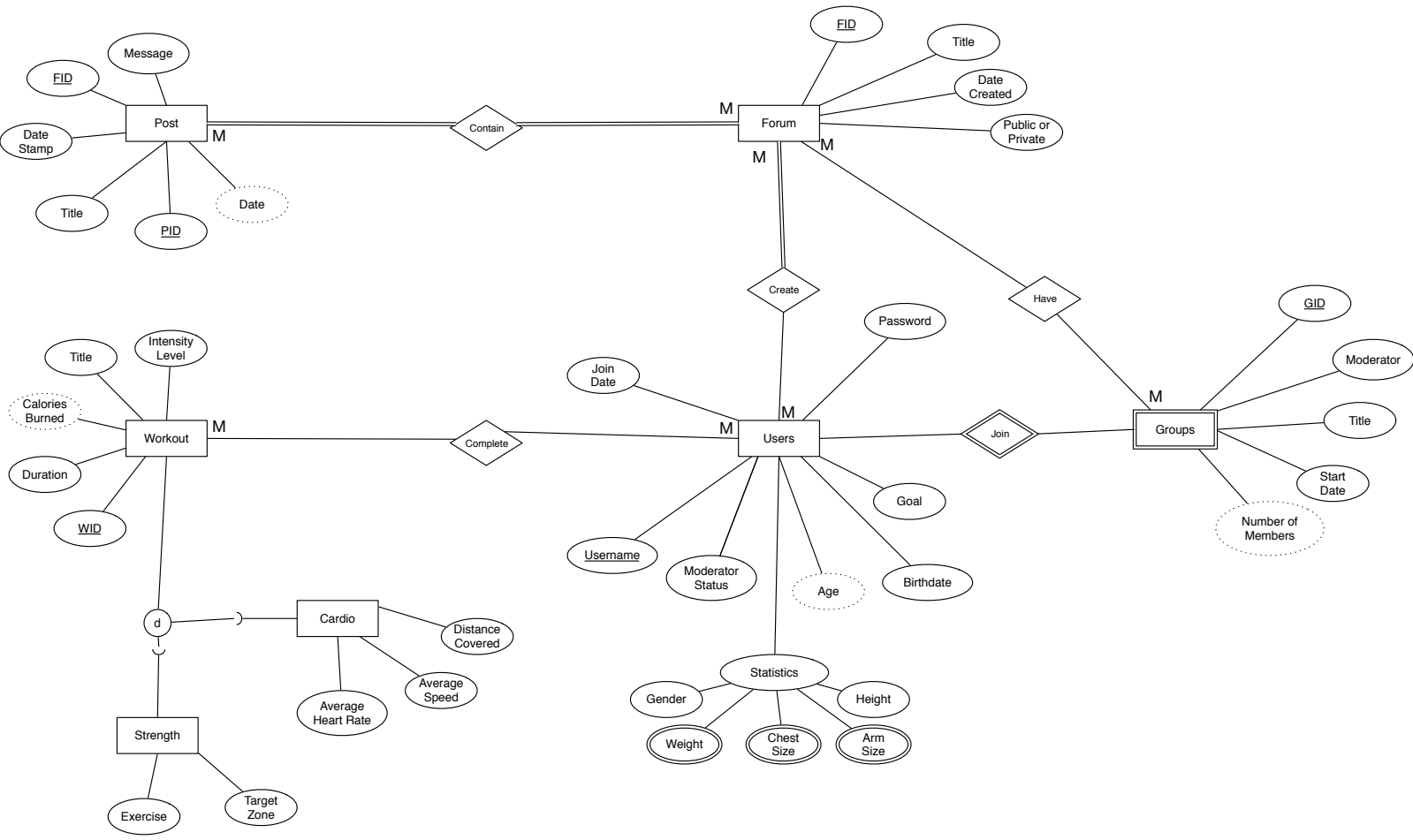
1 Problem Statement

2015 marked a record year for Americans that exercised regularly, hitting just about 55%. However, keeping that momentum is difficult. As blue-collar jobs continue to decline, it has been more important than ever to keep a weekly regiment of eating healthy and exercising.

Living in the most inter-connected generation poses quite a viable idea for social networking: healthy living. With our knowledge of a database systems, we can build a social network that connects users to friends, peers and family to make an online community of healthy living.

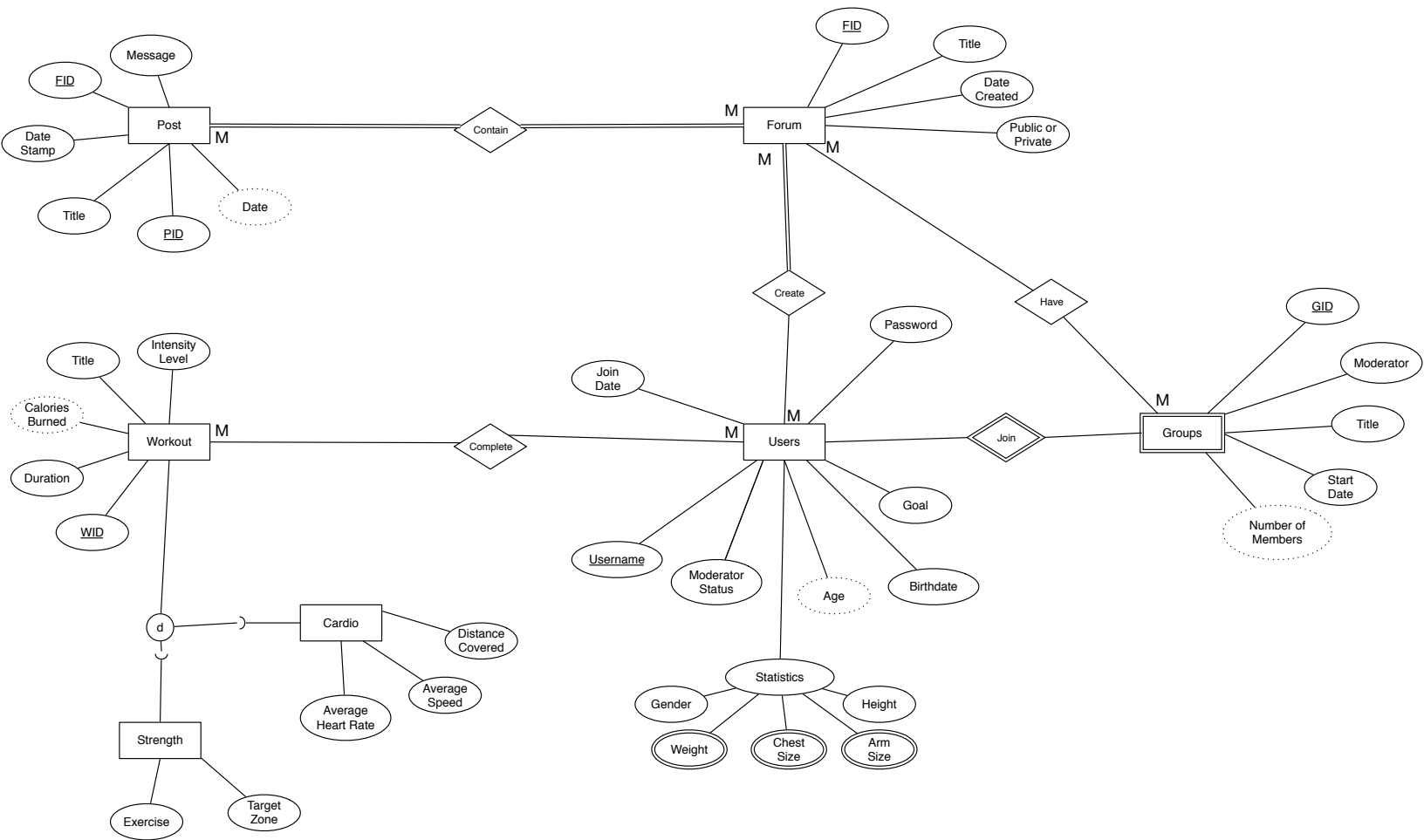
Our database will be essential because it will unite something so mundane and uninteresting with people that are familiar, quite possibly making exercise fun and enjoyable. Maybe we can shape an entire generation by just having motivation a click away.

2 Conceptual Database Design



3 Functional Requirement

- **Users** communicate with **users** by posting **posts**, using **forums**, etc.
- **Groups** contain **users** who communicate with each other.
- **Forums** contain posts.
- **Exercises** get queried by users.
- **Users** get queried by other **users**.



Logical and Physical Database Design

Claire Trebing, Jason Young, Illya Starikov

Due Date: March 18, 2016

1 Revised Problem Statement

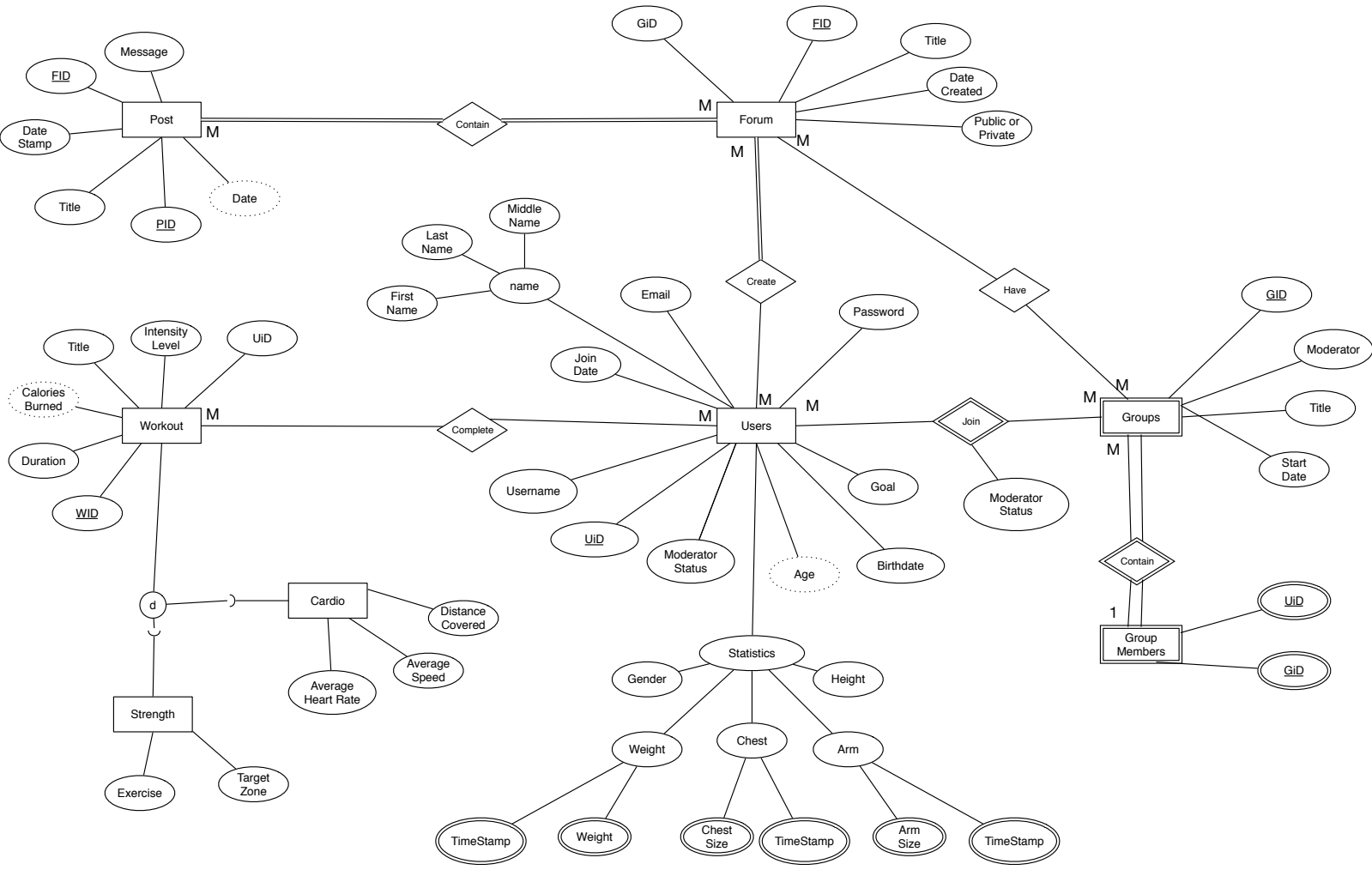
2015 marked a record year for Americans regularly exercising, hitting just over 55%. Keeping that momentum is difficult. As blue-collar jobs continue to decline, it has been more important than ever to keep a weekly regimen of healthy eating and exercising.

Living in the most interconnected generation poses quite a viable idea for social networking: healthy living. We can build a social network that connects users to friends, peers and family to make an online community of healthy living.

Our database will be essential because it will unite something so mundane and uninteresting with familiar faces. This will make exercise more enjoyable and offer a group to hold you accountable. We can shape an entire generation by having motivation a click away.

This database will consist of premade workouts, groups, and reminder emails to help you stay on top of your fitness plan.

2 Revised Conceptual Database Design



3 Logical Database Design

3.1 Relational Set

Please note primary keys are signified by **PK** and foreign keys are signified by *FK*.

Post

<u>PiD</u> PK	<u>FiD</u> <i>FK</i>	Message	DateStamp	Title	Date

Forum

<u>FiD</u> PK	Title	DateCreated	PublicOrPrivate	GiD <i>FK</i>

Groups

<u>GiD</u> PK	Moderator <i>FK</i>	Title	StartDate

Workouts

<u>WiD</u> PK	Duration	Title	IntensityLevel	CaloriesBurned	UiD <i>FK</i>

Strength

<u>WiD</u> PK	Duration	Title	IntensityLevel	CaloriesBurned	UiD <i>FK</i>	Exercise	Target Zone

Cardio

<u>WiD</u> PK	Duration	Title	IntensityLevel	CaloriesBurned	UiD <i>FK</i>	AverageHeartRate
AverageSpeed	DistanceCovered					

Users

<u>UiD</u> PK	Username	Height	Birthdate	Goal	Password	JoinDate	Gender
FirstName	MiddleName	LastName					

Weight

Arm Size

<u>UiD</u> <i>PK, FK</i>	<u>TimeStamp</u> PK	Weight	<u>UiD</u> <i>PK, FK</i>	<u>TimeStamp</u> PK	Arm Size

Chest Size

<u>UiD</u> <i>PK, FK</i>	<u>TimeStamp</u> PK	ChestSize

Group Members

<u>UiD</u> PK	<u>GiD</u> <i>FK</i>	ModeratorStatus

3.2 Summary Table

Attribute	Data Type	Constraints	Meaning
Message	varchar	500 characters	Contents of a post.
FID	int	unique	Forum ID.
DateStamp	timestamp	timestamp of creation	Timestamp of post was creation.
Title	char	35 characters	Title of a post.
PID	int	unique	Post ID.
Date	timestamp	date of creation	Timestamp of post creation.
Title	char	35 characters	Title of a forum.
DateCreated	timestamp	timestamp of creation	Timestamp of forum creation.
PublicOrPrivate	boolean	-	Is forum public.
GID	int	unique	Group ID.
Moderator	char	must match a username	Group owner/moderator.
Title	char	35 characters	Title of the group.
StartDate	timestamp	timestamp of creation	Timestamp of group creation.
IntensityLevel	int	-	Enumerated value, code for different options.
Title	char	35 characters	Title of the workout.
CaloriesBurned	int	num > 0	How many calories burned during exercise.
Duration	timestamp	num > 0	Length of exercise.
WID	int	unique	Workout ID.
Exercise	char	35 characters	Exercise name.
TargetZone	char	35 characters	Area exercise is intended to workout.
AverageHeartRate	int	num > 0	Average heart rate recorded during exercise.
AverageSpeed	int	num > 0	Average speed of cardio exercise.
DistanceCovered	int	num > 0	Distance covered during cardio exercise.
Password	char	-	Password of user.
JoinDate	timestamp	timestamp of creation	Timestamp of users sign up.
Username	char	unique	Users username, used for sign in.
ModeratorStatus	boolean	-	Is a moderator.
Birthdate	date	-	Date of users birth.
Goal	varchar	500 characters	Users intended workout goals.
Gender	char	1 character, M or F	Users gender, male(M) or female(F).
Weight	int	num > 0	Users weight at a certain date.
ChestSize	int	num > 0	Users chest size at a certain date.
ArmSize	int	num > 0	Users arm size at a certain date.
Height	int	num > 0	Users height at a certain date.
UID	int	unique	Users ID.
TimeStamp	date	unique	Date of users weight measurement.
TimeStamp	date	unique	Date of users chest measurement.
TimeStamp	date	unique	Date of users arm measurement

4 Application Program Design

Create a New User

This function creates a new user, accessing only the **user** table.

INPUT	Username, Height, Birthdate, Goal, Password, Gender.
STEPS	1. Check to see if the username is available. If available, proceed. If not, display appropriate message to notify the user. 2. Insert appropriate information to the User table. (Username to username, height to height) 3. Generate a user ID, assign it to the UiD attribute. 4. Take a time stamp, assign it to the JoinDate attribute. 5. Calculate the age based on the BirthDate attribute.
OUTPUT	A new User entity will be inserted into the table, with appropriate data into proper columns (along with computed properties and derived properties).
ASSUMPTIONS	Username, Height, Birthdate, Goal, Password, Gender are all correct (this will be validated in the sign up form).

Delete A Group

This function deletes a group by updating information in **Forum**, and removing the **Group** and **Group Members** tables.

INPUT	The Group ID (GiD) that is to be deleted.
STEPS	1. Check to see if the request is made by the moderator via the Moderator column in the Groups table. If true, approve the request. If not, cancel the request and notify the user. 2. Remove the row that has a matching GiD that was provided for deletion in the Groups table. 3. Query the Group Members table, removing any row that match the GiD provided for deletion. 4. Query the Forum table for any matching Group Ids (GiD), setting the GiD to null if matching.
OUTPUT	The groups are deleted, the group members within that group are deleted, and any reference to the group is deallocated.
ASSUMPTIONS	None.

Modifying User Statistics

Our user statistics have the ability to fluctuate. We would like to accommodate for this fluctuation by allowing users to update their respective statics; specifically, we would like to let users update **Username**, **Height**, **Goal**, **Password**, **Gender** in the **Users** table.

INPUT	The specific attribute(s) of the set Username , Height , Goal , Password , Gender that would like to be updated with the new value.
STEPS	1. Ensure the data is valid (e.g. is not null when applicable, in the proper domain). If it is valid, continue. If not, prompt the user with an error message and try again. 2. Modify the attribute to reflect the new value. 3. Repeat for any additional attributes provided.
OUTPUT	The attribute(s) should now reflect the new value provided.
ASSUMPTIONS	The data is within a proper range (will not overflow).

Query Other Users

This function allows for users to query other users; this can be done via **Username** or **FirstName**, **MiddleName**, and **LastName** from the **Users** table.

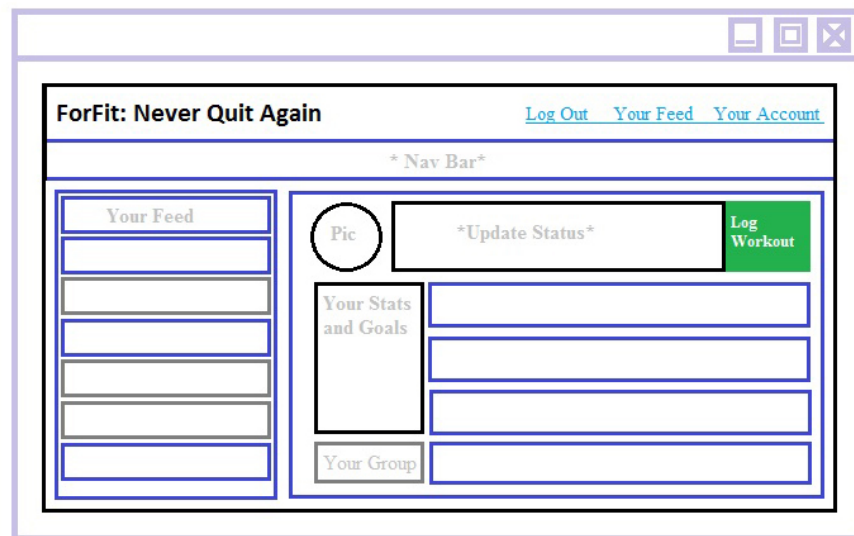
INPUT	Either a Username xor any subset of FirstName , MiddleName , or LastName .
STEPS	1. Check to see if input is valid. If is, proceed. If not, display error message to the user. 2. Query the Users table to see if the user exists. If the user exists, proceed. If not, display appropriate message to the user. 3. Project the profile.
OUTPUT	Either the search user will be projected or an error message is the user does not exist.
ASSUMPTIONS	The first, middle and last name are all provided. The names are unique (solely for the testing purposes).

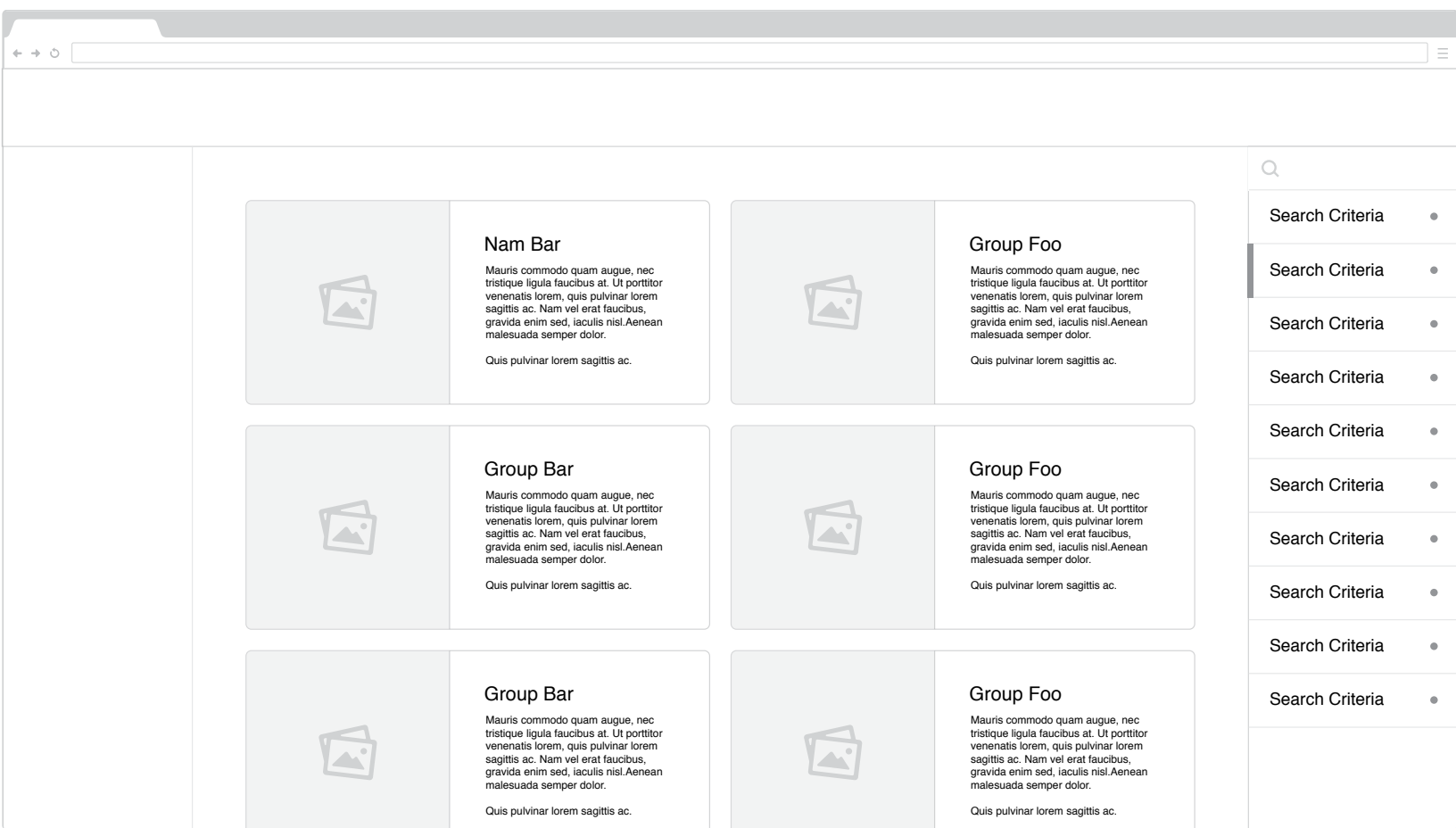
User Leaderboards

Generate the leaderboard based on the workouts accomplished; specifically aggregating data from the **Strength** and **Cardio** table. *Note this is the function that requires multiple tables.*

INPUT	None.
STEPS	1. Merge the User and Workouts table, call the new table Merged. 2. Add up the total duration (call the new property TotalDuration) in the Merged table based on the UiD attribute, making a new table named Sums. 3. Sort the Sums by the TotalDuration attribute. 4. Display the top 10 on the sorted Sums table to the user. 5. Display the user their current rank.
OUTPUT	An eleven-row table displaying the top 10 leaderboards and the user's current rank.
ASSUMPTIONS	There is a bare minimum of eleven users.

5 User Interface Design







 Username 3h

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean urna felis, iacul nec ligula sed, porta lacinia dui. Aenean urna felis, iacul nec ligula sed, porta lacinia dui.

Reply 

 Username 3h

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean urna felis, iacul nec ligula sed, porta lacinia dui. Aenean urna felis, iacul nec ligula sed, porta lacinia dui.

Reply 

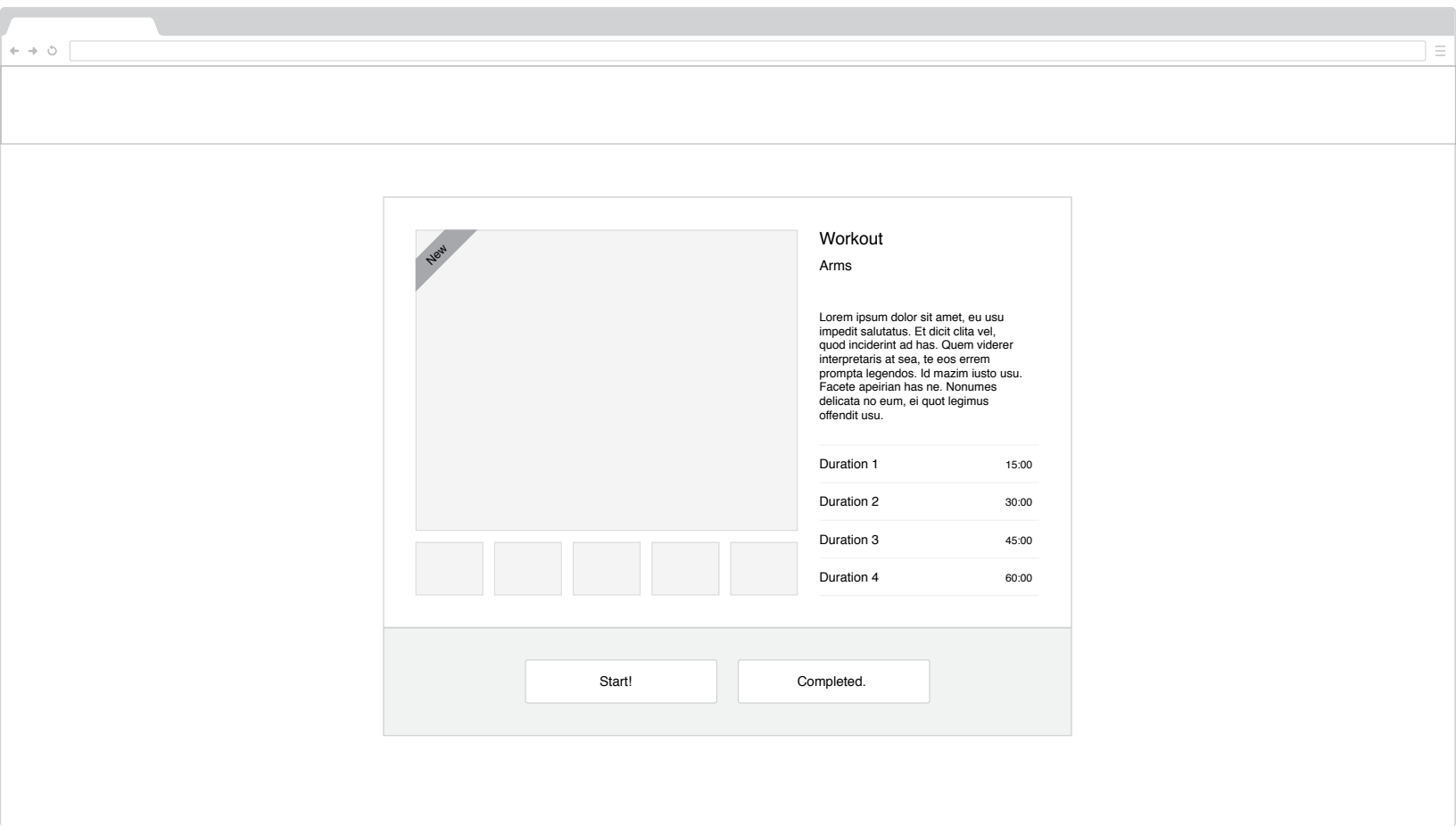
 Username 3h

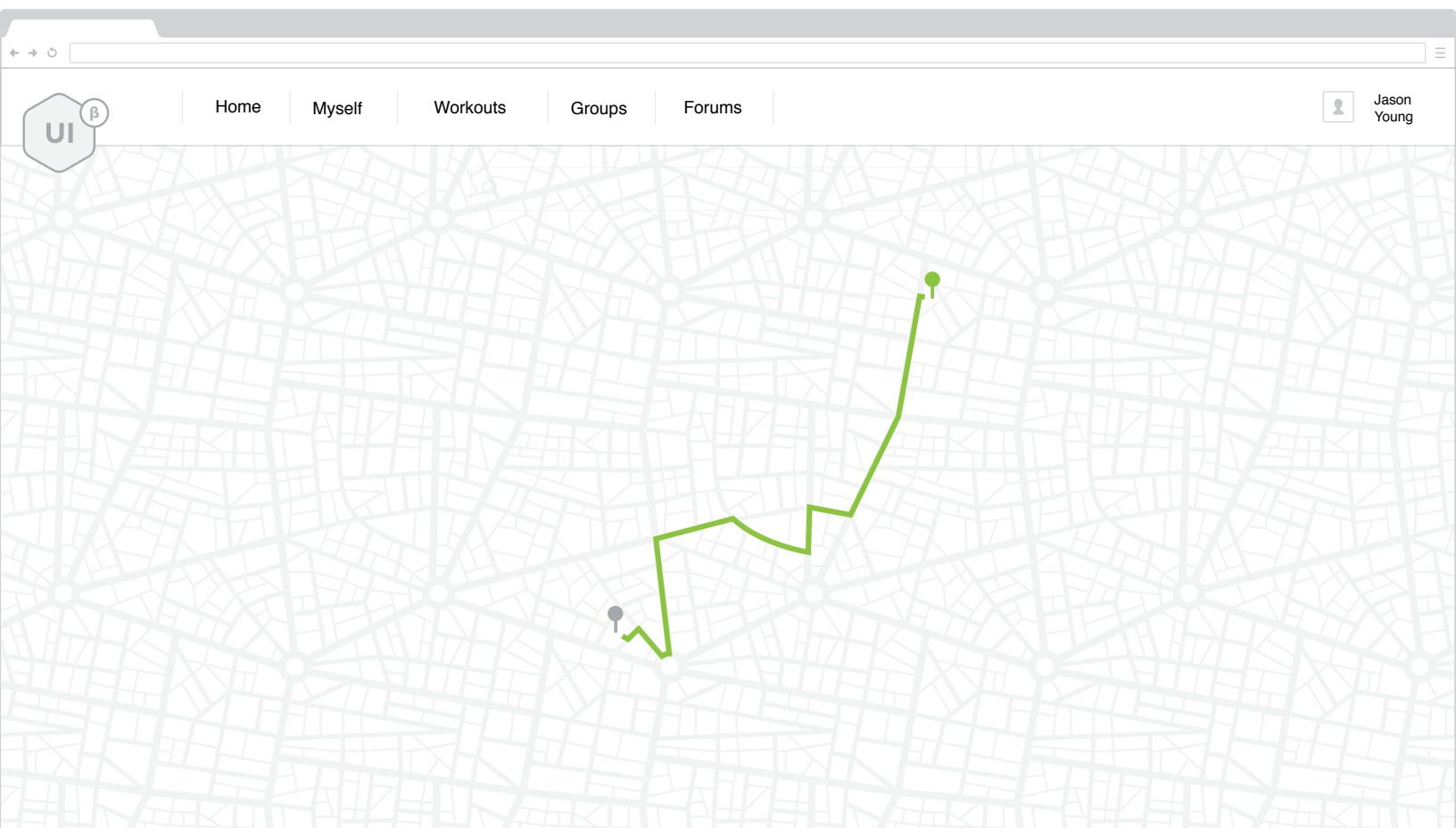
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean urna felis, iacul nec ligula sed, porta lacinia dui. Aenean urna felis, iacul nec ligula sed, porta lacinia dui.

Reply 



	Username Full Name	Followers 3,166	Following 371	Workouts 195
	Username Cras sed feugiat neque. Praesent rhoncus ac dolor eget porttitor. Sed sapien tortor, dictum eget lobortis ac, mattis id tortor.			5h
	Username Cras sed feugiat neque. Praesent rhoncus ac dolor eget porttitor. Sed sapien tortor, dictum eget lobortis ac, mattis id tortor.			5h
	Username Cras sed feugiat neque. Praesent rhoncus ac dolor eget porttitor. Sed sapien tortor, dictum eget lobortis ac, mattis id tortor.			5h





ForFit

Never Quit Again

Illya Starikov, Jason Young, Claire Trebing

July 27, 2025

Contents

I. The Database Design Manual	3
1. Problem Statement	4
2. Conceptual Database Design	5
3. Logical Database Design	7
3.1. Relational Set	7
3.2. Summary Table	8
4. Application Program Design	10
4.1. Create a New User	10
4.2. Delete A Group	10
4.3. Modifying User Statistics	11
4.4. Query Other Users	12
4.5. User Leaderboards	13
II. User Manual	14
5. A Brief Introduction To ForFit	15
6. How It Works	16
7. Test Procedure	17
8. Functions	18
8.1. Delete Group	18
8.2. Modify User	18
8.3. New User	19
8.4. Query User	19
8.5. Total Users	20
8.6. User Leaderboards	20
9. Shell	22
9.1. Create Database	22
9.2. Random Input	22
9.3. Drop Database	23

Part I.

The Database Design Manual

1. Problem Statement

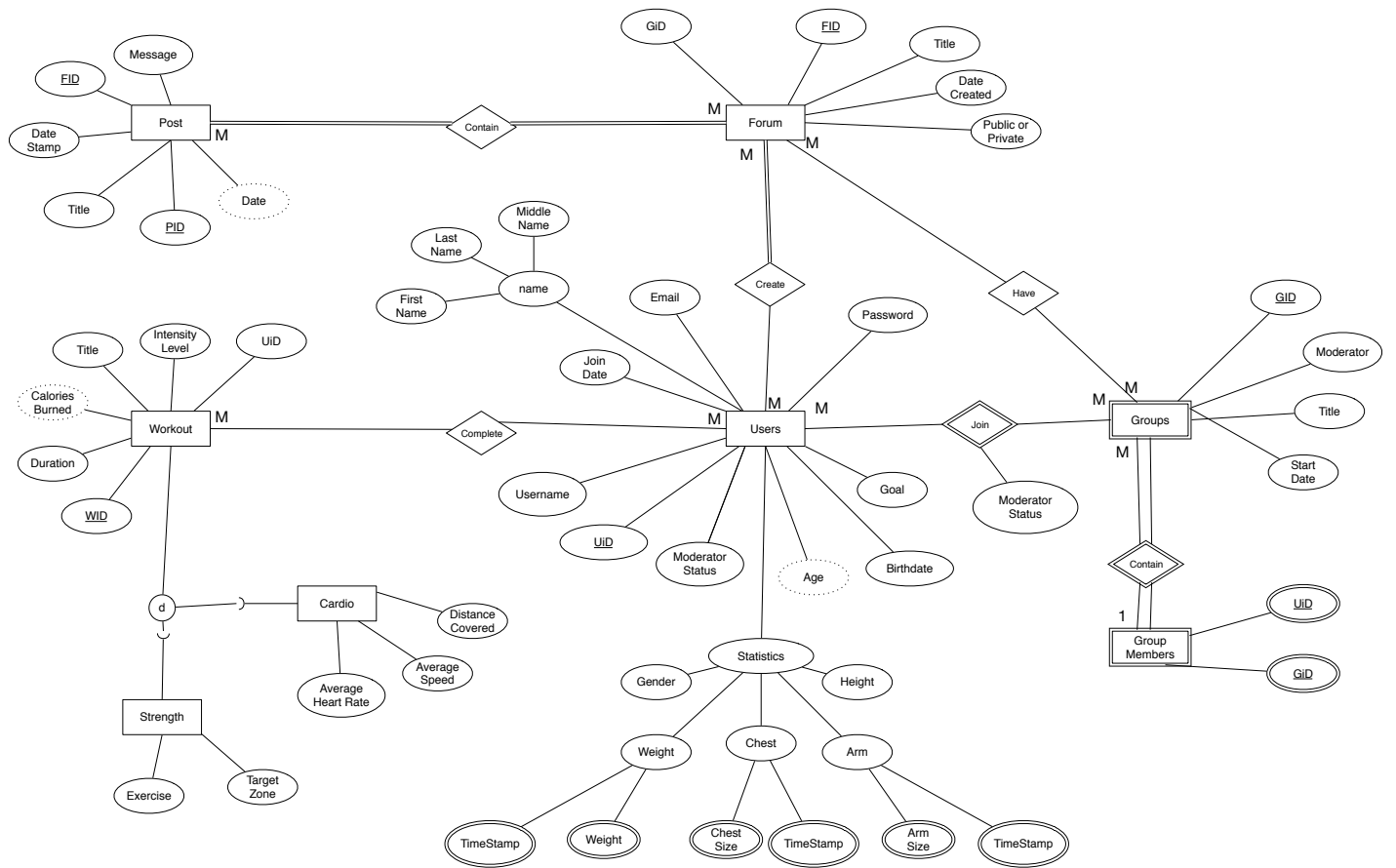
2015 marked a record year for Americans regularly exercising, hitting just over 55%. Keeping that momentum is difficult. As blue-collar jobs continue to decline, it has been more important than ever to keep a weekly regimen of healthy eating and exercising.

Living in the most interconnected generation poses quite a viable idea for social networking: healthy living. We can build a social network that connects users to friends, peers and family to make an online community of healthy living.

Our database will be essential because it will unite something so mundane and uninteresting with familiar faces. This will make exercise more enjoyable and offer a group to hold you accountable. We can shape an entire generation by having motivation a click away.

This database will consist of premade workouts, groups, and reminder emails to help you stay on top of your fitness plan.

2. Conceptual Database Design



3. Logical Database Design

3.1. Relational Set

Please note primary keys are signified by **PK** and foreign keys are signified by *FK*.

Post

<u>PiD</u> PK	<u>FiD</u> <i>FK</i>	Message	DateStamp	Title	Date

Forum

<u>FiD</u> PK	Title	DateCreated	PublicOrPrivate	GiD <i>FK</i>

Groups

<u>GiD</u> PK	Moderator <i>FK</i>	Title	StartDate

Workouts

<u>WiD</u> PK	Duration	Title	IntensityLevel	CaloriesBurned	UiD <i>FK</i>

Strength

<u>WiD</u> PK	Duration	Title	IntensityLevel	CaloriesBurned	UiD <i>FK</i>	Target Zone

Cardio

<u>WiD</u> PK	Duration	Title	IntensityLevel	CaloriesBurned	UiD <i>FK</i>	AverageHeartRate
AverageSpeed	DistanceCovered					

Users

<u>UiD</u> PK	Username	Height	Birthdate	Goal	Password	JoinDate	Gender
FirstName	MiddleName	LastName					

Weight

<u>UiD</u> <i>PK, FK</i>	<u>TimeStamp</u> PK	Weight

Arm Size

<u>UiD</u> <i>PK, FK</i>	<u>TimeStamp</u> PK	Arm Size

Chest Size

<u>UiD</u> <i>PK, FK</i>	<u>TimeStamp</u> PK	ChestSize

3. Logical Database Design

Group Members

<u>UiD</u> ^{PK}	<u>GiD</u> ^{FK}	ModeratorStatus

3.2. Summary Table

3. Logical Database Design

Attribute	Data Type	Constraints	Meaning
Message	varchar	500 characters	Contents of a post.
FID	int	unique	Forum ID.
DateStamp	timestamp	timestamp of creation	Timestamp of post was creation.
Title	char	35 characters	Title of a post.
PID	int	unique	Post ID.
Date	timestamp	date of creation	Timestamp of post creation.
Title	char	35 characters	Title of a forum.
DateCreated	timestamp	timestamp of creation	Timestamp of forum creation.
PublicOrPrivate	boolean	-	Is forum public.
GID	int	unique	Group ID.
Moderator	char	must match a username	Group owner/moderator.
Title	char	35 characters	Title of the group.
StartDate	timestamp	timestamp of creation	Timestamp of group creation.
IntensityLevel	int	-	Enumerated value, code for different options.
Title	char	35 characters	Title of the workout.
CaloriesBurned	int	num > 0	How many calories burned during exercise.
Duration	timestamp	num > 0	Length of exercise.
WID	int	unique	Workout ID.
Exercise	char	35 characters	Exercise name.
TargetZone	char	35 characters	Area exercise is intended to workout.
AverageHeartRate	int	num > 0	Average heart rate recorded during exercise.
AverageSpeed	int	num > 0	Average speed of cardio exercise.
DistanceCovered	int	num > 0	Distance covered during cardio exercise.
Password	char	-	Password of user.
JoinDate	timestamp	timestamp of creation	Timestamp of users sign up.
Username	char	unique	Users username, used for sign in.
ModeratorStatus	boolean	-	Is a moderator.
Birthdate	date	-	Date of users birth.
Goal	varchar	500 characters	Users intended workout goals.
Gender	char	1 character, M or F	Users gender, male(M) or female(F).
Weight	int	num > 0	Users weight at a certain date.
ChestSize	int	num > 0	Users chest size at a certain date.
ArmSize	int	num > 0	Users arm size at a certain date.
Height	int	num > 0	Users height at a certain date.
UID	int	unique	Users ID.
TimeStamp	date	unique	Date of users weight mesurement.
TimeStamp	date	unique	Date of users chest measurement.
TimeStamp	date	unique	Date of users arm measurement

4. Application Program Design

4.1. Create a New User

This function creates a new user, accessing only the `user` table.

INPUT	Username, Height, Birthdate, Goal, Password, Gender.
STEPS	1. Check to see if the <code>username</code> is available. If available, proceed. If not, display appropriate message to notify the user. 2. Insert appropriate information to the User table. (Username to <code>username</code>, height to <code>height</code>) 3. Generate a user ID, assign it to the <code>UiD</code> attribute. 4. Take a time stamp, assign it to the <code>JoinDate</code> attribute. 5. Calculate the age based on the <code>BirthDate</code> attribute.
OUTPUT	A new <code>User</code> entity will be inserted into the table, with appropriate data into proper columns (along with computed properties and derived properties).
ASSUMPTIONS	<code>Username</code> , <code>Height</code> , <code>Birthdate</code> , <code>Goal</code> , <code>Password</code> , <code>Gender</code> are all correct (this will be validated in the sign up form).

4.2. Delete A Group

This function deletes a group by updating information in `Forum`, and removing the `Group` and `Group Members` tables.

4. Application Program Design

INPUT	The Group ID (GiD) that is to be deleted.
STEPS	1. Check to see if the request is made by the moderator via the Moderator column in the Groups table. If true, approve the request. If not, cancel the request and notify the user. 2. Remove the row that has a matching GiD that was provided for deletion in the Groups table. 3. Query the Group Members table, removing any row that match the GiD provided for deletion. 4. Query the Forum table for any matching Group Ids (GiD), setting the GiD to null if matching.
OUTPUT	The groups are deleted, the group members within that group are deleted, and any reference to the group is deallocated.
ASSUMPTIONS	None.

4.3. Modifying User Statistics

Our user statistics have the ability to fluctuate. We would like to accommodate for this fluctuation by allowing users to update their respective statics; specifically, we would like to let users update **Username**, **Height**, **Goal**, **Password**, **Gender** in the **Users** table.

4. Application Program Design

INPUT	The specific attribute(s) of the set Username , Height , Goal , Password , Gender that would like to be updated with the new value.
STEPS	1. Ensure the data is valid (e.g. is not null when applicable, in the proper domain). If it is valid, continue. If not, prompt the user with an error message and try again. 2. Modify the attribute to reflect the new value. 3. Repeat for any additional attributes provided.
OUTPUT	The attribute(s) should now reflect the new value provided.
ASSUMPTIONS	The data is within a proper range (will not overflow).

4.4. Query Other Users

This function allows for users to query other users; this can be done via **Username** or **FirstName**, **MiddleName**, and **LastName** from the **Users** table.

INPUT	Either a Username xor any subset of FirstName , MiddleName , or LastName .
STEPS	1. Check to see if input is valid. If is, proceed. If not, display error message to the user. 2. Query the Users table to see if the user exists. If the user exists, proceed. If not, display appropriate message to the user. 3. Project the profile.
OUTPUT	Either the search user will be projected or an error message is the user does not exist.
ASSUMPTIONS	The first, middle and last name are all provided. The names are unique (solely for the testing purposes).

4.5. User Leaderboards

Generate the leaderboard based on the workouts accomplished; specifically aggregating data from the **Strength** and **Cardio** table. *Note this is the function that requires multiple tables.*

INPUT	None.
STEPS	1. Merge the User and Workouts table, call the new table Merged. 2. Add up the total duration (call the new property TotalDuration) in the Merged table based on the UiD attribute, making a new table named Sums. 3. Sort the Sums by the TotalDuration attribute. 4. Display the top 10 on the sorted Sums table to the user. 5. Display the user their current rank.
OUTPUT	An eleven-row table displaying the top 10 leaderboards and the user's current rank.
ASSUMPTIONS	There is a bare minimum of eleven users.

Part II.

User Manual

5. A Brief Introduction To ForFit

ForFit is a new way to workout. Over the past year, just about 55% of Americans exercised regularly. ForFit is a way to connect you to your friend, family, and other exercise enthusiasts in your area. These exercises can range from traditional bike riding and swimming to newer forms of exercise such as parkour and yoga. ForFit encourages group activity and connection so that workouts are never dull and so that users will be continually motivated and pushed toward their workout goals.

6. How It Works

ForFit has two main components. The first part is the users themselves. Users sign up for the service and then begin recording their workouts. Workouts can be logged as many times as liked, from once a month to several times a day. The more workouts logged, the better data and the more progress toward your exercise goals.

Users can view other users and the workouts they've done. This leads into the second component of ForFit, which is the encouragement of community. Users can create groups and join existing groups. These groups can help users focus on specific areas. For example, a user wanting to run a marathon might join a group of other runners training for a marathon. This way the user can keep in touch and stay motivated through the group and also have a resource to ask questions or get other workouts.

Groups allow users to make forums, for in-depth discussion, and posts, for quick updates. These groups are the main area of community creation that ForFit focuses on in the design. Users can find other users and join groups with users they have been exercising with before. This creates a network of community interaction.

7. Test Procedure

This database was well tested before it reached you, the user. This section will explain how the test database was created and what tests the database went through before completion. The mock data was created using C++ code to randomly generate data. Data from online was used to create First Names, Last Names, Usernames, and Group Names as well as to populate the workout subclasses in strength and cardio. Using C++ code to randomly select and combine information, creating unique data. This mock data was then put through a series of tests to make sure the functions would work properly. Each function was tested over 500 times to error or improper analysis of the data. After confirming that these functions were working properly, they were implemented into the database.

8. Functions

8.1. Delete Group

Delete Group allows you to remove a group from the database. Once a group has been deleted, you can no longer make forums or posts in this group.

Parameters

groupId The GiD of the group that is to be deleted.

Results

The group who's ID matches the given **groupId** will be deleted. If the group does not exist, nothing is displayed.

```
[rMBP:~ postgres$ '/Applications/Postgres.app/Contents/Versions/9.5/bin'/psql -p5432
psql (9.5.2)
Type "help" for help.

postgres=# select delete_group(42);
 delete_group
-----
(0 rows)

postgres=#
```

8.2. Modify User

Modify User allows users to change certain values in the user's data by providing the user name. Users can change their username, height, goal, password, and gender.

Parameters

userID The UiD of the user to who's statistics that will be modified. If this is not provided, there is no way of changing the user.

newUsername The new username property of the user. If this statistic will not change, pass a '0' to the function. Otherwise, proceed with changing.

newHeightFeet See above.

newHeightInches See above.

8. Functions

newGoal See above.

newPassword See above.

newGender See above.

Results

For all attributes that do not have the value of '0', the value of said attribute will be updated (provided it upholds all constraint).

```
(rMBP:~ postgres$ ~/Applications/Postgres.app/Contents/Versions/9.5/bin/psql -p5432
psql (9.5.2)
Type "help" for help.

postgres=# select modify_user(1, 'IllyaStarikov', 0, 0, 42, '0', 'F');
modify_user
(0 rows)

postgres=# select * from USERS where Uid = 1;
 uid | username | email | heightfeet | heightinches | birthdate | goal | password | joindate | gender | fname | minit | lname
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
  1  | IllyaStarikov | CasieScarlett1@gmail.com | 6 | 2 | 1967-03-31 | 42 | EtZEdInfnIBAI | 1999-12-31 | F | Casie | M | Scarlett
(1 row)
```

8.3. New User

New User creates new users in the database. The user must input a username, email, height, birthdate, goal, and password. If any of those are not provided, an error message will be generated.

Parameters

id, username, email, heightFeet, heightInches, birthdate, goal, password All the attributes that can be inserted into user table.

joinDate, gender, firstName, mInitial, lastName See above.

Results

Provided all the attributes meet database constraints, a new user will be inserted into the database. If a constraint is not meant, there will be a message outputting the error and the insert will be rejected.

```
postgres=# select new_user(404, 'IllyaStarikov', 'IllyaStarikov@iCloud.com', 5, 10, '07-20-1994', 165, '42', '07-21-1994', 'M', 'Illya', 'A', 'Starikov');
new_user
(0 rows)

postgres=# select * from USERS where Uid = 404;
 uid | username | email | heightfeet | heightinches | birthdate | goal | password | joindate | gender | fname | minit | lname
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
 404 | IllyaStarikov | IllyaStarikov@iCloud.com | 5 | 10 | 1994-07-20 | 165 | 42 | 1994-07-21 | M | Illya | A | Starikov
(1 row)
```

8.4. Query User

Query User allows users to find others by inputting a first and last name. If a user does not exist than the results are empty.

8. Functions

Parameters

firstName, lastName The first and last name of the user to be queried.

Results

If the user exists, the column of the user will be returned. If the user does not exist, not such column will be returned.

```
rMBP:~ postgres$ '/Applications/Postgres.app/Contents/Versions/9.5/bin'/psql -p5432
psql (9.5.2)
Type "help" for help.

postgres=# select query_user('Illya', 'Starikov');
               query_user
-----
(117,IllyaStarikov,IllyaStarikov@iCloud.com,5,10,1994-07-20,165,42,1994-07-21,M,Illya,A,Starikov)
(1 row)
```

8.5. Total Users

Total Users returns how many users are in the whole database. In the future, this function will give a user count for an individual group.

Parameters

None.

Results

Returns the total count of users, a maintenance function used to compare to the total count in the database.

```
rMBP:~ postgres$ '/Applications/Postgres.app/Contents/Versions/9.5/bin'/psql -p5432
psql (9.5.2)
Type "help" for help.

postgres=# select total_users();
      total_users
-----
              52
(1 row)
```

8.6. User Leaderboards

User Leaderboards gives you a top 10 user count. This is calculated by a count of workouts done by a user. This count includes all the users in a database and is not specific to a group.

8. Functions

Parameters

None.

Results

Returns the top 10 users (calculated via count of workouts done) in the database.

```
lrmBP:~ postgres$ '/Applications/Postgres.app/Contents/Versions/9.5/bin'/psql -p5432
psql (9.5.2)
Type "help" for help.

postgres=# select user_leaderboards();
 user_leaderboards
-----
 (Gregory,Patty,6)
 (Agueda,Heather,4)
 (Bertram,Gerardoy,4)
 (Gay,Arthurury,4)
 (Garry,Shanony,4)
 (Brad,Lana,4)
 (Francisco,Bruce,4)
 (Edgar,Dennisy,4)
 (Alvina,Carter,4)
 (Darlene,Salvatore,3)
(10 rows)
```

9. Shell

We know doing tasks such as creating the database, generating a new set of random inputs, or even running all the function files to make sure all of the functions are created can be tedious. Fortunately, we have a solution — that solution is [Shell scripting](#).

We quickly realized that repetitively running the same commands¹ could get tedious, so we have provided multiple shell scripts that automate most of the monotonous tasks.

To run a shell script, first you must give it privileges to be able to run **Bash** commands. This is accomplished by `chmod -x /path/to/file.sh`. This gives read/write access to the shell script, which then allows the shell script to be run via `./path/to/file.sh`. This then allows the user to do tasks like setting up the entirety of the database (which spans multiple files!) one file path away.

Below are the shell scripts we provide.

9.1. Create Database

A lot of testing requires the use of setting up the database from scratch. Unfortunately, this usually means creating the database scheme, followed by inserting the data, followed by inserting the functions, and so forth. Although an easy fix would be to have all the code in a single file, this would combine concerns. We would rather have a lot of function specific files apposed to one giant file. Our solution: `./create_database.sh`

The create database script works like so:

1. Create the database schema by `create`ing all the tables and `modify`ing all for foreign key constraints.
2. Insert data, which is artificially generated (as will be mentioned below).
3. Go through the *entirety* of the functions directory, and running each function `sql` script to ensure all functions are inserted into the database.

This allows for easily bringing a database back up after it has been reset.

9.2. Random Input

Let face it, every social network occasionally needs some help. That is why we have a shell script that executes our C++ codebase to auto-populate the table. The shell script runs like so:

¹Like running ``/Applications/Postgres.app/Contents/Versions/9.5/bin'/psql -p5432 -f` to create every file!

9. *Shell*

1. Change to the directory of the C++ codebase.
2. Run the makefile
3. Run the executable, pipe that to an output file.
4. Move the shell script to the directory of all the other `sql` files.

This makes for easily produced, new data.

9.3. Drop Database

Drop database simply runs the `sql` script to drop all tables in a cascade fashion. It's a simple script but is much quicker than manually running the command to drop the database.

10. Website

To setup the website you only need to place the website files inside of an apache server and then rewrite the config.php file.

For the config.php file you need to modify the variables;

```
"  
    date_default_timezone_set('America/Chicago');  
    define('DB_SERVER', 'localhost:3306');  
    define('DB_USERNAME', 'JasonY');  
    define('DB_PASSWORD', 'Servant83');  
    define('DB_DATABASE', 'forfit');  
"
```

to fit the website's database.

The website is a blend of PHP and HTML. It allows you to sign up, logout, view groups and forums.

The main page contains the link to signup. After signing up or logging in you can open the menu and view the groups which contain forums. You may also enter your personal information in the profile which tracks your exercises and gains.

End of Semester Material

Illya Starikov

March 24, 2016

Because it is the end of the semester and we have gotten off track with the assignment, I have decided to clarify expectations and rework some things.

1 Changelog

The follow noticeable changes have occurred:

Dropbox Is no longer going to suffice. Everything is moved to `git`.

- Contact me if there is any issue with the transition.

Subsystems Are now going to be split different. They are listed below.

Front-End Development *Jason Young and Illya Starikov*. This will most likely $\frac{2}{3}$ to $\frac{1}{3}$, being heavy on Jason.

Back-End Development *Jason Young*. I am very unfamiliar with Django so it will be up to Jason for the extensive part of the backend development.

Database Development *Claire Trebing and Illya Starikov*. The work load will be mostly be a $\frac{1}{2}$ to $\frac{1}{2}$.

2 Meetings

Meetings have been wishy-washy the last few months, but as we are approaching the end of the semester we are going to need to more frequently meet to discuss how the project is going. **Until the your subsystem is finished, meetings will occur.**

3 Project Phase III

Project Phase III has three parts:

Source Code This will be covered by the Database subsystem.

Report This will be covered by Claire and I.

- The revisions will be made by me.
- The user manual will be made by Claire.

For the user manual, the deliverable is a PDF, Word Document or Google Document. It **must be able to render in plain text** (i.e. I have to be able to paste it into **Sublime Text** without any formatting issues). Please account for this. The document will be gutted and reformatted to L^AT_EX.

The manual must be broken down into sections, subsection, and subsection or subsections (or, subsubsection). Please indicate these in some manner (whether it be verbosely written or through formatting). This is used for a table of contents, to keep in mind.

Formatting such as *italicizing*, **boldfacing** and the like are encouraged. I will take care of make sure they render correctly. Equations of any form (from $\sum \vec{F} = m\vec{a}$ to $\frac{\hbar^2}{2m}\nabla^2\Psi + V(\mathbf{r})\Psi = -i\hbar\frac{\partial\Psi}{\partial t}$) are encouraged but must be easily readable. Any tables, figures, images and the like must be coherent and easy to interpret.

4 Homework #3 & #4

The last two homeworks are SQL-based, so for the last two homeworks:

- Everyone will finish their own homework.
 - Upon completion, email to me.
 - I will run it through a **diff** utility.
 - * Any noticeable disparities will be discussed with respective authors.
- When everyone's homework matches, *Claire* will submit the SQL file.
- Have the homework ready at least two days in advance.

CS2500

Algorithms



1 Algorithms Introduction

- Insertion Sort Example

Input: 503 87 512 61 908 170 897

`/* The brackets signify the sorted array */`

```
[503] 87 512 61 908 170 897
[87 503] 512 61 908 170 897
[87 503 512] 61 908 170 897
[61 87 503 512] 908 170 897
[61 87 503 512 908] 170 897
[61 87 170 503 512 908] 897
[61 87 170 503 512 908] 897
[61 87 170 503 512 897 908]
```

- Do the take home examples.

2 Asymptotic Analysis

- Big-O

$$O(g(n)) = \{f(n) : \text{there exists positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \forall n \geq n_0\}$$

(1)

2 Asymptotic Analysis

2.1 Proofs

Prove the following for big O.

$$n^2 + 2n + \lg n = O(n^2) \quad (1)$$

$$\leq n^2 + 2n + \lg n \quad (2)$$

$$\leq n^2 + 2n^2 + \lg n \cdot n \quad (3)$$

$$\leq n^2 + 2n^2 + n^2 \quad (4)$$

$$= 4n^2 \quad (5)$$

So $c = 4$ and $n_0 = 1$.

2.2 Proof II

Prove the following for big Ω

$$2n^3 - 3n^2 + 2n = \Omega(n^3) \quad (6)$$

$$= 2n^3 - 3n^2 + 2n \quad (7)$$

$$\geq 2n^3 - 3n^2 \quad (8)$$

$$\geq 2n^3 - n^3, \quad n \geq 3 \quad (9)$$

$$= n^3 \quad (10)$$

So $c = 1$, $n_0 = 3$.

2.3 Proof III

Upper bound.

$$\frac{n^2}{2} - \frac{n}{2} = \Theta(n^2) \quad (11)$$

$$= \frac{n^2}{2} - \frac{n}{2} \quad (12)$$

$$\leq \frac{n^2}{2} - \frac{n}{2} \quad (13)$$

$$= \leq \frac{n^2}{2}, c_2 = \frac{1}{2}, n \geq 0. \quad (14)$$

Lower bound.

$$\frac{n^2}{2} - \frac{n}{2} \quad (15)$$

$$\geq \frac{n^2}{2} - \frac{n}{2} * \frac{n}{2} \quad (16)$$

$$= \frac{1}{4}n^2 \quad (17)$$

$$c_1 = \frac{1}{4} \text{ and } n_0 = 2$$

2.4 Growth Classes Proof

Show $n^3 - 10n^2 \neq O(n^2)$

$$0 \leq n^3 - 10n^2 \leq cn^2 \quad (18)$$

$$n^3 \leq 10n^2 + cn^2 \quad (19)$$

$$n^3 \leq (10 + c)n^2 \quad (20)$$

$$n \leq 10 + C \quad (21)$$

This is a contradiction.

2.5 Growth Classes Example II

Show $2^n = o(3^n)$

$$2^n = o(3^n) \quad (22)$$

$$\lim_{n \rightarrow \infty} \frac{2^n}{3^n} = \lim_{n \rightarrow \infty} \frac{2}{3} \quad (23)$$

$$= 0 \quad (24)$$

3 Divide and Conquer

Search for 64.

```
10 20 37 43 52 76 85 93
10 20 37 *43* | 52 76 85 93 [43 vs 64]
52 *76* | 85 93 [76 vs 64]
*52* | 76 [52 vs 64]
```

Three comparisons.

3.1 Merge Sort

```
38 27 43 3 | 9 82 10
38 27 | 43 3      9 82 | 10
38 | 27      43 | 3      9 | 82      10
38      27      43      3      9      82      10
27 38      3 43      9 82      10
3 27 28 43      9 10 82
3 9 10 38 43 82
```

3.2 Substitution Method

$2^n - 1$

Base case: $2 - 1 = 1$

Inductive:

(1)

$$T(n) = 2^n - 1 \quad (2)$$

$$T(n+1) = 2^{n+1} - 1 \quad (3)$$

$$= 2 T(n) + 1 \quad (4)$$

$$= 2(2^n - 1) \quad (5)$$

$$= 2^{n+1} - 1 \quad (6)$$

4 Recurrence

Substitution Example

$$\begin{aligned}T(n) &= cn \\T(n+1) &= c(n+1) \\&= T(n) + c \\&= cn + c \\&= c(n+1)\end{aligned}$$

Recursion Tree

$$2T(n/2) + n^2.$$

Master Theorem

Check given $T(n)$ is the form of $aT(n/b) + f(n)$

$$T(n) = 8T(n/2) + n^2 \tag{1}$$

$a = 8, b = 2, f(n) = n^2$. Prove $f(n) \in O(n^{\log_2 a - \epsilon}) = O(n^{3-\epsilon}) = O(n^{3-1}) = O(n^2)$. $T(n) = \Theta(n^{\log_2 a}) = \Theta(n^3)$

4 Recurrences

$$T(n) = 2T(n/2) + n^3 \quad (1)$$

$$\log_b a = \log_2 2 = 1 \quad (2)$$

$$\log^{b^a} = n \quad (3)$$

$$n^3 \in \Omega(n^{\log_b a + 2}) \quad (4)$$

$$\Omega(n^3) \quad (5)$$

$$af(n/b) \leq cf(n) \quad (6)$$

$$2f(n/2) \leq cf(n) \quad (7)$$

$$2 \times \left(\frac{n}{2}\right)^3 \leq cn^3 \quad (8)$$

$$\frac{n^3}{4} \leq cn^3 \quad (9)$$

$$\text{choose } c < 1 \quad (10)$$

$$\text{choose } c = \frac{1}{4} \quad (11)$$

$$T(n) \in \Theta(n^3) \quad (12)$$

5 Correctness

Loop has three parts, init, maintenance, termination.

6 Quicksort

6 Quick Sort

1. wall, current, pivot

- $\text{Array} \leq \text{pivot} \leq \text{Array}$

```
let wall = |, pivot = []  
| 2 9 5 6 10 0 7 [4]  
2 | 9 5 6 10 0 7  
2 0 | 5 6 10 9 [7]  
0 2 -> 5 -> 6 | 10 9  
5 6 | 10 9
```

Output: 0 2 5 6 7 9 10

7 Heapsort

```
10 9 7 8 4 6 1 3 5 2 -> 10 | 9 7 | 8 4 6 1 | 3 5 1 0 0 0 0 0 |
15 5 11 1 4 10 7 3 9 -> 15 | 5 11 | 1 4 10 7 | 3 9 0 0 0 0 0 0 |
                        9  15 | 9 11 | 5 4 10 7 | 3 1 0 0 0 0 0 0 |
                        1  15 | 5 11 | 1 4 10 7 | 3 9 0 0 0 0 0 0 |
                        3  15 | 5 11 | 1 4 10 7 | 3 9 0 0 0 0 0 0 |
                        =  15 | 5 11 | 1 4 10 7 | 3 9 0 0 0 0 0 0 |
```

The follow the while loop for the heapsort.

8 Counting Sort Algorithm

- We have talked about many algorithms so far
 - Insertion Sort** Worst: $\Theta(n^2)$, Average: $\Theta(n^2)$
 - Merge Sort** Worst: $\Theta(n \lg n)$, Average: $\Theta(n \lg n)$
 - Quick Sort** Worst: $\Theta(n^2)$, Average: $\Theta(n \lg n)$
 - Heap Sort** Worst: $\Theta(n \lg n)$, Average: $\Theta(n \lg n)$
- Most of the basic operations have been so far is comparison.
- Merge sort is asymptotically optimal (in general).
- Any comparison based sorting is bounded $\Omega(n \lg n)$
- Is there any way to sort in linear time?
 - Counting Sort is a way to achieve this.

8.1 Counting Sort

We want to sort integers. $[0 \dots k]$, where k is max in the array.

1. Count the number of each distinct elements (frequency).
2. Determine the position of each element by placing it to the “counting index”.

Specifically,

1. Specifically, given array of size n , and the maximum integer in the array is k .
2. Create “counting array”, whose size is $k + 1$.
3. Fill the counting array with the frequency of each element.
4. Update the counting array accumulatively adding up frequencies
5. Place our numbers in the position of “accumulated numbers”.

8.1.1 Example

Given Array: [1 3 0 2 4 3 1]

Size: 7

Max: 4

Counting array (size of $k+1$): [1 | 2 | 1 | 2 | 1]

Notice how the indexes are how many times the number appears in original array

Update: [1 3 4 6 7]

These are the updated positions of the array elements

Enumerated Steps

Output: [| | 1 | | |]

Now decrement updated array. ([1 2 4 6 7])

Output: [| | 1 | | 3 |] ([1 2 4 5 7])

Output: [| | 1 | | 3 | 4] ([1 2 4 5 6])

So on and so forth.

8.1.2 Psuedocode

Let A = input array, B = output array, n = size of array, k = maximum number.

```
countingSort(A, B, n, k)
    let C[0...k] be the counting array
    for i = 0 to k
        C[i] = 0
    for j = 1 to n
        C[A[j]] = C[A[j]] + 1
    for i = 1 to k
        C[i] = C[i] + C[i - 1]
    for j = n down to 1
        B[C[A[j]]] = A[j]
        C[A[j]] = C[A[j]] - 1
```

We can see that the complexity is $\Theta(n + k)$

9 Examples

9.1 Counting Sort Algorithm

4 6 4 1 3 4 1 4

Assume $[0 \dots k]$. Max integer: 6, size of array: 8.

Original Array: [4 | 6 | 4 | 1 | 3 | 4 | 1 | 4]

Counting Array: [0 | 2 | 0 | 2 | 3 | 0 | 1] (size is max + 1)

Update Array: [0 | 2 | 2 | 4 | 7 | 7 | 8]

Output Array: [1 | 1 | 3 | 4 | 4 | 4 | 6]

(reverse order of original array via indexes, the output is not index based)

9.2 Growth Classes Example

Show $20n^2 + 2n + 5 = O(n^2)$

$$O(n^2) = 20n^2 + 2n + 5 \quad (1)$$

$$= 20n^2 + 2n^2 + 5n^2 \quad (2)$$

$$= 27n^2 \quad (3)$$

So $n_0 = 1$ and $C = 27$.

9.3 Growth Example II

$5n + 7 = o(n^2)$. For every c , find $n_0 > 0$

$$5n + 7 \leq 5n + n = 6n < cn^2 \quad (4)$$

For $n_0 = 7$, this holds. But now we have to prove $\forall c$. We do so via limit.

9.4 Growth Class Example

Prove $5n^2 + 2n = \Theta(n^2)$. Just take big- O and big- Ω

9.5 Growth Class Example

Prove $5n^2 + 15n + 100 = \Omega(n^2)$

$$> 5n^2 - 15n \quad (5)$$

$$\geq 5n^2 - n^2 \quad (6)$$

$$= 4n^2 \quad (7)$$

9.6 Master Theorem

10 Dynamic Programming

$$F(2) = F(0) + F(1) = 1$$

$$F(3) = F(2) + F(1) = 2$$

11 LCS

	1	2	3	4
1	0	36	84	124
2	X	0	72	132
3	X	X	0	120
4	X	X	X	0
So $((A_1A_2)A_3)A_4$				

Test I Study Guide

Illya Starikov

February 15, 2016

1 Sorting Algorithms

1.1 Insertion-Sort

```
1 Insertion-Sort(A)
2   for j = 2 to A.length
3     key = A[j]
4     // Insert A[j] into the sorted sequence A[1..j - 1]
5     i = j - 1
6     while i > 0 and A[i] > key
7       A[i + 1] = A[i]
8       i = i - 1
9     A[i + 1] = key
```

1.2 Merge Sort

```
1 Merge-Sort(A, p, r)
2   if p < r
3     q =  $\lfloor (p + r) / 2 \rfloor$ 
4     Merge-Sort(A, p, q)
5     Merge-Sort(A, q + 1, r)
6     Merge(A, p, q, r)
7
8 Merge(A, p, q, r)
9    $n_1 = q - p + 1$ 
10   $n_2 = r - q$ 
11  let L[1.. $n_1 + 1$ ] and R[1.. $n_2$ ] be new arrays
12  for i = 1 to  $n_1$ 
13    L[i] = A[p + i - 1]
14  for j = 1 to  $n_2$ 
15    R[j] = A[q + j]
16  R[ $n_1 + 1$ ] =  $\infty$ 
17  R[ $n_2 + 1$ ] =  $\infty$ 
18  i = 1
19  j = 1
```

```

20     for k = p to r
21         if L[i] ≤ R[j]
22             A[k] = L[i]
23             i = i + 1
24         else A[k] = R[j]
25             j = j + 1

```

1.3 Heap Sort

```

1  Max-Heapify(A, i)
2      l = Left(i)
3      r = Right(i)
4      if l ≤ A.heap-size and A[l] > A[i]
5          largest = l
6      else largest = i
7      if r ≤ A.heap-size and A[r] > A[largest]
8          largest = r
9      if largest ≠ i
10         exchange A[i] with A[largest]
11         Max-Heapify(A-largest)
12
13 Heapsort(A)
14     Build-Max-Heap(A)
15     for i = A.length downto 2
16         exchange A[1] with A[i]
17         A.heap-size = A.heap-size - 1
18         Max-Heapify(A, 1)

```

1.4 Quicksort

```

1  Quicksort(A, p, r)
2      if p < r
3          q = Partition(A, p, r)
4          Quicksort(A, p, q - 1)
5          Quicksort(A, q + 1, r)
6
7  Partition(A, p, r)
8      x = A[r]
9      i = p - 1
10     for j = p to r - 1
11         if A[j] ≤ x
12             i = i + 1
13             exchange A[i] with A[j]
14     exchange A[i + 1] with A[r]
15     return i + 1

```

1.5 Counting Sort

Let A = input array, B = output array, n = size of array, k = maximum number.

```
1 countingSort(A, B, n, k)
2   let C[0...k] be the counting array
3   for i = 0 to k
4       C[i] = 0
5   for j = 1 to n
6       C[A[j]] = C[A[j]] + 1
7   for i = 1 to k
8       C[i] = C[i] + C[i - 1]
9   for j = n down to 1
10      B[C[A[j]]] = A[j]
11      C[A[j]] = C[A[j]] - 1
```

1.6 Rate of Growth

Insertion Sort Worst: $\Theta(n^2)$, Average: $\Theta(n^2)$

Merge Sort Worst: $\Theta(n \lg n)$, Average: $\Theta(n \lg n)$

Quick Sort Worst: $\Theta(n^2)$, Average: $\Theta(n \lg n)$

Heap Sort Worst: $\Theta(n \lg n)$, Average: $\Theta(n \lg n)$

2 Growth Classes

2.1 O -notation

$O(g(n)) = \{f(n) : \text{there exists positive constant } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq c g(n) \text{ for all } n \geq n_0\}$

2.2 Ω -notation

$\Omega(g(n)) = \{f(n) : \text{there exists positive constants } c \text{ and } n_0 \text{ such that } 0 \leq c g(n) \leq f(n) \text{ for all } n \geq n_0\}$

2.3 Θ -notation

$\Theta(g(n)) = \{f(n) : \text{there exists positive constants } c_1, c_2 \text{ and } n_0 \text{ such that}$
 $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0\}$

2.4 o -notation (Little-o)

$f(n) \in o(g(n))$ iff for every $c > 0$ there is an $n_0 > 0$ such that

$$0 \leq f(n) < c g(n)$$

for all $n \geq n_0$

2.5 ω -notation (Little-omega)

$$f(n) \in \omega(g(n)) \text{ iff } g(n) \in o(f(n)).$$

or $f(n) \in \omega(g(n))$ iff for every $c > 0$ there is an $n_0 > 0$ such that

$$0 \leq c g(n) < f(n)$$

for all $n \geq n_0$

3 Master Theorem

The master theorem can only be applied to recurrence equations of the form:

$$T(n) = aT(n/b) + f(n)$$

3.1 Constants

n The size of the problem

a The number of subproblems

n/b The size of each subproblem

f(n) cost outside of recursive calls (divide, combine)

3.2 Cases

$f(n) \in O(n^{\log_b a - \epsilon})$	$T(n) \in \Theta(n^{\log_b a})$
$f(n) \in \Theta(n^{\log_b a})$	$T(n) \in \Theta(n^{\log_b a} \lg n)$
$f(n) \in \Omega(n^{\log_b a + \epsilon})$	$T(n) \in \Theta(f(n))$

4 Definitions

Algorithm Any well-defined computational procedure that takes a set of values as input and produces a set of values as output in a finite number of steps

Correct Algorithm One returns the correct solution for every valid instance of a problem

Loop Invariance Define a key property about the relationship among variables of the algorithms.

- Holds in the *initial* case.

Initialization The loop invariance must be true prior to the first iteration.

- Is *maintained* each iteration.

Maintenance If the property holds prior to an iteration, it must still hold after the iteration is complete.

- Yields correctness when the loop *terminates*.

Termination The invariant provides a useful property that helps demonstrate the algorithm is correct.

Heapify Go all the way down to the heap and fix the violations of the max-heap property by sifting-up

- Dynamic Programming Two Properties
 - Overlapping Subproblems** A recursive solution contains a small number of distinct subproblems repeated many times
 - Expressed Recursively** An optimal solution to a problem contains optimal solution to subproblems
- Subsequence: A sequence after deleting some elements
- A algorithm always makes the choice that looks best at the moment
 - Used for optimization
 - Make local optimal choices and hope to achieve global optimality (Greedy-choice property)
 - An optimal solution to the problem contains an optimal solution to subproblems (optimal substructure)
- Graph
 - a symbolic representation of a network and of its connectivity
- There are two ways to represent graphs
 - Adjacency list
 - Adjacency matrix
- Cut edge: Edge whose deletion will increase the number of connected components (Disconnect the graph)

1 Complexities

1.1 Space Requirements

Adjacency Lists For directed and undirected $\Theta(V + E)$

Adjacency Matrix For directed and undirected $\Theta(V^2)$

1.2 Search Complexities

Breadth-First $\mathcal{O}(V + E)$

Depth-First $\mathcal{O}(V + E)$

2 Coloring Depth First Search

White undiscovered

Gray discovered but not finished

Black finished (found every reachable vertex from it)

1 Notes

- A shortest may not exist when:
 - Negative weight cycle
 - The graph is not connected
- Two Properties of The Shortest Path
 - The optimal substructure of dynamic shortest path: A subpath of a shortest path itself is a shortest path
 - Triangle inequality
 - * For all $u, v, x \in V$ we have $\delta(u, v) \leq \delta(u, x) + \delta(x, v)$
 - * Shortest path from u to v is at most any particular path, e.g., the blue chain.
- Tractable Problems
 - Problems solvable in a polynomial time
- $P \subset NP$
- NPC: If any other problem (i.e., NP-Hard problems) p in NP is polynomial reducible to problem q

2 Psuedocode

2.1 Dijkstra(G,s)

```
d[s] = 0
for each vertex v in V-{s}
    d[v] = infinity
S = emptySet
Q = V

while Q is not empty
    u = Extract_Min(Q)
    S = S ∪ {u}
    for each vertex v adjacent to u
        if d[v] > d[u] + w(u,v)
            d[v] = d[u] + w(u,v)
```


2.2 Bellman-Ford

Given $G=(V,E,w)$ and source vertex s
 $d[x]$ = distance estimate from s to x

```
Bellman-Ford( $G,s$ )
 $d[s] = 0$ 
for each vertex  $v$  in  $V-\{s\}$ 
     $d[v] = \text{infinity}$ 

for  $i=1$  to  $|V|-1$ 
    for each edge  $(u,v)$  in  $E$ 
        if  $d[v] > d[u] + w(u,v)$ 
             $d[v] = d[u] + w(u, v)$ 

for each edge  $(u, v)$  in  $E$ 
    if  $d[v] > d[u] + w(u, v)$ 
        report that a negative-weight cycle exists
```

2.3 Floyd-Warshall

```
 $D^{\wedge}(0) = W$ 
for  $k = 1$  to  $n$  do
    for  $i = 1$  to  $n$  do
        for  $j = 1$  to  $n$  do
             $d_{ij}(k) = \min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\};$ 

Return  $D^{\wedge}(n)$ 
```

2.4 Ford Fulkerson

```
FORD-FULKERSON( $G, s, t$ )
    for each edge  $(u,v)$  in  $E(G)$ 
        do  $f[u, v] = 0$ 
            $f[v, u] = 0$ 
    while there is a path  $p$  from  $s$  to  $t$  in the residual network  $G_f$ 
         $cf(p) = \min\{cf(u, v)-f[u, v]: (u, v) \text{ is in } p\}$ 
        for each edge  $(u, v)$  on  $p$ 
            if  $(u, v)$  in  $E$ 
                 $f[u, v] = f[u, v] + cf(p)$ 
            else  $f[v, u] = f[v, u] - cf(p)$ 
```

3 Complexities

Dijkstra's $\Theta(V) \times T_{\text{Extract-Min}} + \Theta(E) \times T_{\text{Decrease-Key}}$

Bellman-Ford $\mathcal{O}(VE)$

Floyd Warshall $\mathcal{O}(V^3)$

Ford-Fulkerson $\mathcal{O}(E|f^*|)$

Edmond Carp $\mathcal{O}(VE^2) = \mathcal{O}(V^5)$

Homework #1

Illya Starikov

Due Date: February 3, 2016

1 Insertion Sort

1.1 Operations of Insertion Sort

3 <u>41</u> 52 26 38 57 9 49	1 Comparison(s).
3 41 <u>52</u> 26 38 57 9 49	1 Comparison(s).
3 41 52 <u>26</u> 38 57 9 49	3 Comparison(s).
3 26 41 52 <u>38</u> 57 9 49	3 Comparison(s).
3 26 38 41 52 <u>57</u> 9 49	1 Comparison(s).
3 26 38 41 52 57 <u>9</u> 49	6 Comparison(s).
3 9 26 38 41 52 57 <u>49</u>	3 Comparison(s).
3 9 26 38 41 49 52 57	0 Comparison(s).

1.2 Number of Comparisons

18 total operations.

2 Question 2

B	A	O	o	Ω	ω	Θ
2^n	$2^{\frac{n}{2}}$	no	no	yes	yes	no
$n^{\lg c}$	$c^{\lg n}$	yes	no	yes	no	yes

2.1 Justification

To determine larger asymptotic growth, take the limit of one function over the other. Arbitrarily choosing 2^n for the numerator, we see that:

$$\lim_{n \rightarrow \infty} \frac{2^n}{2^{\frac{n}{2}}} = \lim_{n \rightarrow \infty} 2^{\frac{n}{2}} = \infty \quad (1)$$

Alternatively, if we chose 2^n as the denominator, we notice that $\lim_{n \rightarrow \infty} \frac{2^{\frac{n}{2}}}{2^n} = 0$, so we know that the condition $2^n > 2^{\frac{n}{2}}$ holds, and for certain conditions the functions are equal (take $n = 0, 2^0 = 2^{\frac{0}{2}}$).
 $\therefore 2^n$ is $\omega(2^{\frac{n}{2}})$ and $\Omega(2^{\frac{n}{2}})$. QED.

2.2 Justification II

By the properties of logarithms, we will show that $n^{\lg c} = c^{\lg n}$.

$$n^{\lg c} = c^{\lg n} \quad (2)$$

$$= c^{\log_2 n} \quad (3)$$

$$= c^{\frac{\ln n}{\ln 2}} \quad (4)$$

$$= e^{\ln c \times \frac{\ln n}{\ln 2}} \quad (5)$$

$$= e^{\frac{\ln c}{\ln 2} \times \ln n} \quad (6)$$

$$= e^{\lg c \ln n} \quad (7)$$

$$= n^{\lg c} \quad (8)$$

$\therefore n^{\lg c}$ is $O(c^{\lg n})$, $\Omega(c^{\lg n})$ and $\Theta(c^{\lg n})$. QED.

3 Big-O Implies Big-Ω

Theorem: Let $f(n)$ and $g(n)$ be asymptotically positive functions, $O(g(n))$ be the set $\{f(n) : \exists c, n_0 \in \mathbb{R}^+, \forall n, n_0 \in \mathbb{R}, 0 \leq f(n) \leq c g(n) \wedge n > n_0\}$, and $\Omega(g(n))$ be the set $\{f(n) : \exists c, n_0 \in \mathbb{R}^+, \forall n, n_0 \in \mathbb{R}^+, 0 \leq c g(n) \leq f(n) \wedge n > n_0\}$. Then,

$$f(n) = O(g(n)) \Rightarrow g(n) = \Omega(f(n)) \quad (9)$$

[We will prove so by contradiction.]

Proof: Suppose not. That is, suppose

$$\sim [f(n) = O(g(n)) \Rightarrow g(n) = \Omega(f(n))] \quad (10)$$

$$f(n) = O(g(n)) \wedge \sim [g(n) = \Omega(f(n))] \quad (11)$$

Using the formal definition of $\Omega(g(n))$, we can see the negation is as follows:

$$\sim \Omega(g(n)) = \{f(n) : \forall c, n_0 \in \mathbb{R}^+, \exists n, n_0 \in \mathbb{R}^+, 0 > c g(n) > f(n) \wedge n \leq n_0\} \quad (12)$$

This is a contradiction, for there are no c, n_0 such that $0 \leq c g(n) \leq f(n) \wedge 0 > c f(n) > g(n)$ because c, n_0 are defined as *positive* constants, and $f(n), g(n)$ are defined as *positive* functions. Because no such positive function and positive constants exists to satisfy

$$0 > cf(n) > g(n) \quad (13)$$

This has led us to a contradiction. QED.

4 Asymptotic Proofs

4.1 Proof I

Assuming $n > 1$,

$$n^2 \leq 20n^2 + 2n + 5 \quad (14)$$

$$\leq 20n^2 + 2n^2 + 5 \quad (15)$$

$$\leq 27n^2 \quad (16)$$

$\therefore$ For $C = 27, n_0 = 1, 20n^2 + 2n + 5 = O(n^2)$. QED.

4.2 Proof II

Assume $C = 1, n_0 = 1$. $\therefore$ This satisfies the condition that c and n_0 are positive constants such that $0 \leq Cn^2 \leq 5n^2 - 15n + 100 \forall n \geq n_0$. QED.

4.3 Proof III

4.3.1 Lower Bound

Assume $C = 1, n_0 = 1$. $\therefore$ This satisfies the condition that c and n_0 are positive constants such that $0 \leq Cn^2 \leq 5n^2 + 2n \forall n \geq n_0$. QED.

4.3.2 Upper Bound

Assuming $n > 1$,

$$n^2 \leq 5n^2 + 2 \quad (17)$$

$$\leq 5n^2 + 2n^2 \quad (18)$$

$$\leq 7n^2 \quad (19)$$

$\therefore C = 7, n_0 = 1, 5n^2 + 2n = \Theta(n^2)$. QED.

4.4 Proof IV

To prove that $5n + 7 = o(n^2)$ we must show that $\exists c, n_0 \in \mathbb{R}^+, 0 \leq f(n) < Cg(n) \wedge n > n_0$. We do so by showing that $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$. [*with the help of L'Hôpital's Rule*]

$$\lim_{n \rightarrow \infty} \frac{5n + 1}{n^2} \tag{20}$$

$$= \lim_{n \rightarrow \infty} \frac{5}{2n} \tag{21}$$

$$= 0 \tag{22}$$

QED.

Homework #2

Illya Starikov

Date Due: February 15, 2016

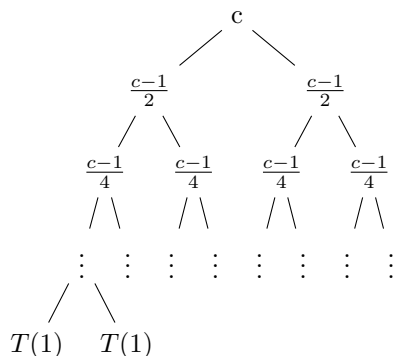
1 Programming Assignment

Code is attached. Binary search implementation is as follows.

```
1  extension Array where Element: Comparable {
2      func binarySearch(key: Element) -> Int? {
3          return binarySearch(key, low: 0, high: self.count - 1)
4      }
5
6      private func binarySearch(key: Element, low: Int, high: Int) -> Int? {
7          if low > high { return nil }
8
9          let middle = Int((low + high) / 2)
10
11         if key == self[middle] {
12             return middle
13         } else if key > self[middle] {
14             return binarySearch(key, low: middle + 1, high: high)
15         } else {
16             return binarySearch(key, low: low, high: middle - 1)
17         }
18     }
19 }
```

2 Recurrence Equation

2.1 Tower of Hanoi



The complexity of this tree is $O(\lg n)$, or alternatively, $O(2^n)$

2.2 Merge Sort

$$T(n) = 2 T\left(\frac{n}{2}\right) + n \quad (1)$$

From the equation, we can see that

$$a = 2, b = 2, c = 1, f(n) = n \quad (2)$$

We observe that

$$f(n) \in \Theta(n^{\log_b a}) = \Theta(n^{\log_2 2}) = \Theta(n) \quad (3)$$

It follows from Case #2 of the Master Theorem

$$T(n) \in \Theta(n^{\log_b a} \lg n) \quad (4)$$

$$= \Theta(n^{\log_2 2} \lg n) \quad (5)$$

$$= \Theta(n^1 \lg n) \quad (6)$$

$$= \Theta(n \lg n) \quad (7)$$

Thus the recurrence relation $T(n)$ is in $\Theta(n \log n)$. QED.

3 Loop Invariant

```

1 Merge(A, p, q, r)
2   leftIndex = q - p + 1
3   rightIndex = r - q
4   let L[1..left + 1] and R[1..right + 1] be new arrays
5   for i = 1 to leftIndex
```



```

6         L[i] = A[p + i - 1]
7     for j = 1 to rightIndex
8         R[j] = A[q + j]
9     L[leftIndex + 1] = sentinel
10    R[rightIndex + 1] = sentinel
11    i = 1
12    j = 1
13    for k = p to r
14        if L[i] <= R[j]
15            A[k] = L[i]
16            i = i + 1
17        else A[k] = R[j]
18            j = j + 1

```

At the end of the for loops on 4-5, 6-7, new arrays will be created, holding the left and right half of the data in the passed array.

Initialization Initially we have two arrays, the left and right side of the original passed array. This holds trivially.

Maintenance During each iteration, we copy over the array's data.

Termination Upon termination, the Left array has the data from $[p + i - 1]$ and Right array has the data of $[q + j]$.

During lines 12 - 17, we merge the arrays to get a properly sorted array.

Initialization Initially we have the left and right arrays, unsorted, and the original array. This holds trivially.

Maintenance During the iterations of the array, we replace the data of the originally passed array with the smaller of the Left and Right array.

Termination Upon termination, we have a fully sorted array from $q...r$.

4 Quicksort

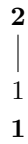
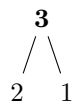
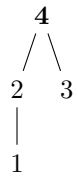
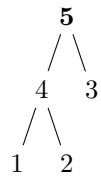
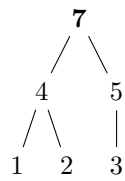
Let $\_$ signify the pivot, $\|$ signify the wall.

1. $\|$ 4 2 7 6 3 5 1
2. 1 $\|$ $\|$ 4 2 7 6 3 5
3. 1 $\|$ 4 $\|$ 2 7 6 3 5
4. 1 $\|$ 4 2 $\|$ 7 6 3 5
5. 1 $\|$ 4 2 3 $\|$ 7 6 5
6. 1 $\|$ 4 2 3 $\|$ 5 $\|$ 7 6
7. 1 $\|$ $\|$ 4 3 $\|$ 5 $\|$ 7 6

8. 1 || 2 || 4 3 || 5 || 7 6
9. 1 || 2 || 3 || 4 || 5 || 7 6
10. 1 || 2 || 3 || 4 || 5 || || 7 6
11. 1 || 2 || 3 || 4 || 5 || 6 || 7
12. 1 || 2 || 3 || 4 || 5 || 6 || 7

5 Heapsort

5.1 Heap Representation



5.2 Enumerated Steps

1. 7
2. 7 5
3. 7 5 4

4. 7 5 4 3

5. 7 5 4 3 2

6. 7 5 4 3 2 1

Homework #3

Illya Starikov

Due Date: February 28, 2016

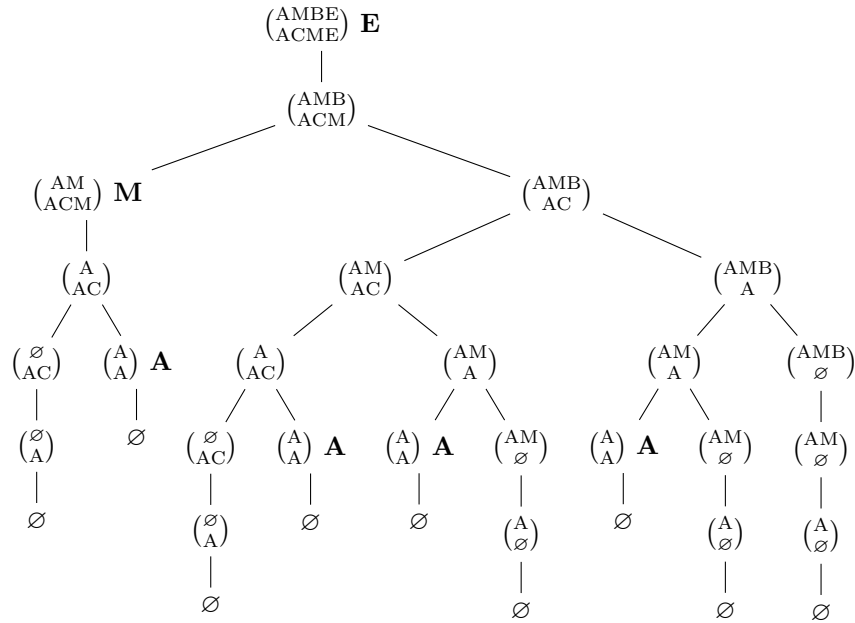
1 Matrix Multiplication

	1	2	3	4
1	0	$20^{[1]}$	$35^{[2]}$	$65^{[3]}$
2	X	0	$12^{[2]}$	$28^{[3]}$
3	X	X	0	$6^{[3]}$
4	X	X	X	0

Therefore, the minimum scalar operations can be achieved via $((A_1 \times A_2) \times A_3) \times A_4$

2 Longest Common Subsequence

2.1 Recursion Tree



Therefore, the longest common substring is AME.

2.1.1 Trace Table

	$\emptyset$	A	M	B	E
$\emptyset$	0	0	0	0	0
A	0	1	1	1	1
C	0	1	2	1	1
M	0	1	2	2	2
E	0	1	2	2	3

Therefore, the longest common substring is AME.

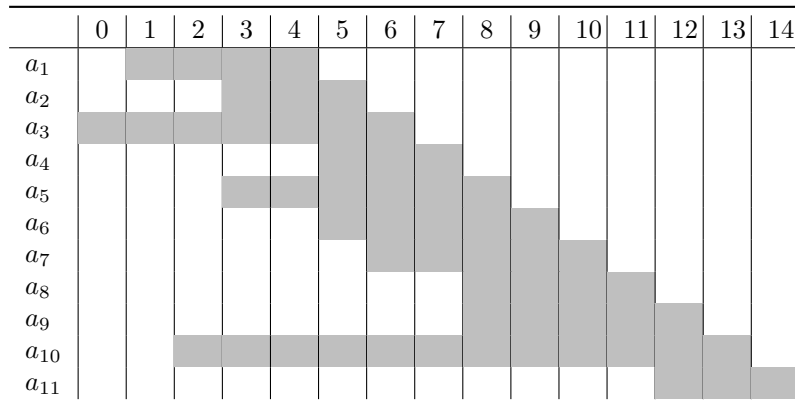
Homework #4

Illya Starikov

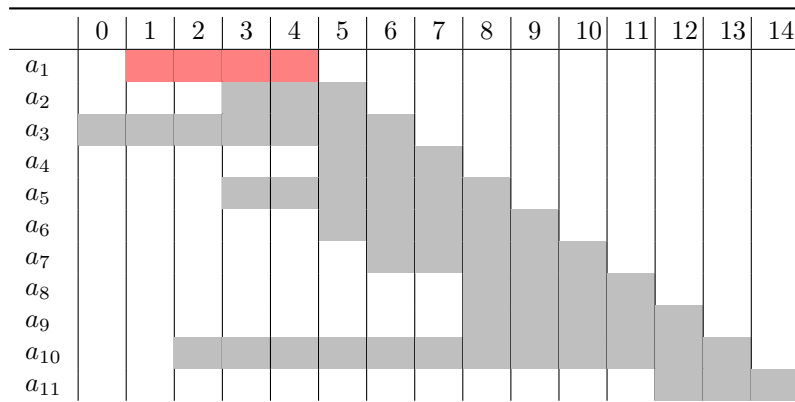
Due Date: March 23, 2016

1 Activity Selection Problem

Plotting this out on a graph,



We can plainly see the optimal solution to be $\{a_1, a_4, a_8, a_{11}\}$. We get this solution by using the ending first greedy algorithm. Here are the steps to obtain the solution.

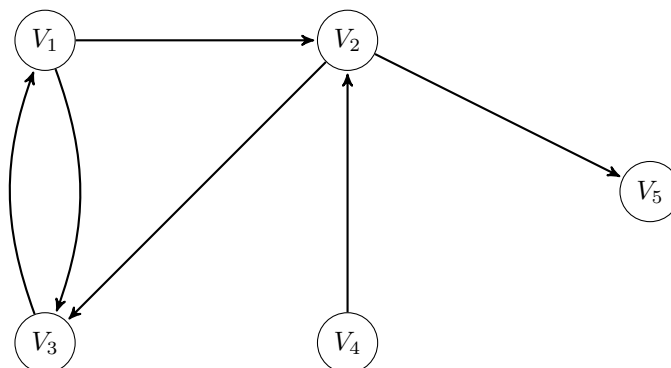


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a_1															
a_2															
a_3															
a_4															
a_5															
a_6															
a_7															
a_8															
a_9															
a_{10}															
a_{11}															

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a_1															
a_2															
a_3															
a_4															
a_5															
a_6															
a_7															
a_8															
a_9															
a_{10}															
a_{11}															

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a_1															
a_2															
a_3															
a_4															
a_5															
a_6															
a_7															
a_8															
a_9															
a_{10}															
a_{11}															

2 Graph And Depth First Search



2.1 Depth First Search

Ouput: V_1 V_2 V_3 V_5 V_4
 Distance: 0 1 1 2 ∞

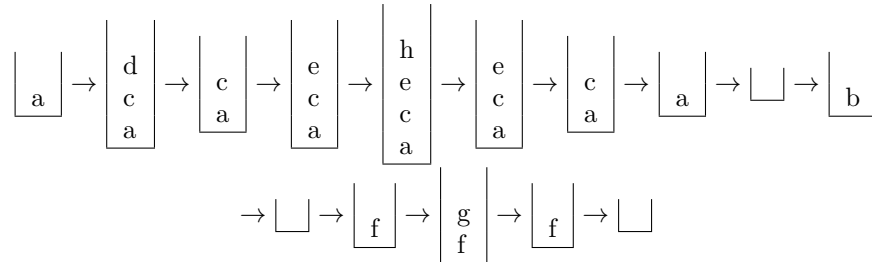
2.1.1 Queue Status

V_1
V_2 V_3
V_3 V_5
V_5
V_4

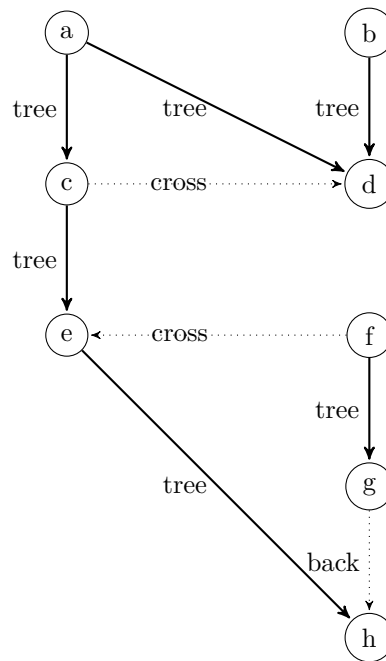
3 Depth First Search and Topological Sorting

Ouput	Discovery	Finish
a	1	10
c	2	9
d	3	4
e	5	8
h	6	7
b	11	12
f	13	16
g	14	15

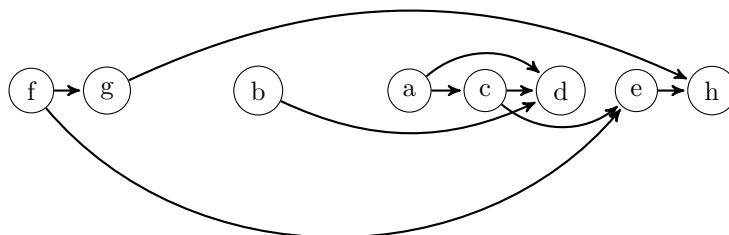
3.0.1 Queue Status



3.0.2 Edge Classification



3.0.3 Topological Sorting



4 Kruskal's and Prim's Algorithm

Please note that my tables are different, however the algorithm is the same is the same.

4.1 Kruskal

Kruskal's works the same way, except my column headers are more verbose and there an Iteration Number column, making it easy to keep track of large algorithms.

Iteration Number	Edge	Weight	Action Taken
1	$\{u, y\}$	1	Added
1	$\{u, x\}$	1	Added
2	$\{s, t\}$	2	Added
3	$\{t, y\}$	2	Added
4	$\{s, u\}$	3	Not Added
5	$\{s, x\}$	3	Not Added
6	$\{t, x\}$	3	Not Added

4.2 Prim

Prim's is also the same, with the exception that my headers are also verbose and the table is sorted linearly by the vertexes and edges that are added. The process is still the same of starting a vertex and branching out based on the best possible choice at the moment.

Iteration Number	Vertex Added	Edge Added	Weight
0	s		
1	t	$\{s, t\}$	2
2	x	$\{t, y\}$	2
3	u	$\{y, u\}$	1
4	y	$\{u, x\}$	1

Both algorithms produce the same result.

Programming Project I, First Report

Illya Starikov, Claire Trebing, & Timothy Ott

Due Date: March 07, 2016

1 Abstract

Smartphone users launch many apps everyday, however one of the most fundamental things a smartphone does is abstracted away: memory management.

Although smartphones have advanced significantly in many ways compared to their first predecessors (in terms of RAM, architecture, processors), deactivation, (the process of “the operating system needing to choose and remove some apps from the memory”, a subproblem of **memory management**) is a solution that is often less-than-perfect. Although Java’s **Garbage Collection** and Swift’s **Automatic Reference Counting** (ARC) have sufficed, there are other methods.

In this project I propose to solve this problem using three techniques:

- Brute Force
- Dynamic Programming
- Greedy Solution

2 Introduction and Motivation

As stated previously, memory management is solved in a less-than-perfect manner. Although current technology suffices, we would like to compare algorithms to show the significant gains via three different approaches (Brute Force, Dynamics Programming, and Greedy).

3 Proposed Solution

For our project we decided to take a more **skeuomorphic** and object oriented approach, modeling objects after their real world counterparts, such as Application or Smartphone. As for the approaches, we have the following solutions:

3.1 Brute Force

For the brute force method, we knew that we had to check every possible subset (and for a set of size n , we know there to be 2^n subsets). We used this to our advantage, creating a lookup table modeled after a truth table. As an example, suppose we have a three items in our knapsack:

Items A	Item B	Item C	Item Number
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

We notice that the knapsack combination can be directly summarized by the n -bit binary representation. Using the previous example of three item knapsack:

$$\text{Subset \#0 : } 0_{10} = 000_2 = \sim (ABC) = \text{No Items} \quad (1)$$

$$\text{Subset \#5 : } 5_{10} = 101_2 = A \sim (B)C = \text{Items A, B} \quad (2)$$

$$\text{Subset \#7 : } 7_{10} = 111_2 = ABC = \text{Items A, B, C} \quad (3)$$

Now that we have a representation of every subset, we can multiply the columns value by the knapsack value to get the item's value into the lookup table. Extending the example, suppose we have the following table:

App	Memory	Cost
A	2	4
B	3	6
C	7	9

Multiplying the columns by the values produces the following results:

App A	App B	App C		App A	App B	App C
0×4	0×6	0×9		0	0	0
0×4	0×6	1×9		0	0	9
0×4	1×6	0×9		0	6	0
0×4	1×6	1×9	=	0	6	9
1×4	0×6	0×9		4	0	0
1×4	0×6	1×9		4	0	9
1×4	1×6	0×9		4	6	0
1×4	1×6	1×9		4	6	9

Adding the columns across we produce the following, and factoring the weight:

Items A	Item B	Item C	Total Cost	Total Memory
0	0	0	0	0
0	0	9	9	7
0	6	0	6	3
0	6	9	15	10
4	0	0	4	2
4	0	9	12	9
4	6	0	10	5
4	6	9	19	12

Now finding the minimal value is straightforward, find the minimal value is traversing down the additive value while looking at the size it frees up. Suppose we have to free up 10 blocks of memory, items A, B would be the most efficient choice.

The pseudocode is as follows, with a few notes:

- Comparison is done while “generating” a table.
- A table is not generate, but the binary representation is used instead.

3.1.1 Pseudocode

```

knapsackBrute(items, knapsackSize)
    mininum = 0
    for i = 0 to  $2^n - 1$ 
        subset = binaryToInteger(i)
        sum = 0

        for i to subset.length
            sum = sum + item[i].benefit * subset[i]
            size = size + item[i].weight * subset[i]

        if size <= knapsackSize && sum < mininum
            mininum = sum
            greatestSubset = i

    subset = binaryToInteger(greatestSubset)
    for i = 0 to subset.size
        if subset[i] == 1
            optimalSolution.append(item[i])

    return optimalSolution

```

3.1.2 Time Complexity

Analyzing this algorithm, we can see the complexity:

$$\sum_{k=1}^n 2^k \times \sum_{k=1}^n c = \mathcal{O}(n2^n) \quad (4)$$

We run in $\mathcal{O}(n2^n)$ time.

3.2 Dynamic Programming

The approach to dynamic programming came from creating a table of the possible memory and cost combinations. The table would contain rows for the values from 0 to M, the largest possible memory we would have to free up. The program starts by finding the lowest amount of memory an app can take up and the lowest cost. This will act as a base case. For example, if the lowest memory amount is 3 and the cost is 1, then if the user needed to clear out 1, 2, or 3 memory spaces then the phone would close that app. Amounts greater than 4 would be calculated when the program started.

3.2.1 Pseudocode

Create_Delete_Table(numberOfApps, numberToDelete, Memory[], Cost[]){

Let d[1 ... M, 3] be new table as shown below

Memory To Delete | Nr. of Apps | Cost | List of Apps Used

0

1

2

3

...

M

int low = findMin(d[], Memory[], Cost[]);

//This function will find a base case for an app with
the lowest memory and lowest cost

for (j = 1 to low)

{

d[j, 0] = d[low, 0];

d[j, 1] = d[low, 1];

d[j, 2] = d[low, 2];

d[j, 3] = d[low, 3];

}

for(j = low to M)

{

for(i = 1 to N){

if(Mi == j && Ci < d[j-low,2] + d[low,2]){

d[j, 1] = 1;

d[j, 2] = Ci;

```

        d[j, 3] = i;
    }

    else{
        d[j,1] = d[j-low,1]++;
        d[j,2] = d[j-low,2] + d[low,2];
        //add app to the list
    }
}
}
return;
}

```

3.2.2 Time Complexity

We can see that this algorithm is $\mathcal{O}(n^2)$.

3.3 Greedy Solution

After considering several greedy algorithm solutions we settled on selecting based on a ratio of cost and memory freed by deactivating a particular app. Once each of the apps' ratio has been evaluated (and the ratio has been stored as a member variable) we sort the array of apps in an ascending order according to this new value. Because the apps are now in order in ascending cost per unit of memory we can now simply add apps to the solution set in the order they appear in the array until our memory requirements are met. For example, if we take our example set of three items:

App	Memory	Cost
A	2	4
B	3	6
C	7	9

We first take the ratio of each item's cost over the capacity of memory it takes up and sort the elements accordingly. Doing this gives us the following updated table:

App	Memory	Cost	Ratio
C	7	9	1.29
A	2	4	2
B	3	6	2

And now we simply add apps until our memory requirement has been met or exceeded. Suppose we need to free up 9 units of memory, according to this approach the optimal solution would be items C and A.

It is worth noting, however, that this solution doesn't always produce the absolute optimal solution. If we use the previous example of freeing up 10 units

of memory this algorithm produces a solution of all the apps, A, B and C as opposed to the actual optimal solution of app C and B. This inaccuracy is a sacrifice in order to benefit from a much lower runtime than that of the other solutions.

We could perhaps introduce a second greedy choice or to further refine our algorithm to improve the accuracy but we would do so at the cost of performance and the solutions chosen by our initial solution are accurate enough for our purposes.

3.3.1 Pseudocode

```
knapsackGreedy()
  A[] = Array of Apps
  S[] = Array of solution set
  M = Desired amount of memory to be freed
  sm = Amount of memory freed by solution set

  n = A.length

  for 0 to n
    A[i].ratio = A[i].cost/A[i].memory

  mergesort(A)

  GreedKnap(A, S)
    S[0] = A[0]
    sm = A[0].memory
    i = 1
    while (sm < M AND i < n)
      S[i] = A[i]
      sm += A[i].memory
```

3.3.2 Time Complexity

Analyzing this algorithm, including both going over the array to produce the ratio that will guide our greedy choices and choosing the actual solution set, we come up with the complexity of:

$$\sum_{k=1}^n k + \sum_{k=1}^n k = \mathcal{O}(2n) \quad (5)$$

This algorithm therefore runs in $\mathcal{O}(2n)$ time.

4 Plan of Experiments

The main objective of our experiments is to extrapolate time and efficiency from the algorithms for sufficiently large input, so our plan of experiments is thus:

1. Generate a random array of apps, and a smartphone with n memory.
2. Take a date stamp.
3. Run the Brute Force algorithm for the random array of apps and the smartphone.
4. Take another date stamp, record the difference.
5. Return to step 2 and run the Dynamic Programming algorithm.
6. Likewise, return to step 2 and run the Greedy algorithm.
7. Record the results from the experiment.
8. Return to step 1, repeat 9 additional times, take the average.
9. Run the algorithm again for a scale factor of n (the input size).

5 Team Roles

Illya Starikov Project Management, Development (Brute Force)

Timothy Ott Development (Greedy), Architecture

Claire Trebing Development (Dynamic), Quality Assurance, Documentation

Programming Project I, Second Report

Illya Starikov, Claire Trebing, & Timothy Ott

Due Date: March 17, 2016

1 Abstract

As proposed in the first report, we would like to compare the efficiency of different memory management methods via a **0-1 Knapsack**. We discussed the current memory management methods, the different approaches we would like to use (Brute Force, Dynamic Programming, Greedy Solution) and our plan of implementations.

After implementing the pseudocode provided in Report #1, rigorous testing and extensive running of sample of data, we have completed conclusive evidence to our proposed problem.

In this report we would like to discuss our findings.

2 Implementation

As we discussed in our earlier report we decided to go with a more object-oriented approach to this problem. We created a class called App to represent each application running on our platform. Each instance of the App class contains (as properties¹) an identifying number, its memory usage, and its cost to recover from storage. These properties are largely randomized in the construction of each instance. For example, every app has a random memory usage in the range of 32 to 1028; the cost is a derived value, taking anywhere from 20 to 50 percent of the memory usage of the App in question; we also have a property called ratio which consists of the cost property divided by the memory property and is calculated at creation of the instance by the constructor, this will be used in our Greedy Algorithm; and lastly each App is given an identifying number based on the index of the cell in the array in which they are initially placed. This ensures that each App will have a unique identifier to help differentiate them whilst checking our solutions against one another.

Initially we considered creating a separate class for our “Smartphone” or the platform on which our Apps would be running but later decided against this as this functionality could easily be represented simply by a list (or array) of all the Apps, cutting down on the amount of code we would need.

¹Or member variables, depending on your preferred notation.

We implemented our three solutions as one overarching test program that produces arrays of apps in ever increasing input sizes and feeding those arrays to our algorithms one at a time, testing the average runtime over 10 iterations. The general flow of our program is as follows:

First we specify a list of input sizes that our program will act on, we will discuss this in more detail further along but for now it should be noted that our first input size is 5. Our program then creates an array of newly created Apps of the specified input size². These newly created apps have each of their properties randomly assigned by its constructor. Next, because our Greedy Solution requires the inputs be sorted based on its ratio variable and the other solutions do not, we sort the list using Python’s built in sort function.

Now we simply feed the newly sorted list to each of the solution functions in turn, timing the run time of each by taking a timestamp before running the algorithm, again after, then subtracting the two times. We run each program ten times per input and take the average of these times for greater accuracy. Once each solution algorithm has run ten times the program outputs the data (which includes the solution set, the amount of memory freed by the solution, the combined cost of the solution and the average time it took) to two text files in the directory of the program.

It is at this point that our program returns to the top of the loop for the next input. To ensure a adequate spread of input sizes we defined a list of “base inputs” which we initialized as 5, 10, 15, and 25. Once the program finishes iterating over this list, testing each solution as mentioned above, the program returns to the beginning of the input list and multiplies each by 10 to the power of n , where n is incremented with each new round of the loop back to the beginning of the base inputs. This entire process is placed in a while loop that is set to infinitely repeat so that we are able to get as many data sets as the program can output in the time we have allotted for testing.

3 Experiment

As described above, the tests were run for a twelve hour period getting our input size to 1,000. Noting that the for input size of 1,000 it took the Dynamic Programming approach 1334.81 seconds, the total time to get accurate results was *3 hours, 42 minutes, and 28 seconds*³. We thought this was a good cut off point.

To ensure accuracy, we decided to run the tests on two independent machines. These tests can be seen on the graphs, but not the data sets. The tests were fairly comparable, so we decided that displaying data from one data set was sufficient.

We noticed that the Brute Force solution was taking a drastically longer time to run (the complexity being $n \times 2^n$). After a sample run, the Brute Force algorithm returned a Memory Error for an input size of 50. Seeing as we were

²Again, in our first case this is 5.

³1334.81 seconds $\times$ 10 iterations = 13348.1 seconds = 222.47 minutes.

not likely to be able to have sufficient enough time, we decided to limit the Brute Force to a maximum of 25 inputs. Below are the results.

3.1 Results Data Set

Below you can find the results from *one of our test*. As stated above, the Brute Force only goes to 25 inputs. Note that the input column is the Input size of applications, the Algorithm is the type of algorithm we used, the Memory was the goal memory we wanted to produce, the Cost was what we wanted to minimize, the Freed column is the memory that was actually freed (because we could go over the goal if it was an optimal solution) and finally the Time is the time taken (strictly of the algorithm in question) measured in seconds.

Input	Algorithm	Memory	Cost	Freed	Time
5	BRUTE	609	206.960013686	769	0.0000652
	DYNAMIC	609	413.920027372	1538	0.008588
	GREEDY	609	206.960013686	769	0.00572586666667
10	BRUTE	1254.0	306.007083914	1307	0.0023169
	DYNAMIC	1254.0	913.397426561	3126	0.033041
	GREEDY	1254.0	418.629747	1961	0.0220281
15	BRUTE	1896.2	462.268333337	1980	0.0933534
	DYNAMIC	1896.2	706.822856172	3129	0.10722175
	GREEDY	1896.2	506.866912451	2383	0.0714820333333
25	BRUTE	2469.0	619.523226819	2479	113.6533167
	DYNAMIC	2469.0	1204.68191427	4035	56.92951935
	GREEDY	2469.0	620.454984919	2562	37.9530137333
50	DYNAMIC	5421.2	1664.89861527	7120	0.870269
	GREEDY	5421.2	1177.03320938	5519	0.43513695
100	DYNAMIC	9783.2	2809.24740865	11268	3.8071061
	GREEDY	9783.2	2392.11409139	9849	1.90355725
150	DYNAMIC	14995.8	4027.50565053	16594	9.4551353
	GREEDY	14995.8	3696.37456362	15462	4.7275798
250	DYNAMIC	26793.4	6591.81933106	28200	32.8375575
	GREEDY	26793.4	6272.26215744	27018	16.41878765
500	DYNAMIC	52354.6	12547.9234913	53429	192.5988691
	GREEDY	52354.6	12365.0250774	52842	96.2994775
1000	DYNAMIC	107148.2	25671.8344823	109000	1334.8144697
	GREEDY	107148.2	25256.6939401	107895	667.4074229

3.2 Results

While looking at the data we got back, we notice a very noticeable trend immediately: the exponential nature of the quasi-exponential nature of the algorithms. When plotting on an input size vs. time graph, we could see even the algorithm with $\mathcal{O}(n)$ (the Greedy Solution) complexity showed some exponential growth. Taking into consideration the [RAM Model of Computing](#) is not completely applicable in this situation, the results are more than staggering.

As can be seen in the above figure, although the input follows the pattern of $5n, 10n, 15n, 25n | n \in \mathbb{Z}^+$, the results typically follow a pattern of $2\times$ their predecessor.

When comparing the Time vs. Costs, the trend is also quite exponential. For, as our algorithms get a larger cost, the time is unquestionably larger. When actually looking at the Time vs. Accuracy, the clear cut winner is Greedy. The time saving are immense compared to Dynamic and Brute Force. On certain cases, Greedy can finish before Dynamic and Brute force combined.

We also see that Greedy Solution outperformed the Dynamic in not only time but accuracy. We are unsure of the reasoning behind this, for our algorithm certainly seemed correct on paper.

Tying this back into the real world example, none of these solution would be able to work. Assuming a smartphone of 2 GB RAM⁴, figuring an optimal solution would take a bare minimum of 38 seconds (not to mention 2 minutes if we are using the Brute Force.)

While this may not be applicable to the real world, it was an interesting thought experiment.

4 Roles

- Illya Starikov
 - Project Manager
 - Brute Force Development
 - Abstract, Experiment Writement
- Timothy Ott
 - Greedy Algorithm Development
 - Development, Conclusion, Interesting Results Writeup
 - Testing Documents

⁴We are assuming our memory is equivalent to 1 MB. Therefore, 2 GB = 2000 MB

- Claire Trebing
 - Dynamic Programming Development
 - Experiment Writing Up
 - Quality Assurance

5 Interesting Results

In the process of going over the data generated in our experiments we noted several interesting results that we would like to highlight here.

- We noticed that for input sizes under roughly 15 the Brute Force method was both more accurate, which we expected, but also significantly faster than the other two methods. When we take into consideration that the Brute Force's complexity is $\mathcal{O}(n2^n)$ and the Greedy Solution is $\mathcal{O}(2n)$, this is an excellent highlight to the fact that the growth of the function is far more important than its initial run speeds.
- There is an interesting anomaly in the data set for the input size of 25. At this point the run times for each algorithm spike noticeably only to then immediately dip once again. What's more interesting is that as we mentioned earlier we ran these experiments twice, independently of one another on two different machines and this anomaly occurred in both rounds of experimentation. So, what is so special about 25 inputs? The short answer is we are not sure. Our best guess is that the increase and subsequent drop are somehow related to the Brute Force algorithm since that is where we stopped testing that particular algorithm. Particularly we think this may have been the fault of the language's garbage collection functionality but without further testing we cannot be sure.
- Also of note is that while initially our dynamic approach was a good deal less accurate than either the Brute Force or even the Greedy approach, as the input sizes increased the accuracy of the dynamic algorithm also increased.

6 Conclusion

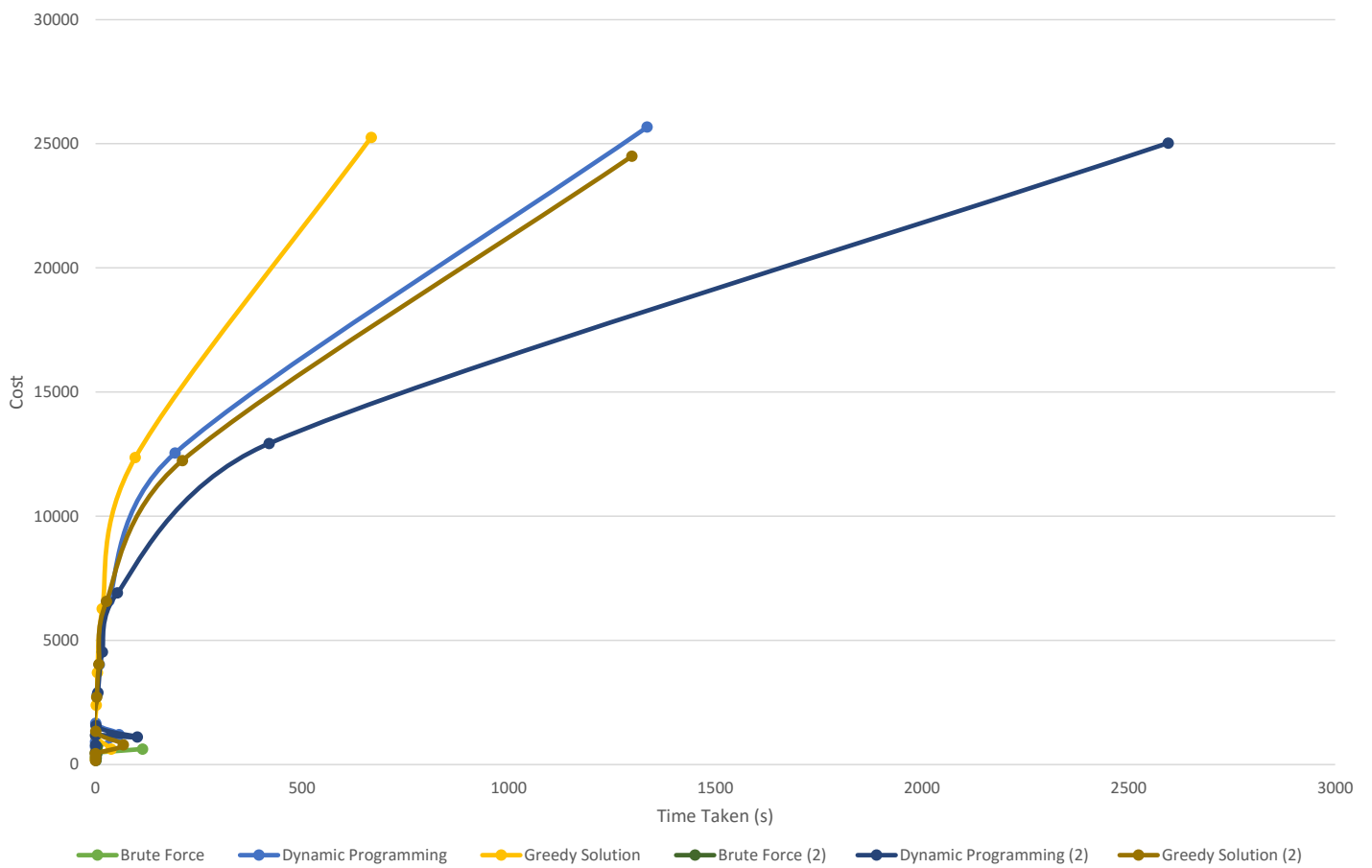
Memory management is a complex issue with many interesting solutions of varying effectiveness. With this project we explored three approaches to solve this issue using a smartphone's memory management system as our testing ground. Our solutions took the forms of:

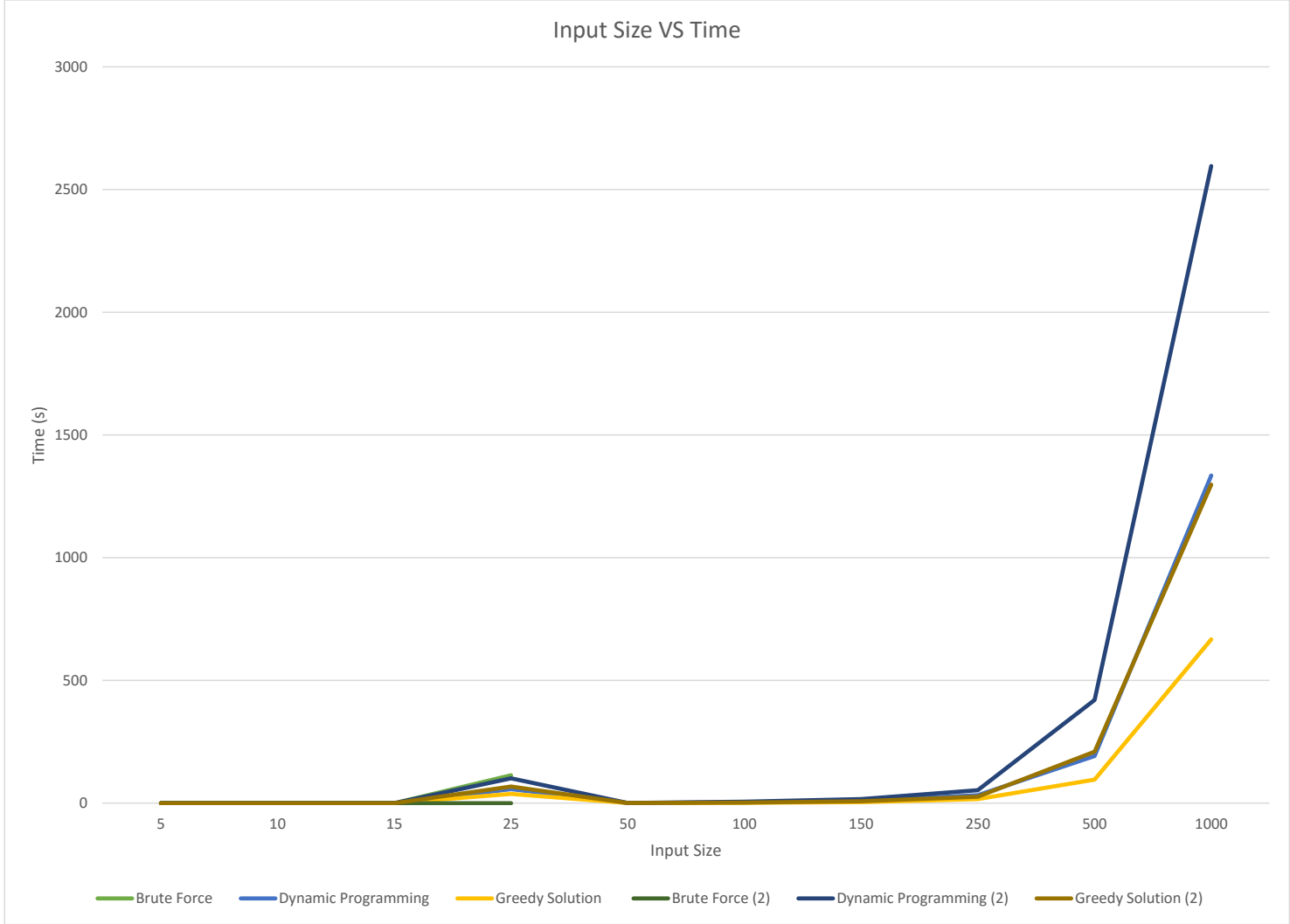
- A brute force approach, comparing every possible subset of solutions to find an optimal choice at the expense of a high complexity and eventual run-time;

- A dynamic programming solution which built a table of solutions to effectively try every solution but with a lower complexity due to being able to look up values that were already computed; and
- A greedy algorithm approach which took a ratio of the cost and memory and produced a solution that was simply taking the lowest ratio apps until the memory requirements were met, in effect trading the accuracy of the other two approaches for a significant increase in speed.

Through our data gathered during experimentation we see that these early predictions hold fairly consistently, despite some unaccounted for inaccuracy on the part of our dynamic solution. While it is tempting to label one solution as the “best” or “optimal” solution, this would be misleading as each approach shines under different circumstances. For instance, for small input sizes the brute force algorithm seemed to exceed both other solutions in terms of both accuracy and speed. For higher input sets the greedy algorithm was the clear winner in terms of speed but theoretically would lose out on accuracy to the dynamic solution every time. Thus, we see that is important to tailor your solutions to the problem and not simply favor one approach over another arbitrarily.

Time VS Cost





CS3001

Skills Development



Illya Starikov (isgx2)

CS5400-SP2018 Puzzle3 Grading Rubric

Datastructures: 100.00%

Looks good.

Action & state generation: 100.00%

Looks good.

Search Algorithm: 100.00%

Looks good.

Programming Practice: 100.00%

Good documentation!

Code reliability: 100.00%

Good.

Bonus: 100.00%

Good analysis!

**Total = 3.00%(Datastructures) + 2.00%(Action & state generation) + 75.00%(Search Algorithm) +
10.00%(Programming Practice) + 10.00%(Code reliability) + 15.00%(Bonus)**

Total: 115.00%

CS3100

Software Engineering I



Camelot System Requirements

All relevant documentation for system requirements.
Created and maintained by Ian Howell, Hunter Mathews,
Illya Starikov, William Thurman, Zachary Wileman. *Server
Team #1*

Last Revised: *March 25, 2017*
ILLYA STARIKOV

Michael Gosnell
Immediate Supervisor
mrghx4@mst.edu

Status: **Validated**
Version: **1.2**

Contents

1	Introduction	3
1.1	Purpose	3
1.2	How to Use This Document	3
1.2.1	Types of Reader	3
1.2.2	Technical Background Required	3
1.2.3	Overview Sections	4
1.2.4	Reader-Specific Sections	4
1.3	Scope of the Product	4
1.4	Business Case for the Product	4
1.5	Overview of the Requirements Document	5
2	Description	6
2.1	Product Perspective	6
2.2	Product Functions	6
2.2.1	User Authentication and Login	6
2.2.2	Channels	7
2.2.3	Send/Receive Messages	8
2.3	User Characteristics	9
2.4	General Constraints	9
2.5	Assumptions and Dependencies	9
3	Specific Requirements	10
3.1	User Requirements	10
3.1.1	End User Requirements	10
3.1.2	Developer Requirements	10
3.2	System Requirements	10
3.3	Interface Requirements	10
4	References	11
5	Version History	12
6	Server/Client Validation	13
7	Glossary	14

1 Introduction

Camelot is JSON-based server written exclusively in [Python 3](#). The purpose of Camelot is to provide an IRC-esque chatroom on the server, leaving implementation details for the client up open to interpretation.

1.1 Purpose

This document is intended to guide development of Camelot. Not only will it provide the requirements (i.e. system requirements, user requirements, and interface requirements), but it will provide a decent outline for what the server entails (i.e. functions, constraints, dependencies, etc.). Understanding the system requirements will give a more clear and concise understanding of the Application Programming Interfaces (APIs) and their purpose.

1.2 How to Use This Document

Because this document will be reviewed by various different skill sets, this section will break down which parts should be reviewed by which type of reader.

1.2.1 Types of Reader

This document is going to be aimed at two different type of people: developers and end-users. The first set, client/server side developers, will be the Python3 Programmers, server-side programmers, Information Technology (IT) personal, and others that might have communication needs. These people are going to be the ones who wish to build upon or plan to use the code commercially. The other set will be the end-users. These people will be running the server in order to hold a chat session(s).

1.2.2 Technical Background Required

Programming competency *is imperative* to understanding the documentation. Programming methodology, jargon, and general concepts will be assumed in subsequent sections. Server side programming is recommended, but not required.

The document will stay to a high level server design. For the purpose of this document, source code will not appear. Specific tools/programming languages capability is not required.

1.2.3 Overview Sections

To get a general understanding of the document, the following sections and subsections should be (chronologically) read:

1. User Characteristics (Section 2.3)
2. Assumptions and Dependencies (Section 2.5)
3. Specific Requirements (Section 3)

1.2.4 Reader-Specific Sections

Types of readers are described as follows.

- Server-side Developers: This would entail anyone interested in working on this particular instance of the server. All of this document should be read.
- Client-side Developers: This would entail all who would want to get involved in creating a client. The following sections should be read:
 1. Description (Section 2)
 2. Specific Requirements (Section 3)
- End Users: This would entail anyone that is generally interested, but has no intentions of truly using it standalone.
 1. Product Perspective (Section 2.1)
 2. Product Functions (Section 2.2)
 3. User Characteristics (Section 2.3)

1.3 Scope of the Product

Truthfully, the intention of this project is to get an A in software engineering. But I actually have to put something here, so.

There is an expectation at the end of the development lifecycle to open source this server. After open sourcing, the server team hopes that Camelot will serve as a model for server-side programming. Because the JSON-based framework would be familiar to people, the team hopes for some market adoption. Overall, this would be a good model for how a simple server would be maintained.

1.4 Business Case for the Product

There is a real pandemic: there are not enough chat applications. Sure, there's [Messenger](#), [Slack](#), [Skype](#), [Viber](#), [WeChat](#), [WhatsApp](#), [Line](#), [GroupMe](#), [Snapchat](#), [Voxer](#), but they are garbage. The market needs a great server to tackle on these giants, and Camelot will do that.

1.5 Overview of the Requirements Document

1. Raspberry Pi 3 needed to host the server
2. Knowledge of Python to program the server
3. JSON based framework

2 Description

This section will give the reader an overview of the project, including why it was conceived, what it will do when complete, and the types of people we expect will use it. We also list constraints that were faced during development and assumptions we made about how we would proceed.

The Camelot Server project will create a means of communication between chat clients, in order to allow end users to communicate with each other.

2.1 Product Perspective

We have chosen to develop this project in order to create a standard form of communication between the chat clients that are being developed to use the server. The developers will use this in order to create an interface to house the information being transmitted.

2.2 Product Functions

The server will have the following capabilities.

2.2.1 User Authentication and Login

Users Will have the option to either create an account or to log in with an existing account.

Users will be able to create accounts by submitting a username that hasn't been taken. The client will send a JSON encoded file that looks something like:

```
1 {  
2   "create_account": {  
3     "username": "some username",  
4     "password": "some password",  
5     "server_password": "some server password"  
6   }  
7 }
```

If the username has already been taken, then the client will receive a JSON encoded file that looks something like:

```
1 {  
2   "error": "That username is already taken."  
3 }
```

When the client sends a request to login, it should send a JSON encoded package to the server. It should look something like this:

```
1 {
2   "login": {
3     "username": "some username",
4     "password": "some password",
5     "server_password": "some server password"
6   }
7 }
```

If the username doesn't exist or if the password the user enters doesn't match the password associated with the user, then the client will receive a JSON encoded file that looks something like:

```
1 {
2   "error": "The username and/or password do not exist in the database."
3 }
```

There will also be a server password given to each client (as seen above) that the user doesn't have to enter but the client will have to pass along with the user login so that the client can be authenticated.

2.2.2 Channels

Channel Creation

As of right now, there will be a set number of channels created by the server that the clients will have the option of joining. Later on in development, we may add the option for users to create channels. Upon creation, the creator will become the admin of that channel. Each user will have a limit on the number of channels they may create.

Channel Deletion

As of this writing, only the server team will have the option of deleting specific channels. When users gain the ability to create channels, they will also gain the ability to delete channels that they have created¹.

Initial Joining of Channels

After a user has logged in, the server will send the client a list of default channels that the user has the option of joining². It will look something like this:

```
1 {
2   "channels": [
3     "Client_Team",
4     "Server_Team",
5     "Software_Eng."
```

¹The initial channels will be owned by the server team.

²Later on, the user will have the option to search for channels based on a keyword.

```
6     ]
7 }
```

The client will then need to send back a JSON encoded file to the server describing what channels the user would like to join. The JSON file should look something like this:

```
1 {
2     "join_channel": [
3         "Client_Team",
4         "Server_Team"
5     ]
6 }
```

After the user successfully joins the channels, the specified user will have access to the channels that they decided to join.

Joining Channels After Selecting Initial Channels to Join

Users will have the option to list the channels when logged in. They will be able to join channels in the same fashion as they did initially³.

2.2.3 Send/Receive Messages

Whenever the server receives notice that a new message has been posted to a given channel, the server will send out a message to each user who is connected to that given channel. The message will be a JSON file that looks something like this:

```
1 {
2     "new_message": {
3         "channel_receiving_message": "Client Team",
4         "user": "username",
5         "timestamp": "2017-03-14 14:11:30",
6         "message": "the actual message that the user posted"
7     }
8 }
```

When a user (client) wishes to send a message to a certain channel, the JSON object should look something like this:

```
1 {
2     "new_message": {
3         "channel_receiving_message": "Client Team",
4         "user": "username",
5         "timestamp": "2017-03-14 14:11:30",
6         "message": "the actual message that the user posted"
7     }
8 }
```

Notice that both the JSON file being received by the client and the JSON being sent out by the server look exactly the same. This is done so that the server can simply

³Later on, the user will have the option to leave channels.

broadcast messages to all users without having to decode a JSON file server side. This will give the least amount of delay in messages being sent out to each user. Also, the user that sends the message out will also receive the message. It's up to the client in how they want to approach this situation.

2.3 User Characteristics

Most users will be developers who will make clients to interface with the server. They will be mostly of a more technical background, with education background involving computer programming. They will be interfacing with the server in order to create a client for end users to communicate with each other. They may encounter obstacles with reading messages if they have no experience with parsing JSON.

2.4 General Constraints

For the constraints of our server, we didn't have any specific constraints as to what Integrated Development Environment (IDE) each person used or the platform that they developed on. The only specific constraints that we have so far is that the server is going to be developed with Python3 using JSON for data transfer and also that each person use Git for collaborating on the development of the server. We have not run into any issues with making our server compatible with other software as of yet.

2.5 Assumptions and Dependencies

The assumptions made with the development of this project is that each person that will be working on the server has a working knowledge of some programming language (most likely C++) and is willing to learn python. Its also assumed that at least some people working on the server have some background knowledge as to how to program in python as to help others within the server group who aren't as familiar with Python.

3 Specific Requirements

The follow sections will go in-depth about the specific requirements of the Camelot system.

3.1 User Requirements

Our user base will be split across two use cases:

1. The end users, who will be referred to as the user, and
2. The client teams, who will be referred as the developers

3.1.1 End User Requirements

The user will need to have a client that can connect with the Camelot server. They will need to be able to communicate with other users across channels. They should be able to send and receive messages. Messages should be sent and received in a logical order.

3.1.2 Developer Requirements

The developers will require the ability to request a list of channels. The list will contain information regarding the channel names and the users currently in each channel. With a successful login, they should be able to create a unique user with a specific identifier. Users should be deleted upon signing out or error. The developers should be able to request message information, such as message text, senders, requested channel, and timestamps.

3.2 System Requirements

The Camelot server will be run on a Raspberry Pi 3 Model B. It will need need to have internet access. The hardware will need to have Python3 installed as well as PostgreSQL for database management.

3.3 Interface Requirements

The Camelot will need to interface with a client using JSON formatting as a data transfer protocol.

4 References

Below is a list of all relevant documentation/references for this document.

C++ <http://www.cplusplus.com>

Git <https://git-scm.com>

JSON <http://www.json.org>

PostgreSQL <https://www.postgresql.org>

Python3 <https://www.python.org/download/releases/3.0/>

5 Version History

Below are major releases and their respective changes.

- 1.0 Initial release
- 1.1 Bug Fixes
 - Fixed typos, mistakes, improper wording, etc
 - Fixed Pandoc spacing issue where a `\n` would appear after 80 characters
 - Further explained functionality
- 1.2 Added verification chapter

6 Server/Client Validation

The following signatures from partnering client teams will signify the approval of requirements document. This means approval of, but not limited to,

- The API guidelines
- The conceptual design
- The goals and vision for the product

The one signature below is representative of everyone in the group. Signee is held accountable for entirety of client team. Terms and condition apply. Limit to one daily. Side effects include upset stomach, indigestion, and spontaneous combustion.

Server Team #1

Client Team #1

Client Team #7

.....
Signature

Signature

Signature

.....
Date

Date

Date

7 Glossary

APIs Application Programming Interfaces. 3

C++ A general-purpose, object oriented programming language. 9, 11

Git A free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. 9, 11

IDE Integrated Development Environment. 9

IT Information Technology. 3

JSON JSON is a syntax for storing and exchanging data, written with JavaScript object notation. 4, 6–11

PostgreSQL Sophisticated open-source Object-Relational DBMS supporting almost all SQL constructs. 10, 11

Python An interpreted programming language popular for server side programming. 3, 9–11

Camelot Protocol Document

A description of how the system protocol will behave.
Created and maintained by Ian Howell, Hunter Mathews,
Illya Starikov, William Thurman, Zachary Wileman. *Server
Team #1*

Last Revised: *March 14, 2017*
ILLYA STARIKOV

Michael Gosnell
Immediate Supervisor
mrghx4@mst.edu

Status: **Validated**
Version: **1.1**

Contents

1	Authentication	4
1.1	Input	4
1.2	Output	4
2	Channel	5
3	Message	6
4	Edit(s)	7

Contents

In essence, our server will act as an IRC-esque chatroom. Because we are starting with the Minimal Viable Product (MVP), it will not particularly have all the features of an IRC server; also, because we have to provide a JSON-based API, that alters the design drastically. This document will provide a sufficient outline on the authentication process, the basic objects (user, channel, message), and other miscellanies. Required fields are marked by a red asterisk (*), and possibly required items are marked with a blue asterisk (*).

The server will be asynchronous. The messages can be sent from the client at anytime, and will be handled accordingly. The messages should also be able to be received at anytime. It is up to the client to add listeners to be able to receive said messages.

1 Authentication

At the time of the authentication (i.e. the login process) a few item will have to be provided.

1.1 Input

Server Location* The server location must be provided to get access to the server. Duh. **String.**

Server Password* An optional password might be required. **String.**

User* A unique identifier to represent the user. If submitted, and not unique to channel, a randomized one will be returned. Can only consist of a mixture of numbers $[1 \cdots 9]$, letters $([a \cdots z][A \cdots Z])$ or underscores (`_`). **String. String.**

Channels *After* authentication (i.e. after successfully joining the server), channels can be joined. Refer to Section 2 for details. **Array of Objects** `[{ Channel, Password }, ... ]`

1.2 Output

Session Key Will be used in future development. **Array.**

Username Will be used to verify if submitted username was allowed. **String.**

Channel List Success Will return an a array of objects to specify the success of joining the channels. **Array of Objects** `[{Name, Success Status, [Users, ...] }, ...]`.

2 Channel

The channels is the actual location of the chat room. All messages posted by the user(s) will appear here. A channel can act as a direct message mechanism as well, by simply having two users.

Name The name of the channel. The name of the channel must be unique. Can only consist of a mixture of numbers $[1 \cdots 9]$, letters $([a \cdots z][A \cdots Z])$ or underscores (`_`) **String**.

Password If there a password, specify it. If there is no password, any submissions (including SQL injections) will be accepted. **String**.

3 Message

A message is a simple object with what you'd expect a message to have. Simple message object will be sent and received (with all the contents below).

Timestamp The time *the client* sent this message. A JavaScript format time object should be sent. **String.**

Sender* The *username* of the sender. **String.**

Message Actual message contents. No format characters will be accepted (i.e. `\t`, `\n`, or `\r`). **String.**

Channel What channel the message was posted on. **String.**

4 Edit(s)

This section is dedicated compatibility-breaking edit(s).

Illya Starikov – Anonymity Discontinued

Not only is true anonymity hard to implement and frustrating to maintain, a lot of the time it is only a hassle for users and programmers. The only people gaining from anonymity are typically trolls or abusers of the system. From this point forward, the bare minimum one can provide is a username. It is to the clients if they wish to make a randomized ID, but it should be displayed. *Changed on February 28, 2017.*

Camelot Progress Presentation

Software Engineering

Ian Howell, Hunter Mathews, Illya Starikov, William Thurman, Zachary Wileman (*Server Team #1*)

March 24th, 2017

Missouri University of Science and Technology

About the Camelot Server

- Camelot is an asynchronous *Python3* server.
- The server is housed on a Raspberry Pi (as provided by Ian Howell) so that the client team will be able to interact and test their code on it (assuming the server is up and running) and also so that interaction with the server can be done remotely.
- The database is being tackled specifically by a module in Python called *psycpg2* which uses *PostgreSQL*.
- All communication between the server and client applications is done via *JSON* encoded data.

Camelot Inspiration



Figure 1: “On second thought, let’s not go to Camelot. ’Tis a silly place.” — Arthur, Monty Python

What Has Worked Well

Teamwork Task delegation and team work has made us able to accomplish more, in less time. Having teammates with a wide array of complementing skills and backgrounds.

What Has Worked Well

Teamwork Task delegation and team work has made us able to accomplish more, in less time. Having teammates with a wide array of complementing skills and backgrounds.

Meetings Our meetings have been productive, where topics such as documentation, implementation details, workload are discussed.

What Has Worked Well

Teamwork Task delegation and team work has made us able to accomplish more, in less time. Having teammates with a wide array of complementing skills and backgrounds.

Meetings Our meetings have been productive, where topics such as documentation, implementation details, workload are discussed.

Changes Sometimes when you innovate, you make mistakes (i.e. choices for users...). It is best to admit them quickly, and get on with improving your other innovations.

What Has Worked Well

Teamwork Task delegation and team work has made us able to accomplish more, in less time. Having teammates with a wide array of complementing skills and backgrounds.

Meetings Our meetings have been productive, where topics such as documentation, implementation details, workload are discussed.

Changes Sometimes when you innovate, you make mistakes (i.e. choices for users...). It is best to admit them quickly, and get on with improving your other innovations.

Tools By picking some of the best tools (Github, $\text{\LaTeX}$, PostgreSQL), it has made for a much better workflow.

Past Changes

Below are past decisions that if we could change, we would.

Past Changes

Below are past decisions that if we could change, we would.

-
-
-
-
-
-
-
-
-
-

Tools I

- Server code is written in Python3.
 - Documentation is written in Doxygen.
- All development is on Github.
 - This includes meeting notes, documentation, and all production code (i.e. database and server).
 - Github issues/milestones for task management.
 - Contributions to make sure even workload.
- All documentation/presentation is written in $\text{\LaTeX}$.
- Using PostgreSQL for the database.
- Using Discord for team/client chat – by any memes necessary.

Documentation Demo

Tools II

Python enables easy server setup.

```
1     def main():
2         server = ThreadedTCPServer(("0.0.0.0", 9009), ThreadedTCPRequestHandler)
3         ip, port = server.server_address
4         server.socket.listen(10)
5
6         server_thread = threading.Thread(target=server.serve_forever)
7         server_thread.dameon = True
8         server_thread.start()
9         print("Server loop running in thread:", server_thread.name)
10
11     try:
12         # Loop forever
13         while True:
14             pass
15     except KeyboardInterrupt:
16         print("Cleaning up server..")
17         server.shutdown()
18         server.server_close()
19         print("Done! Goodbye")
20
21
```

Server Demo

Github Demo

Challenges

Some of the challenges we encountered this far are:

1. Communication between the client and server teams.

Description & Solution

Even though we have set up a system for communicating between the client and server teams (Discord), there is little to no talk between the teams. *Solution: Communicate more..?*

Challenges

Some of the challenges we encountered this far are:

1. Communication between the client and server teams.
2. Direction and vision of the product.

Description & Solution

Although there is an established, core vision (IRC clone), working out the details has proven to be difficult. *Solution: Break it down to basic principles and work with teammates.*

Challenges

Some of the challenges we encountered this far are:

1. Communication between the client and server teams.
2. Direction and vision of the product.
3. Learning curve of tools, teammates, and project.

Description & Solution

Because we're relatively new to each other, and are unaware of each other's skill sets, assigning tasks becomes tricky. Also learning how to implement new things can be difficult. *Always be sure teammates are comfortable with the tasks they're given.*

Extras

- Encryption.
 - Still debating if we'll be using ROT13 (znlor vs jr tvq thq, jr'yy hfr EBG26.) or an in-house, post-quantum cryptographic hash function using a multivariate-quadratic public-key signature system.
- Open source

Extras

- Encryption.
 - Still debating if we'll be using ROT13 (znlor vs jr tvq thq, jr'yy hfr EBG26.) or an in-house, post-quantum cryptographic hash function using a multivariate-quadratic public-key signature system.
- Open source at the end of the semester.
- Full documentation guide courtesy of Doxygen.

Current and Future Plans

1. Consolidate the client and server teams to get unified protocol and systems specification document.
2. Finish coding the actual chatroom.
 - Server
 - Database
 - Script to automate a chatroom environment
3. Finish documenting the chatroom (possibly API guide).

Current and Future Plans

1. Consolidate the client and server teams to get unified protocol and systems specification document.
2. Finish coding the actual chatroom.
 - Server
 - Database
 - Script to automate a chatroom environment
3. Finish documenting the chatroom (possibly API guide).
4. ...
5. Profit

In Closing

Spam Illya with all questions, comments, and insults.

 [@IllyaStarikov](#)

 starikov@mst.com

Special thanks to our awesome team.

 Ian Howell

 Hunter Mathews

 Illya Starikov

 William Thurman

 Zachary Wileman

Camelot Final Presentation

Software Engineering

Ian Howell, Hunter Mathews, Illya Starikov, William Thurman, Zachary Wileman (*Server Team #1*)

April 1st, 2017

Missouri University of Science and Technology

The Database

The Server

Server Demo

In Closing

Spam Illya with all questions, comments, and insults.

 [@IllyaStarikov](#)

 starikov@mst.com

Special thanks to our awesome team.

 Ian Howell

 Hunter Mathews

 Illya Starikov

 William Thurman

 Zachary Wileman

Camelot Final Summary

A summary of the Camelot server. Created and maintained
by Ian Howell, Hunter Mathews, Illya Starikov, William
Thurman, Zachary Wileman. *Server Team #1*

Last Revised: *May 5, 2017*
ILLYA STARIKOV

Michael Gosnell
Immediate Supervisor
mrghx4@mst.edu

Status: **Approved**
Version: **1.0**

For our project, we decided to implement an asynchronous server written in Python3 and uses a PostgreSQL database to store user and channel information. The Python module `pytest` was also used for writing unit tests so that we may try to have full code coverage on the project and “bring to light” any bugs that may exist in our project in terms of the Server and Database class files.

Some of the main accomplishments that were made on this project include: designing a database which was used for storing the user and channel information, creating a class file for both the server and database where the server file was dedicated to server functions (most functions mainly used by clients) and the database file was dedicated to interacting with the PostgreSQL database, writing unit tests to help better test our code so we could find any bugs/problems that may have existed, and created a multi-threaded asynchronous server for use with clients along with a client file used for testing purposes.

When designing the architecture, we decided to follow a Model View Controller paradigm. The controller was the server, the model was the database, and the view was the respective Client teams graphical user interface. In true MVC fashion,

- the server acts as the bridge between the client and the database
- the database stores all the data
- the client provides a view for the data

Our server runs on an asynchronous model. This means that we are able to service multiple clients independently of each other. This is accomplished by making use of Python3’s threading library. Each client is thrown onto its own thread, and each thread is thrown into a list of threads. When a client sends a message, we loop over each client (within the respective channel), sending the message to each. This is extended to all other actions performed by a client, such as logging in and deleting channels. When a client closes the connection to the server, it is removed from the client list.

The database we designed uses three tables, one for the user, one for the channel, and one for the user in channel relationship. The `USER` table stores the username and password of the user. The `CHANNEL` table stores the channel name and the admin of the channel. The `CHANNELS_JOINED` table stores which users are in which channels. Not only does the database allows us to query which users should receive a message, but it also allows us to perform other more complex operations such as notifying all users when a channel is deleted. An additional PostgreSQL file was also included that adds some initial channels to the database that all users are, by default, added to when they create their account.

While we waited for the client teams to finish their clients and begin testing the connection to our server, we created a handful of unit tests that would allow us to test certain functions that we would provide to the client teams to use. These functions included creating accounts, logging in, creating channels, deleting channels, deleting accounts, sending messages, along with many other things. For each different test, we covered all of our bases, both fail and success. Some examples include:

- If the user was not logged in, then what would the server tell the client
- If there was an invalid JSON file, then what would the server tell the client
- If the function was to fail, make sure the server handled the exception properly

As far as any deviations from the requirements document, none seemed to occur for our team, but one of our client teams did experience a change to their requirements document that affected us. Specifically, one of the client teams was originally wanting us to implement private messaging as a feature but the team decided to abandon that idea after they found out they weren't going to be able to implement this feature. So, due to that decision, the team decided to no longer pursue the implementation of the private messaging feature.

CS3200

Numerical Methods



1 Introduction

- The differential equation for newton's equation is $\frac{dv}{dt} = g - \frac{cv}{m}$, where cv is the drag coefficient $\times$ velocity.

$$\frac{dv}{dt} = g - \frac{cv}{m} \tag{1}$$

$$dv = (g - \frac{cv}{m})dt \tag{2}$$

$$\int \frac{1}{g - \frac{cv}{m}} dv = \int dt \tag{3}$$

$$\lg(g - \frac{cv}{m}) = t + c \tag{4}$$

$$v = mg/c(1 - e^{c/mt}) \tag{5}$$

2 Error Tolerance

- There will be 15% extra credit.
- Two important terms, accuracy & precision.
 - Accuracy refers to how close the computed value is to the true value.
 - Precision means how close the values are together.
- Because we want everything relative, we calculate with two different errors.
 - Absolute error = $e_a = |\text{True value} - \text{Approximation}|$
 - Relative error = $\epsilon_t = \frac{\text{True value} - \text{Approximation}}{\text{true}} \times 100$
- These equations can be used for precision as well.
 - $e_a = |\text{previous} - \text{current}|$
 - $\epsilon_t = \frac{\text{previous} - \text{current}}{\text{current}} \times 100$
- Stopping point is when it's below a certain threshold.
- If true value is 0, shift over by 1.
- For a result to be accurate or precise, should be $< 0.5 \times 10^{2-n}$, where n is the number of significant digits.
- the function of e^x can be represented as $e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!}$
 - for $x = 1$, $e^1 = 2.71828$
 - for $x = -1$, $a = 1 - 1 = 0$

4 Roots of Equations

- The bisection method is one way to find the root.
- There are two requirements.
 1. $f(x)$ is continuous on $[a, b]$
 2. $f(a) \times f(b) = -(\text{something})$

Numerical Methods Crib Sheet

Illya Starikov

Integration Equations

Note that h usually refers to $(b - a)$. Also, Trapezoidal needs 2 points, Simpson's 1/3 uses 3, Simpson's 3/8 uses 4 and Boole's 5. Also note that for something like $[0, 4]$, $h = 4, n = 1, h = 2, n = 2, h = 1, n = 4$

Trapezoidal $I = \frac{h}{2} [f(x_0) + 2 \sum_{i=1}^n f(x_n)]$
Richardson $I \approx \frac{4}{3}I(h_2) - \frac{1}{3}I(h_1)$
 $\approx \frac{4}{3}I(\text{current cell}) - \frac{1}{3}I(\text{previous cell})$

Romberg $I_{j,k} = \frac{4^{k-1}I_{j+1,k-1} - I_{j,k-1}}{4^{k-1} - 1}$
 $I_{j,k} = \frac{4^{\text{cell your on}} - 1}{4^{k-1} - 1} I_{\text{cell your on}} - I_{\text{cell one up}}$

Simpson 1/3 $\frac{h}{6}(f(a) + 4f(a + h) + f(b))$
Simpson 3/8 $\frac{3h}{8}(f(a + h) + 3f(a + h) + 3f(a + 2h) + f(b))$

Differentiation Equations

Forward Finite-Divided

$$\frac{d}{dx} = \frac{f(x + h) - f(x)}{h} \quad O(h)$$

Backward Finite-Divided

$$\frac{d}{dx} = \frac{f(x) - f(x - h)}{h} \quad O(h)$$

Central Difference

$$\frac{d}{dx} = \frac{f(x + h) - f(x - h)}{2h} \quad O(h^2)$$

$$\frac{d^2}{dx^2} = \frac{f(x + h) - 2f(x) + f(x - h)}{h^2} \quad O(h^4)$$

Richardson Extrapolation

$$I = I(h) + \mathcal{E}(h)$$

$$I = I(h_1) + \mathcal{E}(h_1) = I(h_2) + \mathcal{E}(h_2)$$

$$\mathcal{E} \approx -\frac{b-a}{2} h^2 \bar{f}''$$

$$\frac{\mathcal{E}(h_1)}{\mathcal{E}(h_2)} \approx -\frac{\frac{b-a}{2} h_1^2 \bar{f}''}{\frac{b-a}{2} h_2^2 \bar{f}''} \approx \frac{h_1^2}{h_2^2}$$

$$\mathcal{E}(h_1) \approx \mathcal{E}(h_2) \left(\frac{h_1}{h_2} \right)^2$$

$$I \approx I(h_1) + \mathcal{E}(h_2) \left(\frac{h_1}{h_2} \right)^2 \approx I(h_2) + \mathcal{E}(h_2)$$

$$\mathcal{E}(h_2) \approx \frac{I(h_2) - I(h_1)}{(h_1/h_2)^2 - 1}$$

$$I \approx I(h_2) + \frac{I(h_2) - I(h_1)}{(h_1/h_2)^2 - 1}$$

$$I \approx \frac{4}{3} I(h_2) - \frac{1}{3} I(h_1)$$

Differential Equations

Midpoint Method

Note that $y(a) = b \implies x_i = a, y_i = b$.

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_1 + h/2, y_i + k_1 h/2)$$

$$y_{i+1} = y_i + k_2 \cdot h$$

Heun Method

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + h, y_i + k_1 h)$$

$$y_{i+1} = y_i + \frac{h \cdot (k_1 + k_2)}{2}$$

RK-3

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + h/2, y_i + k_1 h/2)$$

$$k_3 = f(x_i + h, y_i + (-k_1 + 2k_2)h)$$

$$y_{i+1} = y_i + h \cdot (k_i + 4k_2 + k_3)/6$$

RK-4

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_1 + h/2, y_i + k_1 h/2)$$

$$k_3 = f(x_i + h/2, y_i + k_2 h/2)$$

$$k_4 = f(x_i + h, y_i + k_3 h)$$

$$y_{i+i} = y_i + h(k_1 + 2k_2 + 2k_3 + k_3)/6$$

Numerical Methods Equations

Illya Starikov

July 27, 2025

1 Taylor Series

$$f(a) + f'(a)(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \frac{f^{(3)}(a)}{3!}(x - a)^3 + \dots \quad (1)$$

$$= \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x - a)^k. \quad (2)$$

2 Taylor Series of e^x

$$e^x = 1 + x + \frac{x^2}{2!} + \dots \quad (3)$$

$$\frac{1}{e^{-x}} = \frac{1}{1 - x + \frac{x^2}{2} + \dots} \quad (4)$$

3 False Position

$$r = b - \frac{(b - a) \cdot f(b)}{f(b) - f(a)} \quad (5)$$

4 Newton's Method

$$x_{n+1} = x_n + \frac{f(x_n)}{f'(x_n)} \quad (6)$$

5 Secant Lines

$$x_3 = x_1 - \frac{(x_2 - x_1) \cdot f(x_1)}{f(x_2) - f(x_1)} \quad (7)$$

6 Modified Secant Method

$$x_2 = x_1 - \frac{f(x_1)\Delta}{f(x_1 + \Delta) - f(x_1)} \quad (8)$$

Homework _num9

Illya Starikov

Due Date: December 1st, 2016

1 Simpson 1/3 Rule For $1 + 2x + 3x^2$

$$\begin{aligned}\int_0^4 1 + 2x + 3x^2 dx &\approx h/6(f(a) + 4f(a+h) + f(b)) \\ &= 4/6(1 + 4(1 + 4 + 12) + (1 + 8 + 48)) \\ &= 2/3(1 + 68 + 57) \\ &= 84\end{aligned}$$

2 Simpson's 3/8 Rule For x^3

$$\begin{aligned}\int_0^3 x^3 dx &\approx 3/8h(f(a) + 3f(a+h) + 3f(a+2h) + f(b)) \\ &= 3/8(0 + 1 + 8 + 27) \\ &= 3/8 \cdot 36 \\ &= 13.5\end{aligned}$$

3 Trapezoidal, Richardson and Romberg Method of x^4

We know the equation for the trapezoid rule to be

$$I = \frac{h}{2} \left[f(x_0) + 2 \sum_{i=1}^n f(x_i) \right]$$

From this, we obtain

$$\begin{aligned} I_4 &= \frac{4}{2} (f(0) + f(4)) = 512 \\ I_2 &= \frac{2}{2} (f(0) + 2f(2) + f(4)) = 288 \\ I_1 &= \frac{1}{2} (f(0) + 2f(2) + 2f(3) + f(4)) = 226 \end{aligned}$$

For Richardson, we calculate the terms as follows:

$$\begin{aligned} I &\approx \frac{4}{3}I(h_2) - \frac{1}{3}I(h_1) \\ I_2 &= \frac{4}{3}(288) - \frac{1}{3}(512) = \frac{640}{3} = 213.\bar{3} \\ I_1 &= \frac{4}{3}(226) - \frac{1}{3}(288) = \frac{616}{3} = 205.\bar{3} \end{aligned}$$

And for Romberg,

$$\begin{aligned} I_{j,k} &= \frac{4^{k-1}I_{j+1,k-1} - I_{j,k-1}}{4^{k-1}} \\ I_1 &= \frac{4^2(\frac{616}{3}) - \frac{640}{3}}{15} = \frac{1024}{5} = 204.8 \end{aligned}$$

We observe that the table looks like

h	trapazoidal	Richardson	Romberg
4	512		
2	288	213. $\bar{3}$	
1	226	205. $\bar{3}$	204.8

4 Derivation of Richardson Extrapolation

We recall that the value of integration I is equal to approximation + error, which we can eloquently write

$$I = I(h) + \mathcal{E}(h) \quad (1)$$

Where $\mathcal{E}$ represents our error and $I(h)$ represents our approximation. Supposing we have two different steps size (h_1 and h_2), we can rewrite equation (??) in the form

$$I = I(h_1) + \mathcal{E}(h_1) = I(h_2) + \mathcal{E}(h_2) \quad (2)$$

Now, let us expand $\mathcal{E}(h)$ from equation (??) and equation (??).

$$\mathcal{E} \approx -\frac{b-a}{2}h^2 \bar{f}'' \quad (3)$$

Where b and a are the upper and lower bounds, respectively, and $\bar{f}''$ is the average value of $\frac{d}{dx}f(x)$ (where $f(x)$ is the function we are integrating). Therefore, we can a ratio of the two errors

$$\frac{\mathcal{E}(h_1)}{\mathcal{E}(h_2)} \approx -\frac{\frac{b-a}{2}h_1^2 \bar{f}''}{\frac{b-a}{2}h_2^2 \bar{f}''} \approx \frac{h_1^2}{h_2^2}$$

Which can be algebraically manipulated to

$$\mathcal{E}(h_1) \approx \mathcal{E}(h_2) \left(\frac{h_1}{h_2} \right)^2 \quad (4)$$

Now we can substitute equation (??) back into equation (??) to obtain

$$I \approx I(h_1) + \mathcal{E}(h_2) \left(\frac{h_1}{h_2} \right)^2 \approx I(h_2) + \mathcal{E}(h_2)$$

Which can be solve for

$$\mathcal{E}(h_2) \approx \frac{I(h_2) - I(h_1)}{(h_1/h_2)^2 - 1}$$

Now that we have an estimate (hence the $\approx$) of the truncation error, We once plug this back into equation (??) to get

$$I \approx I(h_2) + \frac{I(h_2) - I(h_1)}{(h_1/h_2)^2 - 1}.$$

Because we know the interval to be halved, we can safely assume $h_2 = h_1/2$. Thus, our equation reduces to

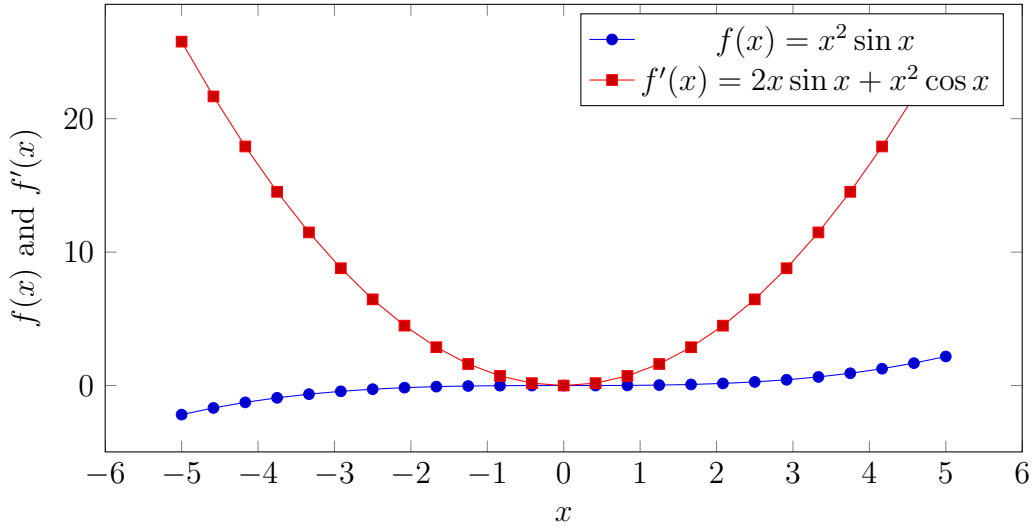
$$I \approx \frac{4}{3} I(h_2) - \frac{1}{3} I(h_1) \quad (5)$$

5 Derivative of $x^2 \sin x$

From the chain rule, we get

$$\frac{d}{dx} x^2 \sin x = 2x \sin x + x^2 \cos x$$

It can be plotted as follows



To compute the two derivatives for $h_1 = 0.2$ and $h_2 = 0.1$, we use the centered difference formulas. For h_1

$$\begin{aligned} \frac{d}{dx} &= \frac{f(x_{i+1}) - f(x_{i-1}))}{2h} \\ &= \frac{f(x+h) - f(x-h)}{2h} \\ &= \frac{f(x + 1/5) - f(x - 1/5)}{2/5} \\ &= \frac{(x + 1/5)^2 \sin(x + 1/5) - ((x - 1/5)^2 \sin(x - 1/5))}{2/5} \end{aligned}$$

Similarly, for h_2

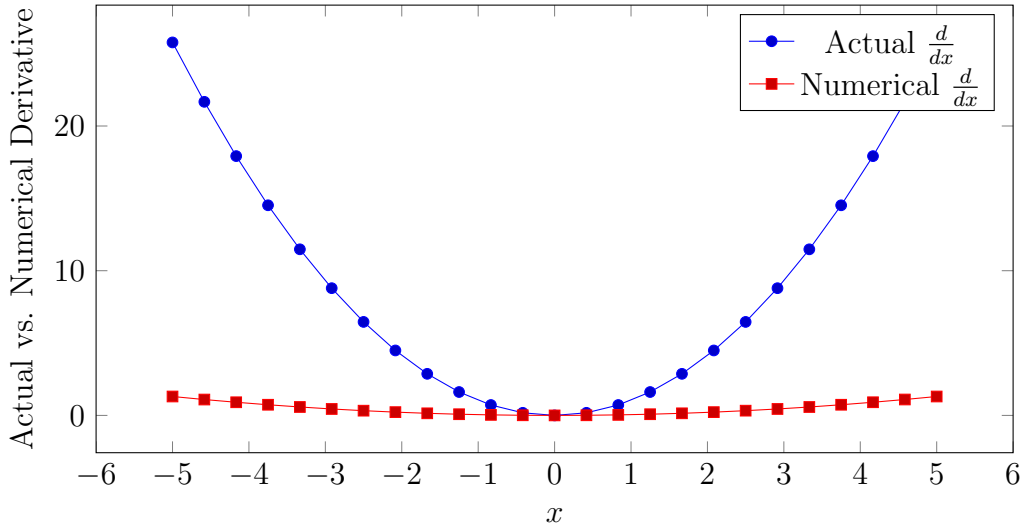
$$\frac{(x + 1/10)^2 \sin(x + 1/10) - ((x - 1/10)^2 \sin(x - 1/10))}{1/5}$$

We combine these for obtain the Richardson extrapolation formula,

$$D \approx 4/3 D(h_2) - 1/3 D(h_1) \quad (6)$$

$$\approx 4/3 \frac{(x + 1/10)^2 \sin(x + 1/10) - ((x - 1/10)^2 \sin(x - 1/10))}{1/5} \quad (7)$$

$$- 1/3 \frac{(x + 1/5)^2 \sin(x + 1/5) - ((x - 1/5)^2 \sin(x - 1/5))}{2/5} \quad (8)$$



6 Central Difference of x^4

6.1 First Order

From the power rule, we notice that

$$\frac{d}{dx} x^4 = 4x^3 \implies f'(1) = 4$$

Using centered difference formula,

$$\begin{aligned}
\frac{d}{dx} &= \frac{f(x_{i+1}) - f(x_{i-1}))}{2h} \\
&= \frac{(1 + 1/2)^4 - (1 - 1/2)^4}{(2^{1/2})} \\
&= 5
\end{aligned}$$

We notice the error to be $|\frac{4-5}{4}| = |- .25| \implies \mathbf{25\%}$

6.2 Second Order

From the power rule, we notice that

$$\frac{d^2}{dx^2} x^4 = 12x^2 \implies f'(1) = 12.$$

Using centered difference formula,

$$\begin{aligned}
\frac{d^2}{dx^2} &= \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1}))}{h^2} \\
&= \frac{(1 + 1/2)^4 - 2(1^4) + (1 - 1/2)^4}{(1/2)^2} \\
&= 25/2
\end{aligned}$$

We notice the error to be $|\frac{12-25/2}{12}| = |- .041\bar{6}| \implies \mathbf{4.16\%}$.

Homework_num10

Illya Starikov

Due Date: December 6th, 2016



CALENDAR CAMPUS MAP FIND PEOPLE A-Z INDEX

COMMUNITY FACULTY & STAFF ALUMNI & FRIENDS CURRENT STUDENTS FUTURE STUDENTS

Course Evaluation

Instructor Evaluation

Instructor: **Sabharwal, Chaman L**
Section: **COMP SCI 3200 - 1A - LEC**
Department: **Computer Science**
Course: **Intro Numerical Methods**

Your updates to this evaluation have been recorded. Thank you for your participation.

[Return to list of courses](#)

For any problems or corrections please contact the Solution Center at 573-341-HELP
or submit a [Help Request Ticket](#).

© Missouri University of Science and Technology | Rolla, MO 65409 | 573-341-4111 | 1-800-522-0938
Contact: [IT HelpDesk](#) · webmaster@mst.edu

CS3800

Operating Systems



1 Introduction

- Interrupts can come about by many sources
 - Keyboard, system clock, etc.
- Interrupts do not effect exceptions
- There is a bit in the PSW to determine if user/superuser mode

2 An Overview

- Horizontal — Generalized
- Vertical — Specialized

- Give an example of something that a process might do to initiate a voluntary context switch. Relate this answer to the Process State graph. When a process initiates an I/O Read/Write or event wait(). In the

- In dining philosophers problem, which scenario may lead to a deadlock situation? If the simulation code is written in such a way that each philosopher first acquires the right fork and then attempts to acquire the left fork, a deadlock may occur. Because it is possible that philosophers may enter a circular wait situation in which each one holds the fork on the right and tries to get the fork on the left (which will never happen).

- (such as `rand.r()`). However, `rand.r()` doesn't exist on some multithreaded operating systems (e.g. LINUX), therefore, you need to execute such calls inside a Critical Section to make sure that only one thread can call the function at any one time.

- Which of the following strategies are used for deadlock prevention (circle all that apply)? A. Processes request all of the resources they need at once. They either get it all or get nothing and wait

- If a desired page frame is not currently resident in RAM, A **Page Fault** occurs.
 - If a memory management system uses dynamic partitioning, **External** fragmentation may occur.
 - Since paging system uses **fixed size-sized** pages, **internal** fragmentation may occur.
 - Swapping out a piece of a process (i.e. pages of a process) just before that piece is needed is called **Thrashing**.
 - The least recently used (LRU) page replacement strategy is based on the principle of **temporal locality** as opposed to **spatial locality**.
 - The top four levels in the memory hierarchy, starting with the fastest, are: **Registers; cache memory; RAM; Disk**.
 - The two lowest layers in the 7-layer ISO Open Systems Interconnect (OSI) model are **Physical** and **Data Link** layers and their primary function is to provide **signaling technology and frame management**.
 - Two transport protocols, **Transmission Control Protocol (TCP)** and **User Datagram Protocol (UDP)**, are defined and handled at the Transport Layer.
 - **DMA (Direct Memory Access)** is a form of I/O in which a special module controls the exchange of data between main memory and an I/O device. During this I/O transfer, CPU is free to do other computation.
 - In which one of the following OSI layers Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are defined and implemented? a. Application b. Physical c. Transport d. Data Link e. Session c.
 - When we compare clusters with SMP (Symmetric Multiprocessors), which of the following are true (circle all that apply)? a. Clusters are easier to manage and configure b. Clusters take up less space and draw less power c. Clusters are better for incremental and absolute scalability d. Clusters are superior in terms of availability e. Clusters have superior price/performance c, d, e
 - Which of the following are among the direct goals of process scheduling algorithms (circle all that apply): a. improve response time b. minimize interrupts c. improve throughput d. minimize page faults e. improve turnaround time for jobs f. increase memory efficiency a, c, e
 - Which of the following features are specific to Real-Time OS? (circle all that apply) a. Small size b. Fast context switch c. Less user control d. Nondeterministic delays e. Fail-safe operation a, b, e
 - Which of the following malicious software need a host program to operate? (circle all that apply) a. Logic Bomb b. Worm c. Zombie (bots) d. Trojan Horse e. Virus a, d, e
 - Which of the following scheduling policies may cause starvation for certain jobs? (circle all that apply) a. First Come First Serve (FCFS) b. Feedback c. Round Robin d. Shortest Process Next (SPN) e. Shortest Remaining Time Next (SRT) b, d, e
 - Which of the following strategies is not used in a Disk Scheduling Algorithm? a) First in first out (FIFO) b) Last in first out (LIFO) c) Shortest service time first (SSTF) d) Longest service time first (LSTF) e) Back and forth over disk (SCAN) d
 - Which one of the following is not among the 7-layers defined for ISO Open Systems Interconnect (OSI) model? a) Application b) Routing c) Transport d) Data Link e) Physical b
 - Which one of the following is not among the set of events that may take place between the time a page fault occurs and the time the faulting process resumes execution? a) OS blocks the process and puts it into a wait queue. b) One of the processes in the ready queue is selected to run. c) A DMA is initiated to load the page from disk into main memory d) A page replacement strategy is used to find a page frame to load the new page e) Page table is updated to reflect the change. f) none of the above f
 - Which one of the following is not among the set of events that may take place between the time a page fault occurs and the time the faulting process resumes execution? a) OS blocks the process and puts it into a wait queue. b) One of the processes in the ready queue is selected to run. c) A DMA is initiated to load the page from disk into main memory d) The last page that the faulting process was executing is replaced with the newly loaded page. e) Page table is updated to reflect the change. f) none of the above d
 - What are the three popular strategies for allocating free memory blocks to processes in dynamic memory partitioning? Explain briefly how each strategy works
- First-fit:** chooses the first free block in the list that is large enough for the request; **Best-fit:** chooses the free block that is closest in size to the request; **Next-fit:** chooses the first free block that is large enough for the request and comes after the 'Last Allocated Block' in the list
- What interrupt is created when a desired page frame is not currently resident in RAM? **Page fault trap**
 - How does the hardware know that a desired page frame is not currently resident in RAM? **Valid bit**
 - What precisely does it mean if the dirty bit is set for a page frame? The page frame has been modified
 - What is good vs. bad program locality? **Good locality means that the process executes in clustered pages. Bad locality means that the process executes in scattered pages**
 - Explain when/how internal fragmentation may occur **When fixed-sized pages are used, the last page of a program may be partially filled. This is called internal fragmentation**
 - Explain when/how external fragmentation may occur **Segmentation system breaks up the memory space into variable-sized pieces. After a sequence of allocation and deallocations, free memory may get fragmented into small pieces. Even if the total size of free memory is large enough to satisfy large memory requests, a large request may not be met due to the lack of continuity between small fragments. This is called external fragmentation. Compaction is needed to put free blocks into one large memory block**
 - What is a global allocation scheme? **Global replacement allows a process to select a replacement frame from the set of all frames, even if that frame is currently allocated to some other process; one process can take a frame from another**
 - What is a working set model? **The working set model assumes that processes execute in localities. The working set is the set of pages in the current locality. Accordingly, each process should be allocated enough frames for its current working set**
 - Comparing global allocation vs. working set allocation, which would be more adversely affected by a program with bad locality? and WHY would that be true? **Working set allocation would be more adversely affected by a program with bad locality.**
- This is because the program with bad locality has poorly defined working sets and therefore, many page faults are likely to occur
- What is the "largest" program that could execute on a machine with a 24-bit virtual address? **2²⁴ byte**
 - What is the "largest" program that could execute on a machine with a 24-bit physical address? **Can't tell. Need to know the size of the virtual (logical) address**
 - The address contained in a TLB entry (PTE_i) is (physical—logical) **physical**
 - List at least 3 flags that are contained in a PTE **Valid bit, Reference bit, Dirty bit**
 - Define hit-ratio in a memory management context in a two-level memory (cache-RAM or RAM-Harddisk), the fraction of all memory accesses that are found in the master memory (i.e. the cache)
 - How does the kernel know where on disk the desired information is for a non-resident frame? **If valid bit=0, Page Table Entry should contain the Disk address**
 - Describe what demand paging means **The technique of only loading virtual pages into memory as they are accessed is known as demand paging. If the demand pages are not in memory, a page fault trap happens, and the operation system swaps them in**
 - Describe what prepaging means **Prepaging brings in more consecutive pages than needed. If a virtual page X causes a pagefault, then virtual page (X+1) is also brought in along with X. It is less overhead to bring in pages that reside contiguously on the disk**
 - Explain what the following C calls do both when the call is successful and when it is unsuccessful. 1. `socket(AF_INET, SOCK_STREAM, 0)` 2. `bind(sd, (struct sockaddr*)&server_addr, sizeof(server_addr))` 3. `socket(AF_INET, SOCK_DGRAM, 0)` 4. `accept(sd, (struct sockaddr*)&client_addr, &client_len)` 1. creates an internet stream (TCP) socket and returns the socket descriptor. If the call fails, it returns -1 2. Binds the definition of a socket (socket descriptor) to a port number. If the call fails, it returns -1 3. creates an internet datagram (UDP) socket and returns the socket descriptor. If the call fails, it returns -1 4. Blocks execution until a client connection is received. When that happens, it returns
- a descriptor for the connection. If the call fails, it returns -1
- What does an Internet Protocol do? 1. **Provides a naming scheme which uses a uniform format for host addresses** 2. **Provides a delivery mechanism by defining a standard packet format**
 - What are the possible goals that any scheduling policy might try to accomplish (list at least three)? **To improve response time, Turnaround time (TAT), Throughput, Processor Efficiency**
 - Which decisions are made by Long-term, Medium-term, and Short-term scheduling? **Be brief Long-term scheduling determines which programs are admitted to the system for processing and controls the degree of multiprogramming. Medium-term scheduling determines which programs will be resident. Part of the swapping function. Swapping-in decision is based on the need to manage the degree of multiprogramming. Short-term scheduling determines which program will be executed on CPU next. Known as the dispatcher Executes most frequently**
 - Name 3 things that are essential to launch a bot attack 1) attack software 2) a large number of vulnerable machines 3) locating these machines (scanning or fingerprinting)
 - Dennis Ritchie and Ken Thompson are generally credited with the invention of C/Unix.
 - Bill Gates and Paul Allen started Microsoft in 1975.
 - Steve Jobs and Steve Wozniak co-founded Apple. Steve Jobs then started NeXT, and was the CEO of Pixar.
 - MS/DOS was 90
 - What person Ed Roberts what company **MITS** built the 1st commercially available personal computer in 1975?
 - Gordon Moore is one of the Intel founders.
 - World's first personal computer, Altair 8800, was designed by Ed Roberts and was introduced in 1975
 - The first mass market PC company is **Apple**.
 - What corp may fairly take credit for inventions like the mouse, windows, pull-down menus etc.?
 - What did Steve Jobs see while visiting PARC that inspired him to build a different kind of computer? **GUI**
- What did Jobs see that he completely ignored? **object oriented programming and E-mail.**
 - What was the 1st computer that Jobs built based on this inspiration (that flopped)? **Lisa.**
 - What was the 2nd one that didn't flop? **Macintosh**
 - What product got Microsoft into the microcomputer software business? **BASIC language interpreter**
 - What lucky event got Microsoft into the operating system market? **Gary Kildall didn't eagerly pursue IBM when they requested a new OS. His wife and attorney would not sign a nondisclosure agreement. Bill Gates of Microsoft saw this as an opportunity and jumped in.**
 - What company purchased NeXT and their OS **NeXTStep?** What year? **Apple, in 1996**
 - What is a killer application? **Software that's so useful that people will buy computers just to run it.**
 - What was the killer app for the Apple II? **Visicalc**
 - What was the killer app for the IBM PC? **Lotus 1-2-3**
 - What was the killer app for the Apple Macintosh? **Wysiwyg - What you see is what you get -L Desktop Publishing**
 - Why didn't IBM create their own OS for their 1st PC? **wanted to manufacture and market it very fast; within one-year "...Once IBM decided to do a personal computer and to do it in a year - they couldn't really design anything, they just had to slap it together, so that's what they did ..."**
 - Who should have sold IBM their operating system for the 1st IBM PC? **Gary Kildall of Digital Research**
 - What was the one part of the 1st IBM PC that was proprietary (that Compaq had to later reverse engineer)? **ROM-BIOS**
 - Why did IBM decide to build the PC using open architecture? **To save time, instead of building a computer from scratch, IBM initially decided to buy PC components off the shelf and assemble them - in IBM terms, this was called an open architecture. IBM made some changes to this initial decision. What was the almost immediate result of IBM having made that decision? IBM had to buy the OS and other software from other companies as well.**
 - What was IBM's motivation for designing/building PS-2/OS-2? **IBM planned to steal the market from Gates with a brand new OS called OS/2.**

Homework #2

Illya Starikov

Due Date: April 6th, 2017

For this assignment, the tasks were to simulate a memory manager with three page replacement algorithms

1. Clock Page Replacement
2. Least Recently Used Page Replacement
3. First In First Out Page Replacement

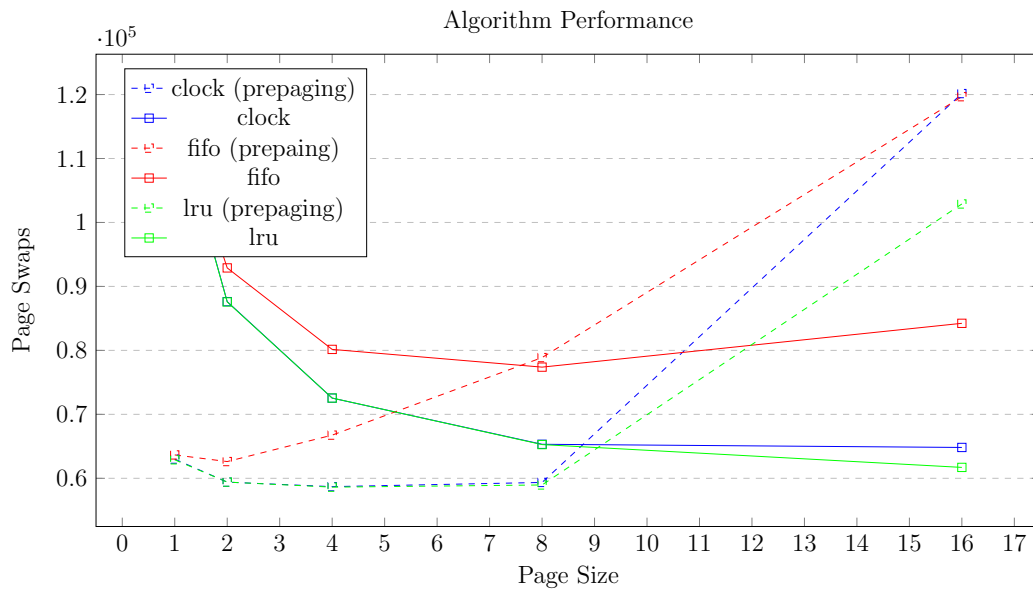


Figure 1 – Page Size vs. Page Swaps

This was accomplished by creating objects for the individual pages (eloquently named `Page`), programs (`Program`), and the memory simulator.

The individual algorithms were implemented as methods of the `MemorySimulation` class.

The results can be summarized by Figure 1. For the most part, the page swaps ranged from 60 000 to 120 000. The different page sizing affected the algorithms the same, *depending on if there was prepaging*. For prepaging, higher page size increased the page swaps. The inverse also holds, for on demand paging, the algorithms fared better for higher page size. Overall, *Least Recently Used* and *Clock Based Paging* with prepaging worked the best for page sizes of 2, 4, and 8. The worst was *Clock Based Paging* and *First In First Out* with prepaging. The median solution was *Least Recently Used* with prepaging.

In retrospect, one algorithm was no significantly more difficult to implement than another. Conceptually, the *Least Recently Used* algorithm was the most difficult, and even that one was not hard. *First In First Out* was the easiest to implement, but was the worst performance wise. *Clock Based Paging* was the hardest, but had some of the best performance.

Overall, the complexity of the algorithms wasn't the deciding factor on which one to use in day-to-day situations. With very high page swapping happening in an operating system, the deciding factor is the performance.

CS4096

Software Systems Development



Presentation #1

Senior Design (Comp Sci 4096)

Illya Starikov, Ian Howell, Deacon Seals, Michael Harrington,
Luke Parton, Eric Michalak, Adam Evans

Introduction

The Game-Plan

For Now...

We are delivering:

- Mobile game (iOS and Android) with server back-end
- Rectangular chunk of land as the board/play area
- Paint the land your team's color by running around
- Whichever team has the most area painted after a set time wins

The Game-Plan

For The Future...

We plan to deliver:

- End of the semester (hopefully)
- Have a public launch — with monetization options

Customer Related

Inteded Beneficiary College students (i.e., a campus club like Humans vs. Zombies)

Users As aforementioned.

Provides Users With An entertaining game and a fun way to get outside and exercise.

Design

The Human Interaction

1. Join a game (either “the public game” or custom game)
2. When game starts, run around in physical space
3. Tap to select and use items

Along with this, there will be leaderboards, trophies, achievements, and possibly challenges.

Design

The Human Interaction (Introduction)

The game will be similar to most human-based multiplayer games.

- A settings menu will allow users to customize the sound, paint transparency, and other relevant game mechanics.
- There are two games modes: a custom game and a public game.
 - Custom games lets a player change all parameters of the game.
 - Public games a pre-configured games that count towards leaderboards.

Whatever can be offloaded to the client side will be done on the client side, all synchronous game data will be pushed to the server.

Design

The Architecture

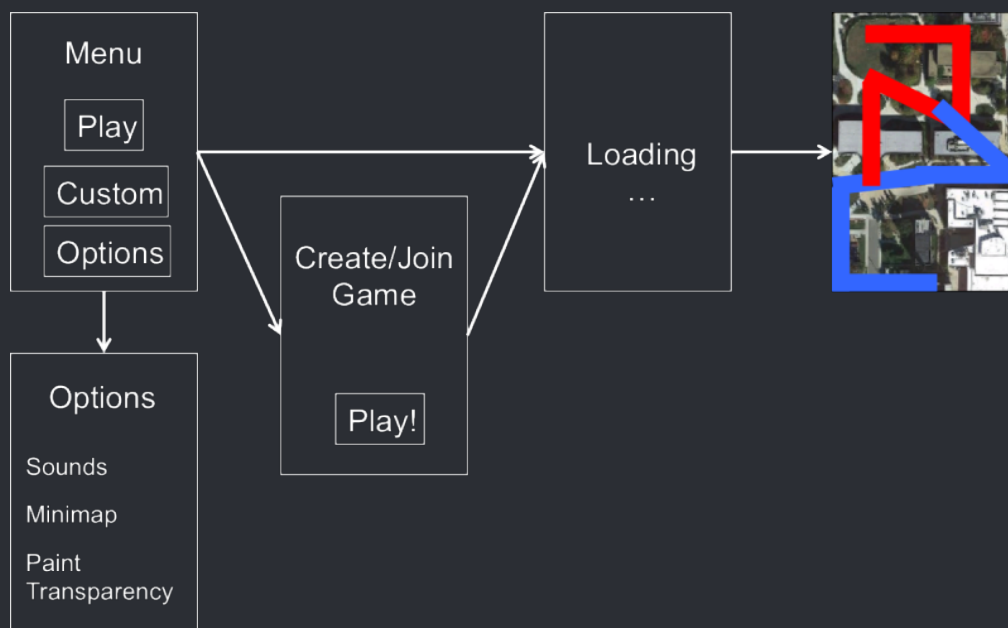


Figure: The Main Storyboard

Lessons Learned

Mistakes Were Made. Lessons Were Learned.

What worked:

- Breaking up into server, Android, and iOS teams — distributes workload well
- Swift instead of Objective-C for iOS — safer to develop in

What didn't work:

- Difficult to get the map functionality we want with `MKMapView` on iOS

In Closing

Spam Eric Michalak (Team Lead) or Illya Starikov (PR/Angel Investing) with all questions, comments, and insults.

 esxmv3@mst.edu

 starikov@mst.com

Special thanks to the awesome team.

-  Ian Howell
-  Deacon Seals
-  Michael Harrington
-  Luke Parton
-  Adam Evans

Backup Slides

Game Model

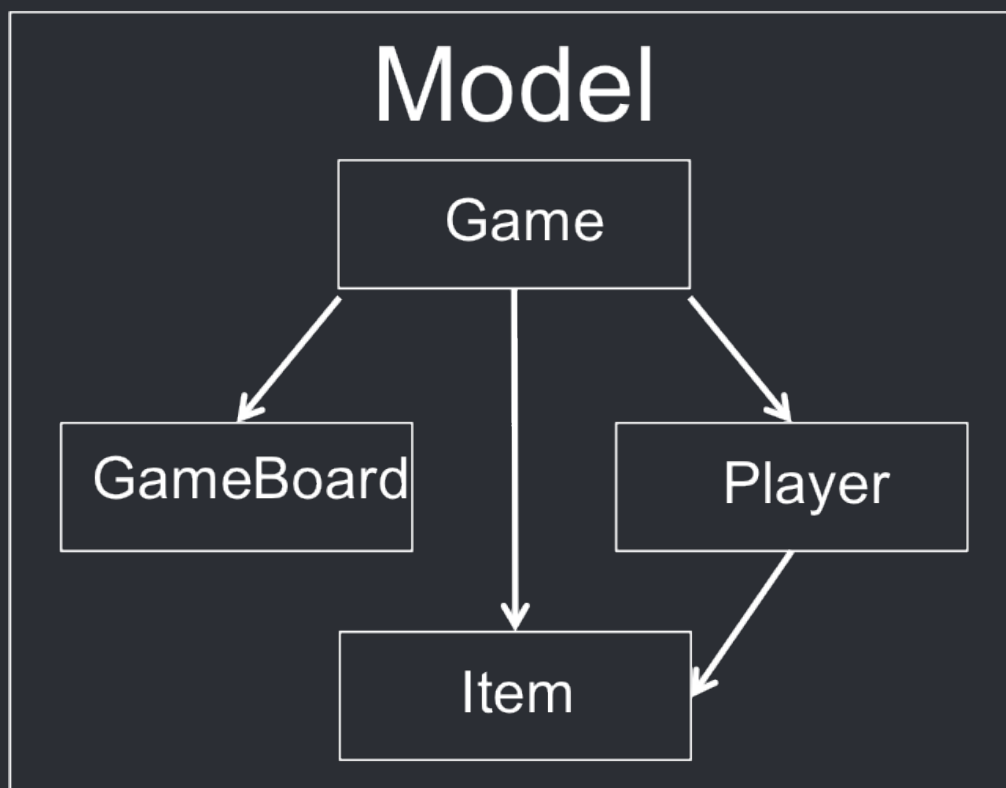


Figure: The Game Model

Splatoonio Communication Protocol

A RESTful guide to the communication of Splatoonio.
Created and maintained by Adam Evans, Michael
Harrington, Ian Howell, Eric Michalak, Luke Parton, Deacon
Seals, and Illya Starikov.

Last Revised: *December 5, 2017*
ILLYA STARIKOV

Ricardo Morales
ricardom@mst.edu
Senior Design

Status: **Draft**
Version: **1.0**

The server sends the current server time in every response.

- What are users for leaderboard x for page y of size z ?
- What are the achievements for user?
 - Send the `userID`.
 - Send back all the achievement information.
- Query for full board (send checksum, if not equal to server's version, get back full board).
- User x requests to join a private game (specific `gameID`).
- User x requests to join “the” public game (special `gameID`).
 - Send back new `playerID` for the game (privacy).
 - Send back `gameID`.
 - Send back full board data.
- Player queries if the game is ready.
- Player x moved to location lat/lon (player updates server and gets back board update).
 - Send pixels painted and timestamp when each pixel was painted.
 - Send player speed — anti-spoofing handling.
 - Send back board diff (list of pixels that changed since last update to this player).
 - Send back new items and which items were acquired.
- Player x acquired/moved over `itemID`.
 - Send `itemID`
 - Send back true if the requesting player successfully acquired the item.

Splatoonio Feature List

A design document detailing all possible features for Splatoonio. Created and maintained by Adam Evans, Michael Harrington, Ian Howell, Eric Michalak, Luke Parton, Deacon Seals, and Illya Starikov.

Last Revised: *December 5, 2017*
ADAM EVANS

Ricardo Morales
ricardom@mst.edu
Senior Design

Status: **Draft**
Version: **1.1**

Contents

1	Index	3
2	Server	4
2.1	Tasks	4
3	Mobile	5
3.1	Tasks	5
4	Game Mechanics	6
4.1	Games	6
4.2	Player	6
4.3	Items	6
4.4	Scoring System	6

1 Index

Game Board: The pixel map data structure that tracks enumerated pixel color

2 Server

Server Team: Eric Michalak, Deacon Seals, Michael Harrington

2.1 Tasks

- **Entire team (done):** Create client/server communication protocol
- **Deacon, Michael (done):** Research server frameworks
- Create games with unique IDs
 - Add players to games with unique IDs (added to smallest team)
 - Add players via group lobbies
- **Eric:** Manage account information in a database
 - **(done):** Leaderboards
 - **(done):** Achievements
 - Player Level
 - Statistics (items used, accuracy, players hit, total distance, total painted)
 - Unlockables (new items, etc)
- **Eric:** Maintain game state in a database
 - Bounds (lat/long)
 - Color of the *Game Board* map pixels
 - Active players and player information (held items, brush size, name)
 - Game clock
- **Deacon, Michael:** Play games
 - Handle game state changes (e.g. adding players, items spawning...)
 - **Deacons:** Update *Game Board*
 - **Michael:** Implement public REST API
- **Deacon:** Multiple concurrent games
- **Michael:** Generic API calls (winning team, percent controlled)

3 Mobile

iOS Team: Adam Evans, Illya Starikov

Android Team: Ian Howell, Luke Parton

3.1 Tasks

- Plays game
 - **(iOS, Android done):** Draws main display map in background - potentially fancy map (stretch goal)
 - **Ian (iOS done):** Paint board pixels slightly transparently over the map
 - **(iOS done):** Draw minimap displaying only the board (not map)
 - Display location of other players with arrows on the board edges or on the minimap (do we even want to do this anymore?)
 - **Ian:** Track the user's location/direction and move/rotate accordingly
 - **Luke:** Update game state from server
 - Game Features (see section 3)
- **(iOS done):** Stub out server calls
- **Adam:** Query server for leaderboards and achievements
- Query server for available game by user's location
- Query server to create/join a public game
- Query server to create/join a private game
 - Custom map and boundaries
 - Custom item spawns
- **(iOS done):** Create an options page

4 Game Mechanics

4.1 Games

- 3 teams per game: red/green/blue
- Game duration: 10/20/30 minutes
- Team with the most points wins

4.2 Player

- Paints ground under player with team's color
- brush size slowly increases with distance

4.3 Items

- Players collect items
 - Items spawn randomly?
 - Random Items every n seconds
 - Select an item every n seconds
- Items can damage players, resetting their brush size
- Bomb: paint a circle anywhere on the map by tapping the screen
- Green Shell: throws a shell which travels in a line painting everything under it

4.4 Scoring System

- Teams get points based on percentage. (10% = 10 points)
- Teams get additional points for controlling a marked region (CompSci Building = 10 additional points)

Ethical, Security, Legal, and Social Impact

CS 4096, Dr. Morales

Team Splatoonio*

July 27, 2025

War Paint is an augmented reality application that utilizes the player's current geographic location as a stylus to compete against other users. As part of gameplay this application collects user data and displays some for other users to see. While applications that give others your current location exist (i.e., SnapChat or Find My Friends) none of these display your location to anonymous users. In order to play War Paint the user must accept that their current location is being shared with other users; this creates an ethical conundrum. First, it would be illegal to allow children 13 years or younger to use the application without parent permission. There is a legal gray area in sharing a user's location from age 13 to 18, so to mitigate these ethical concerns, the application will likely be restricted to users 18 years or older.

The creators of War Paint also have an ethical responsibility to preserve the security of user data. War Paint will likely collect user data for the purpose of targeted advertisement; however, this data will never be sent to the War Paint server. In other words, we do not permanently store user data for longer than is necessary to play the game, and the data we do collect is all anonymous. It is important that this, as well as user passwords and profile information, remain secure and confidential. Passwords are currently stored on the server in plaintext due to time constraints, but before release, the server would be switched over to using a standard salted hashing function to protect user data. Clients would also be forced to send communication over HTTPS ensuring their connections were not visible in transit. In addition, most, if not all, of our server endpoints are also vulnerable to SQL injection, and would be fixed before releasing to mobile app stores. We would also like to have someone perform a penetration test of our system to help verify its security.

New users would be required to read and accept an End-User License Agreement stating that they are at least 18 years of age. It will clarify that the War Paint mobile app will collect user data and display the user's current location anonymously to other

*Adam Evans, Ilya Starikov, Ian Howell, Deacon Seals, Michael Harrington, Luke Parton, Eric Michalak

players while the game is in session. It will state that the creators and owners of War Paint are not responsible for damages or risks incurred by sharing the user's location.

The EULA will also clarify that users are responsible for their own physical safety while playing. War Paint is a physically active game. The most successful way to play is to maintain the highest possible speed for the duration of a game session. This would encourage especially competitive users to take risks in order to succeed. To prevent people from using cars or other motorized vehicles, a strict speed limit of 20 miles per hour will be imposed on the players. If this speed limit is exceeded, the user will cease painting the map until their speed falls back to acceptable levels. Repeatedly violating the speed limit will result in a temporary ban. In addition, there will be a report system that allows players to indicate which players are participating in dangerous behaviors while playing.

After witnessing the social impact Pokémon GO had on cities and parks, it is hard to estimate the potential impact of War Paint. On a smaller scale it could increase the number of people running on college campuses. Encouraging college students to exercise with their peers would be a positive social impact. On a larger scale this game could cause congestion in areas where it is the most popular. Matchmaking could keep track of the number of users in any specific area to prevent too many people from playing in too small a location.

This page is intentionally left blank.

Reflection Essay

Senior Design (Comp Sci 4096)

Illya Starikov

Due Date: Wednesday, December 13th

Throughout the course of Senior Design, I have learned a lot of useful information that will translate not only to the remainder of school, but to the actual development in the real world. Along the way of creating War Paint, I have learned several lessons that I hope to take with me upon graduating college.

For backstory, War Paint is a real-time, augmented reality game who's primary objective is to traverse as much of a terrain as possible. Specifically,

1. The player is boxed in a particular perimeter. Traversing outside said parameter is strictly forbidden (and will not count towards team score).
2. Players are split into even teams. As a player moves around the boxed perimeter, "paint" is placed down (colored with the team's primary color, red or blue).
3. Paining over another team's paint voids the other team's paint, and contributes only to who's team has the last layer of paint on the field.
4. At the end of a designated period of time, the team with the most paint on the board wins.

During the semester, I got to work with a team that had to deal with a whole stack. Our particular team had

- An Android Team
- An iOS Team
- A R&D Team (to focus on potential of hardware)
- A Server Team

Because my specialty was iOS Development, I took the role of working on the iOS application. Most of the software development process was simple: design the Model View Controllers (MVCs). The MVCs already had a hierarchy to them, so barely any architecture work had to be done.

The takeaway I gained from this class wasn't a particular tool or methodology; it was a particular mindset. Most of my group projects up to this point have placed me in a leadership role of some sorts. Because of my busy schedule this semester, I was not comfortable in that position. Because of this, I had to learn how to coordinate with others, be a team player, and put my faith into a different team lead.

My knowledge of the Software Development Process has not particularly changed up to this point. I have held several internships and have worked on massive projects before; most of my knowledge had been learned outside of the classroom.

The most important lesson I learned was how to not bite off more than I can chew. I had to usually take the minimal amount of work per week, for I had priorities in other classes and my job. This will greatly impact the way I handle work, where a work-life balance will be hard to maintain. I hope to keep this lesson in the back of my mind for future reference.

CS5200

Analysis Of Algorithms



Homework #1

Analysis of Algorithms

Illya Starikov

Due Date: September 1st, 2017

Question #1

- (a) 4.54 ± 0.05 billion years (https://en.wikipedia.org/wiki/Age_of_the_Earth)
- (b) 5 billion years (http://earthguide.ucsd.edu/virtualmuseum/ita/05_3.shtml)
- (c) 13.6 billion years $\pm$ 800 million years (<https://www.space.com/263-milky-age-narrowed.html>)
- (d) 13.82 billion years (http://www.slate.com/blogs/bad_astronomy/2013/03/21/age_of_the_universe_planck_results_show_universe_is_13_82_billion_years.html)
- (e) 7.79 billion years (<https://www.livescience.com/39775-how-long-can-earth-support-life.html>)
- (f) 3.25 billion years. Guaranteed existential risks, human prevention not possible.
 - Asteroid impact
 - Extraterrestrial invasion (i.e. , aliens taking over the world)
 - Geomagnetic reversal of the North and South pole

Prevention possible.

- Artificial super intelligence enslavement of the man race. Steps are unclear, besides keeping all experiments in a Faraday cage.
- Biotechnology engineering. A bio-engineered virus or disease that gets loose can cause a global pandemic. Steps to prevent are clear.
- Nuclear holocaust. Steps to prevent are clear.

Extinction prevention matters are to become a space-ferrying species, attempt some sort of peace treaty amongst all nations, and to try to prevent global pandemics. (https://en.wikipedia.org/wiki/Global_catastrophic_risk#Asteroid_impact)

- (g) 10-12 billion years. The sun will become a Red Giant, no longer being able to provide fuel. Will consume several planets along the way. (<https://www.forbes.com/sites/startswithabang/2017/01/27/how-our-solar-system-will-end-in-the-far-f#20ea714f4e5d>)

Eventually, about 5–7 billion years down the line, we'll run out of nuclear fuel in the Sun's core, which will cause our parent star to become a Red Giant, engulfing Mercury and Venus in the process. Due to the particulars of stellar evolution, the Earth/Moon system will probably be pushed outwards, and be spared the fiery fate of our inner neighbors.

- (h) Either the universe will stop contracting, reach an influx in the gravitational pull on all cosmic bodies, and come back to a singularity; or, heat death, where entropy reaches its max, and all energy is evenly dissipated throughout the universe, freezing everything. The first theory has 1-100 trillion years, the second has no estimate. (<http://www.bbc.com/earth/story/20150602-how-will-the-universe-end>)
- (i) 584.6 billion years. Between 0.585%–58.5%.

Question #2

- (a) A standard, 12pt L^AT_EX document can fit roughly 39 lines of text.

$$\frac{9 \times 10^8}{39} \approx 2.31 \times 10^8 \text{ sheets}$$

- (b) 1500 sheets is approximately \$14.99, before tax. (https://www.amazon.com/Georgia-Pacific-5dp/B00BB5DJU6/ref=sr_1_2?s=office-products&ie=UTF8&qid=1503934559&sr=1-2&keywords=printer+paper)

$$(2.31 \times 10^8 \text{ sheets}) \times \frac{\$14.99}{1500 \text{ sheets}} \approx \$2.31 \times 10^6$$

Assuming a standard filing cabinet can store roughly 15 reams, with 100 sheets in a ream,

$$2.31 \times 10^8 \text{ sheets} \times \frac{1 \text{ reams}}{100 \text{ sheets}} \times \frac{1 \text{ cabinet}}{15 \text{ reams}} = 154\,000$$

- (c) The monks expected the project to take *15,000 years* by hand, and *3 months* by modern technology. Assuming an average human lifespan of 79 years, approximately 50 years could be used writing.

$$\frac{15\,000 \text{ years}}{50 \text{ years}} = 300 \text{ people}$$

- (d) The time allocated to the computer was *3 months*. To print all 3 billion, it would have to print approximately 1120.07 names per second.
- (e) I enjoyed the story, and the first thing I did was a mental check to see if the numbers added up. Naturally, they did not, but that did not detract from the story.

Question #3

```

1 def alternating_difference(numbers):
2     if not numbers:
3         return 0
4
5     return numbers[0] - alternating_difference(numbers[1:])

```

Question #4

```

1 def f53q(n):
2     if n < 8:
3         return ValueError('Value less than 8')
4     elif n % 5 == 0:
5         return (0, n // 5)
6     elif n % 5 == 1:
7         return (2, (n - 5) // 5)
8     elif n % 5 == 2:
9         return (4, (n - 12) // 5)
10    elif n % 5 == 3:
11        return (1, n // 5)
12    else:
13        return (3, (n - 9) // 5)

```

Question #5

$$f(1) = 91$$

$$f(-6) = 91$$

$$f(200) = 190$$

$$f(27) = 91$$

A more appropriate, non-recursive function would be as follows:

```

1 def f(n):
2     if n < 101:
3         return 91
4     else:
5         return n - 10

```

This will always produce the same output. Let us write $f(n)$ as a piecewise-defined function.

$$f(x) = \begin{cases} n - 10 & 100 < n < \infty \\ f(f(n + 11)) & -\infty < n \leq 100 \end{cases}$$

We see that, for values less than 100, n will grow until it hits the first case. For values $90 \dots 200$, we see that n will oscillate until it reaches 101, upon which one final $f(x)$ will be applied.

A more simple explanation would be to insert $n - 10$ into $f(f(n + 11))$ to see that $f(x) = f(f(n + 1))$ until it reaches 101, 1 past the bounds, and gets $n - 10$ applied to it one more time.

Question #6

The function is copy and pasted from implementation.

```
1 def A(x, y):
2     if x == 0:
3         return y + 1
4     elif y == 0:
5         return A(x - 1, 1)
6     else:
7         return A(x - 1, A(x, y - 1))
```

The maximum value of my system (Late-2013 MacBook Pro) is roughly $A(3, 5)$. For $A(2, 2)$, there should be 42 438 function calls.

Question #7

```
1 def gcd2(a, b):
2     if a == 0:
3         return (b, 0, 1)
4     else:
5         g, s, t = gcd2(b % a, a)
6         return [g, t - (b // a) * s, s]
```

Question #8

```
1 # defined as list_ to not stomp out the typename _list_
2 def super_reverse(list_):
3     if len(list_) < 2:
4         return list_
5     else:
6         last_element = list_[-1]
7         first_element = list_[0]
8
```

```

9         # if is a list or a tuple, sort that as well
10        if isinstance(first_element, list):
11            first_element = super_reverse(first_element)
12        if isinstance(last_element, list):
13            last_element = super_reverse(last_element)
14
15        return [last_element] + super_reverse(list_[1:-1]) + [
            first_element]

```

Question #9

```

1 def anagram(string_):
2     if string_ == "":
3         return [""]
4
5     result = []
6
7     for partial_anagram in anagram(string_[1:]):
8         for position in range(len(partial_anagram) + 1) :
9             right_side = partial_anagram[position:]
10            left_side = partial_anagram[:position]
11            original_element = string_[0]
12
13            result += [left_side + original_element + right_side]
14
15    return result

```

Table 1: Results (units in milliseconds)

String Size	One Recursive Call	<i>n</i> Recursive Calls	Absolute Difference	Percent Difference
1	0.006	0.004	-0.002	-56.250%
2	0.009	0.009	-0.000	-2.703%
3	0.015	0.026	0.011	42.202%
4	0.036	0.104	0.068	65.367%
5	0.139	0.528	0.389	73.668%
6	0.672	3.124	2.452	78.486%
7	5.312	25.050	19.738	78.794%
8	39.671	204.850	165.179	80.634%
9	332.021	1936.445	1604.424	82.854%
10	3471.336	21813.828	18342.492	84.087%
11	43223.737	267635.574	224411.837	83.850%

Homework #2

Analysis of Algorithms

Illya Starikov

Due Date: September 11th, 2017

Question #1

We define $Q = \{q | q + n = n + k\}$, and $n \in \mathbb{Z}^+$. Given that $n + 0 = n = 0 + n$

$$\begin{aligned} q + (k + 1) &= (q + k) + 1 \\ &= (k + q) + 1 \\ &= k + (1 + k) \\ &= (k + 1) + m \\ \therefore q &\in Q \\ \therefore n + q &= q \end{aligned}$$

Question #2

Theorem 1. $\forall i \in \mathbb{Z}^+$,

$$\sum_{i=1}^n \frac{i^2}{(2i-1)(2i+1)} = \frac{n(n+1)}{2(2n+1)} \quad (1)$$

Proof. We will use the principle of recursion to solve this problem.

Step #1 Our problem states for an arbitrary integer i , summed to an arbitrary integer n , we have the summation (we denote by $f(x)$) and explicit forms ($g(x)$) described by Equation 1. We take the stopping value to be 0.

Step #2 Checking two values is trivial; take 1 and 2.

$$\sum_{i=1}^1 \frac{i^2}{(2i-1)(2i+1)} = \frac{1(1+1)}{2(2*1+1)} = \frac{1}{3}$$

$$\sum_{i=1}^n \frac{i^2}{(2i-1)(2i+1)} = \frac{2*(2+1)}{2(2*2+1)} = \frac{3}{5}$$

To check the stopping value, we check the 0th value. We clearly see that $f(x) = g(x) = 0$.

Step #3 If n in $\mathbb{Z}^+$ triggers a recursive call, then $n > 0$. The only value used in the call is $n - 1$, which is in $\mathbb{Z}$ and greater than or equal to 0, because it is an integer and $n1 \geq 0$ since $n > 0$.

Step #4 We use the integer n as the counter. When recursion is called the function is called with the value $n - 1$. The counter strictly decreases and the recursion halts.

Step #5 To prove recursion stops, we take the following:

$$\begin{aligned} f(n) &= \sum_{i=1}^n \frac{i^2}{(2i-1)(2i+1)} && \text{definition} \\ &= f(n-1) + \frac{n^2}{(2n-1)(2n+1)} && \text{recursive structure} \\ &= g(n-1) + \frac{n^2}{(2n-1)(2n+1)} && \text{by our assumption} \\ &= \frac{(n-1)((n-1)+1)}{2(2(n-1)+1)} + \frac{n^2}{(2n-1)(2n+1)} && \text{replacing } g(n) \text{ with definition} \\ &= \frac{n(n+1)}{2(2(n+1))} && \text{algebraic simplification} \\ &= g(n) \end{aligned}$$

Because $f(n) = g(n)$, the property is inherited recursively.

Step #6 Since Steps 1–5 have been verified, it follows from the Principle of Recursion that P holds for all values in $\mathbb{Z}^+$, i.e., $f(n) = g(n), \forall n \in \mathbb{Z}, n \geq 0$

□

Table 1: The results from the explicit form, the series form, and the difference for the first 40 values.

Sum Value	Explicit Values	Difference
0.333 333 333 333	0.333 333 333 333	0.000 000 000 000 000
0.600 000 000 000	0.600 000 000 000	0.000 000 000 000 000
0.857 142 857 143	0.857 142 857 143	0.000 000 000 000 000
1.111 111 111 111	1.111 111 111 111	0.000 000 000 000 000
1.363 636 363 636	1.363 636 363 636	0.000 000 000 000 000
1.615 384 615 385	1.615 384 615 385	0.000 000 000 000 000
1.866 666 666 667	1.866 666 666 667	0.000 000 000 000 000
2.117 647 058 824	2.117 647 058 824	0.000 000 000 000 000
2.368 421 052 632	2.368 421 052 632	0.000 000 000 000 000
2.619 047 619 048	2.619 047 619 048	0.000 000 000 000 000
2.869 565 217 391	2.869 565 217 391	0.000 000 000 000 000
3.120 000 000 000	3.120 000 000 000	0.000 000 000 000 001
3.370 370 370 370	3.370 370 370 370	0.000 000 000 000 001
3.620 689 655 172	3.620 689 655 172	0.000 000 000 000 001
3.870 967 741 935	3.870 967 741 935	0.000 000 000 000 001
4.121 212 121 212	4.121 212 121 212	0.000 000 000 000 001
4.371 428 571 429	4.371 428 571 429	0.000 000 000 000 001
4.621 621 621 622	4.621 621 621 622	0.000 000 000 000 002
4.871 794 871 795	4.871 794 871 795	0.000 000 000 000 001
5.121 951 219 512	5.121 951 219 512	0.000 000 000 000 002
5.372 093 023 256	5.372 093 023 256	0.000 000 000 000 002
5.622 222 222 222	5.622 222 222 222	0.000 000 000 000 002
5.872 340 425 532	5.872 340 425 532	0.000 000 000 000 003
6.122 448 979 592	6.122 448 979 592	0.000 000 000 000 003
6.372 549 019 608	6.372 549 019 608	0.000 000 000 000 003
6.622 641 509 434	6.622 641 509 434	0.000 000 000 000 004
6.872 727 272 727	6.872 727 272 727	0.000 000 000 000 004
7.122 807 017 544	7.122 807 017 544	0.000 000 000 000 004
7.372 881 355 932	7.372 881 355 932	0.000 000 000 000 004
7.622 950 819 672	7.622 950 819 672	0.000 000 000 000 004
7.873 015 873 016	7.873 015 873 016	0.000 000 000 000 003
8.123 076 923 077	8.123 076 923 077	0.000 000 000 000 004
8.373 134 328 358	8.373 134 328 358	0.000 000 000 000 004
8.623 188 405 797	8.623 188 405 797	0.000 000 000 000 004
8.873 239 436 620	8.873 239 436 620	0.000 000 000 000 004
9.123 287 671 233	9.123 287 671 233	0.000 000 000 000 004
9.373 333 333 333	9.373 333 333 333	0.000 000 000 000 004
9.623 376 623 377	9.623 376 623 377	0.000 000 000 000 004
9.873 417 721 519	9.873 417 721 519	0.000 000 000 000 004

Question #3

Theorem 2. For an arbitrary $n \in \mathbb{Z}$, such that $n \geq 0$,

$$2^n \leq \text{fib}(n) \leq 2^{\frac{n}{2}}$$

Proof. *Step #1* Prove by induction that for $n > 0$

$$2^n \leq \text{fib}(n) \leq 2^{\frac{n}{2}}$$

Step #2 Check the base case and two other values,

$$2^0 \leq \text{fib}(0) \leq 2^0, 2^1 \leq \text{fib}(1) \leq 2^{\frac{1}{2}}, 2^2 \leq \text{fib}(2) \leq 2^1$$

Step #3 To prove the recursive case, we take the following.

$$\begin{aligned} 2^n &\leq \text{fib}(n) \leq 2^{\frac{n}{2}} \\ 2^{n-1} &\leq \text{fib}(n) + \text{fib}(n-1) \leq 2^{\frac{n-1}{2}} \\ \frac{2^n}{2} &\leq \text{fib}(n) + \text{fib}(n-1) \leq \frac{2^{\frac{n}{2}}}{\sqrt{2}} \end{aligned}$$

Because we know $\frac{2^n}{2} \leq 2^n$ and $\frac{2^{\frac{n}{2}}}{\sqrt{2}} \leq 2^{\frac{n}{2}}$, we know that $f(n)$ must be bounded by those function.

Step #4 Since the hypotheses of Simple Induction are true, the conclusion follows, namely, for all natural numbers, $f(n) \bmod 133 = 0$.

□

Question #4

Upon inspection, it's quite apparent that the series is the sum of all natural numbers, with every other number being negative. More formally,

$$\sum_{q=1}^n (-1)^{q-1} q^2 = (-1)^{n-1} \frac{n * (n+1)}{2}$$

The results can be summarized below.

Table 2: The results from the explicit form, the series form, and the difference for the first 40 values.

Sum Value	Explicit Values	Sum of Numbers
1.0000000000	1.0000000000	1.0000000000000000
-3.0000000000	-3.0000000000	3.0000000000000000
6.0000000000	6.0000000000	6.0000000000000000
-10.0000000000	-10.0000000000	10.0000000000000000
15.0000000000	15.0000000000	15.0000000000000000
-21.0000000000	-21.0000000000	21.0000000000000000
28.0000000000	28.0000000000	28.0000000000000000
-36.0000000000	-36.0000000000	36.0000000000000000
45.0000000000	45.0000000000	45.0000000000000000
-55.0000000000	-55.0000000000	55.0000000000000000
66.0000000000	66.0000000000	66.0000000000000000
-78.0000000000	-78.0000000000	78.0000000000000000
91.0000000000	91.0000000000	91.0000000000000000
-105.0000000000	-105.0000000000	105.0000000000000000
120.0000000000	120.0000000000	120.0000000000000000
-136.0000000000	-136.0000000000	136.0000000000000000
153.0000000000	153.0000000000	153.0000000000000000
-171.0000000000	-171.0000000000	171.0000000000000000
190.0000000000	190.0000000000	190.0000000000000000
-210.0000000000	-210.0000000000	210.0000000000000000
231.0000000000	231.0000000000	231.0000000000000000
-253.0000000000	-253.0000000000	253.0000000000000000
276.0000000000	276.0000000000	276.0000000000000000
-300.0000000000	-300.0000000000	300.0000000000000000
325.0000000000	325.0000000000	325.0000000000000000
-351.0000000000	-351.0000000000	351.0000000000000000
378.0000000000	378.0000000000	378.0000000000000000
-406.0000000000	-406.0000000000	406.0000000000000000
435.0000000000	435.0000000000	435.0000000000000000
-465.0000000000	-465.0000000000	465.0000000000000000
496.0000000000	496.0000000000	496.0000000000000000
-528.0000000000	-528.0000000000	528.0000000000000000
561.0000000000	561.0000000000	561.0000000000000000
-595.0000000000	-595.0000000000	595.0000000000000000
630.0000000000	630.0000000000	630.0000000000000000
-666.0000000000	-666.0000000000	666.0000000000000000
703.0000000000	703.0000000000	703.0000000000000000
-741.0000000000	-741.0000000000	741.0000000000000000
780.0000000000	780.0000000000	780.0000000000000000

Question #5

Question #6

```
1 def depth(n):
2     if n < 2:
3         return 1
4     if n % 2 == 1:
5         return 1 + depth(3 * n + 1)
6     else:
7         return 1 + depth(n // 2)
8
9
10 def main():
11     for i in range(101):
12         print("%3.0f & %5.0f\\\\" % (i, depth(i)))
13
14
15 if __name__ == "__main__":
16     main()
```

Iteration	Depth				
0	1	34	14		
1	1	35	14	68	15
2	2	36	22	69	15
3	8	37	22	70	15
4	3	38	22	71	103
5	6	39	35	72	23
6	9	40	9	73	116
7	17	41	110	74	23
8	4	42	9	75	15
9	20	43	30	76	23
10	7	44	17	77	23
11	15	45	17	78	36
12	10	46	17	79	36
13	10	47	105	80	10
14	18	48	12	81	23
15	18	49	25	82	111
16	5	50	25	83	111
17	13	51	25	84	10
18	21	52	12	85	10
19	21	53	12	86	31
20	8	54	113	87	31
21	8	55	113	88	18
22	16	56	20	89	31
23	16	57	33	90	18
24	11	58	20	91	93
25	24	59	33	92	18
26	11	60	20	93	18
27	112	61	20	94	106
28	19	62	108	95	106
29	19	63	108	96	13
30	19	64	7	97	119
31	107	65	28	98	26
32	6	66	28	99	26
33	27	67	28	100	26

Question #7

Theorem 3. For an arbitrary $n \in \mathbb{Z}$, such that $n \geq 0$,

$$\sum_{i=1}^n = 1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

Proof. Step #1 Prove by induction that for $n = 0$

$$\sum_{i=1}^n i^2 = 1^2 + 2^2 + \dots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

We choose the summation form as $f(x)$ and the explicit form $g(x)$.

Step #2 Check the base case and two other values,

$$\begin{aligned}\sum_{i=1}^1 n^2 &= 1 = \frac{1(1+1)(2*1+1)}{6} \\ \sum_{i=1}^2 n^2 &= 5 = \frac{1(1+1)(2*1+1)}{6} \\ \sum_{i=1}^3 n^2 &= 14 = \frac{2*(2+1)(2*2+1)}{6}\end{aligned}$$

Step #3 $\forall n > 0$, prove that $P(n)$ is true if $P(n-1)$ is true,

$$\begin{aligned}f(x) &= \sum_{i=1}^n i^2 && \text{by definition} \\ &= f(x-1) + n^2 && \text{by recursion} \\ &= g(x-1) + n^2 && \text{by hypothesis} \\ &= \frac{(n-1)(n-1+1)(2(n+1)+1)}{6} + n^2 \\ &= \frac{n(n+1)(2n+1)}{6} && \text{algebraic simplification}\end{aligned}$$

Step #4 Since the hypotheses of Simple Induction are true, the conclusion follows, namely, for all natural numbers, $f(n) = g(n)$.

□

Question #8

Theorem 4. For an arbitrary $n \in \mathbb{Z}$, such that $n \geq 0$,

$$11^{n+2} + 12^{2n+1} \mod 133 = 0$$

Proof. *Step #1* Prove by induction that for $n > 0$

$$11^{n+2} + 12^{2n+1} \mod 133 = 0$$

Step #2 Check the base case and two other values,

$$11^3 + 12^3 \mod 133 = 0$$

$$11^4 + 12^5 \mod 133 = 0$$

Step #3 $\forall n > 0$, prove that $P(n)$ is true if $P(n - 1)$ is true,

$$\begin{aligned} f(x) &= 11^{n+2} + 12^{2n+1} \\ &= 11^{n+1} + 12^{2n-1} + 11^{n+2} + 12^{2n+1} \end{aligned}$$

Because we assume $11^{n+2} + 12^{2n+1}$ to be divisible, we can rewrite it to be $133a_1$, signifying it is evenly divisible by 133.

$$\begin{aligned} f(x) &= 11 * 11^n + 12^{-1} * 144^n + 133a_1 \\ &= a_2 11^n + a_3 144^n + 133a_1 \end{aligned}$$

We know the superposition of all of these numbers to be divisible by 133.

Step #4 Since the hypotheses of Simple Induction are true, the conclusion follows, namely, for all natural numbers, $f(n) \mod 133 = 0$.

□

Question #9

Let $F(0)$ denote the root of the tree T . If we start at $F(0)$, there must be an adjacent vertex to it, namely, v_1 . There are infinitely many vertexes by going through the tree (this must be the case, or the tree T would be finite).

We can repeat this process, for at any vertex v_n , there exists a vertex v_{n+1} . Because the tree is infinite, there are infinitely many v_{n+1} .

By induction, for any n , there exists a path of length L in T .

Question #10

```
1 class Binary_Search_Tree:
2     def __init__(self, data):
3         if data is None:
4             self.data = [0, [], []]
5         else:
6             self.data = data
```

```

7
8     def add_node(self, node):
9         if node.data[0] < self.data[0]:
10             self.__add_node_left(node)
11         else:
12             self.__add_node_right(node)
13
14     def __add_node_left(self, node):
15         if self.data[1] == []:
16             self.data[1] = node
17         else:
18             self.data[1].add_node(node)
19
20     def __add_node_right(self, node):
21         if self.data[2] == []:
22             self.data[2] = node
23         else:
24             self.data[2].add_node(node)
25
26     def leaf_count(self):
27         if self.data[1] == []:
28             if self.data[2] == []:
29                 return 1
30             else:
31                 return self.data[2].leaf_count()
32         elif self.data[2] == []:
33             return self.data[1].leaf_count()
34         else:
35             return self.data[1].leaf_count() + self.data[2].leaf_count()
36
37     def internal_count(self):
38         if self.data[1] == []:
39             if self.data[2] == []:
40                 return 0
41             elif self.data[2] == []:
42                 return 1 + self.data[1].internal_count()
43             else:
44                 return 1 + self.data[1].internal_count() + self.data[2].
45                 internal_count()
46
47 def main():
48     T = Binary_Search_Tree([
49         42,
50         Binary_Search_Tree([29, [], []]),
51         Binary_Search_Tree([51, [], []])
52     ])
53
54     print("Leaf: {0}, Internal: {1}".format(T.internal_count(), T.
55     leaf_count()))
56
57     T.add_node(Binary_Search_Tree([25, [], []]))

```



```

57     T.add_node(Binary_Search_Tree([22, [], []]))
58     T.add_node(Binary_Search_Tree([45, [], []]))
59     T.add_node(Binary_Search_Tree([8, [], []]))
60     T.add_node(Binary_Search_Tree([1000, [], []]))
61     T.add_node(Binary_Search_Tree([3, [], []]))
62     T.add_node(Binary_Search_Tree([1, [], []]))
63
64     print("Leaf: {0}, Internal: {1}".format(T.internal_count(), T.
        leaf_count()))
65
66
67 if __name__ == "__main__":
68     main()

```

From the results, we can very much see that Internal Nodes = Leaf Nodes − 1.

10.1 Proof

Proof. Take n to be the number of nodes, $I(T)$ the number of internal nodes and $L(T)$ to be the number of leaf nodes. We know the internal nodes to be the sum of previous internal nodes (i.e., $I(t) = I(t_1) + I(t_2) + 1$).

From this, we get

$$I(T) = I(t_1) + I(t_2) + 1 \quad (2)$$

$$I(t_1) = L(t_1) - 1 \quad (3)$$

$$I(t_2) = L(t_2) - 1 \quad (4)$$

$$= L(T) - 1 \quad (5)$$

□

Homework #3

Analysis of Algorithms

Illya Starikov

Due Date: September 19th, 2017

Question #1

```
1 def fn(n):
2     if n > 100:
3         return n - 10
4     return ((n - 101) % (c - 10)) + 91
```

Question #2

The following function tests random values up to in $\{-1\,000\,000, \dots, 1\,000\,000\}$. `is_gcd_fast` is a faster alternative because $\gcd \in \mathcal{O}\left(\log_{\frac{a}{a \bmod b}}(a \times b)\right)$ while $\gcd_{\text{fast}} \in \mathcal{O}(c)$.

```
1 from fractions import gcd
2 import random
3
4
5 def is_gcd(gcd_value, a, b):
6     return gcd_value == gcd(a, b)
7
8
9 def is_gcd_fast(gcd_value, a, b):
10    return a % gcd_value == b % gcd_value == 0
11
12 for i in range(1000):
13     a = random.randint(-1000000, 1000000)
14     b = random.randint(-1000000, 1000000)
15     g = gcd(a, b) if random.choice([True, False]) else random.randint
16         (-1000000, 1000000)
17     print(is_gcd_fast(g, a, b) == is_gcd(g, a, b))
```

Question #3

3.1 All Sums

```
1 # Warning, for all lists (with the exclusion of [1], [0] and []) this
   function does not stop
2 def generate_all_sums(l):
3     for element in l:
4         if element == 0:
5             return [0] + generate_all_sums(element[1:])
6         else:
7             return [element] + [generate_all_sums([element*element])]
8
9     return l
```

3.2 $\text{Sum}(T) \notin \{\text{all odds}\}$

Take Sum to be the function that maps all elements to all of their multiples.

Theorem 1. *There is not set T such that $\text{Sum}(T) = \{\text{all odd integers}\}$.*

We prove so by proof of contradiction.

Proof. Suppose not. That is, suppose that is a set T such that $\text{Sum}(T) = \{\text{all odd integers}\}$. Set T , at a minimum, must have a single element (for the empty set $\emptyset$ cannot produce any integers). We take an arbitrary element from this set T and name it q . There are two forms for q ($\exists m \in \mathbb{Z}$),

$$q = \begin{cases} 2m & \Leftrightarrow q \text{ is even} \\ 2m + 1 & \Leftrightarrow q \text{ is odd} \end{cases}$$

If q is even, there is already a contradiction. If q is odd, then we have the following contradiction. Supposing we sum q twice, we have the following form:

$$q = a(2m + 1) + b(2m + 1)$$

Because a and b are arbitrary, we take them to be 1.

$$\begin{aligned} q &= (2m + 1) + (2m + 1) \\ &= 4m + 2 \\ &= 2(2m + 1) \end{aligned}$$

Because we know the sum of integers and multiplication of integers to be integers, $2(2m + 1) \equiv 2n, \exists n \in \mathbb{Z}$, which has the form of an even integer.

This has lead us to a contradiction. Therefore, our hypothesis is false, concluding there is not set T such that $\text{Sum}(T) = \{\text{all odd integers}\}$.

□

3.3 Fast Algorithm For S

To get d from an arbitrary set S , we simply take the greatest common divisor from all the sets.

```
1 from functools import reduce
2
3 def gcd(numbers):
4     def gcd_single(a, b):
5         while b:
6             a, b = b, a % b
7         return a
8
9     return reduce(gcd_single, numbers)
```

3.4 $d \in \{540051690381, 5404079462298, 3485942644184\}$

547.

Question #4

Theorem 2. $\forall i \in \mathbb{Z}^+$,

$$\sum_{i=1}^n (-1)^{i-1} i^3 = -\frac{1}{8}(-1)^n (4n^3 + 6n^2 - 1) - \frac{1}{8}$$

Proof. *Step #1* We wish to prove that for all natural numbers $f(n) = g(n)$, where f and g are as defined as follows:

$$f(x) = \sum_{i=1}^n (-1)^{i-1} i^3$$
$$g(x) = -\frac{1}{8}(-1)^n (4n^3 + 6n^2 - 1) - \frac{1}{8}$$

Let D be the set of natural numbers (i.e., $D = \mathbb{Z}^+$). D includes the stopping value 1.

Step #2 Checking two values is trivial; take 2 and 3.

$$\sum_{i=1}^2 (-1)^{i-1} i^3 = -\frac{1}{8}(-1)^2 (4 * 2^3 + 6 * 2^2 - 1) - \frac{1}{8} = -7$$
$$\sum_{i=1}^3 (-1)^{i-1} i^3 = -\frac{1}{8}(-1)^3 (4 * 3^3 + 6 * 3^2 - 1) - \frac{1}{8} = 20$$

To check the stopping value, we check the first value. We clearly see that $f(x) = g(x) = 1$.

Step #3 If n in $\mathbb{Z}^+$ triggers a recursive call, then $n > 0$. The only value used in the call is $n - 1$, which is in $\mathbb{Z}$ and greater than or equal to 0, because it is an integer and $n - 1 \geq 0$ since $n > 0$.

Step #4 We use the integer n as the counter. When recursion is called the function is called with the value $n - 1$. The counter strictly decreases and the recursion halts.

Step #5 To prove recursion stops, we take the following:

$$f(x) = f(n - 1) + f(n) \tag{1}$$

$$= f(n - 1) + (-1)^{n-1}n^3 \tag{2}$$

$$= -\frac{1}{8}(-1)^{(n-1)}(4(n - 1)^3 + 6(n - 1)^2 - 1) - \frac{1}{8} + (-1)^{n-1}n^3 \tag{3}$$

$$= -\frac{1}{8}(-1)^n(4n^3 + 6n^2 - 1) - \frac{1}{8} \tag{4}$$

$$= g(x) \tag{5}$$

Because $f(n) = g(n)$, the property is inherited recursively.

Step #6 Since Steps 1–5 have been verified, it follows from the Principle of Recursion that P holds for all values in $\mathbb{Z}^+$, i.e., $f(n) = g(n), \forall n \in \mathbb{Z}^+$

□

Table 1: The results from the explicit form, the series form, and the difference for the first 40 values.

Sum Value	Explicit Values	Difference
1	1	0
-7	-7	0
20	20	0
-44	-44	0
81	81	0
-135	-135	0
208	208	0
-304	-304	0
425	425	0
-575	-575	0
756	756	0
-972	-972	0
1225	1225	0
-1519	-1519	0
1856	1856	0
-2240	-2240	0
2673	2673	0
-3159	-3159	0
3700	3700	0
-4300	-4300	0
4961	4961	0
-5687	-5687	0
6480	6480	0
-7344	-7344	0
8281	8281	0
-9295	-9295	0
10388	10388	0
-11564	-11564	0
12825	12825	0
-14175	-14175	0
15616	15616	0
-17152	-17152	0
18785	18785	0
-20519	-20519	0
22356	22356	0
-24300	-24300	0
26353	26353	0
-28519	-28519	0
30800	30800	0

```

1  def sum_solution(n):
2      if n <= 1:
3          return 1
4      else:
5          return (-1)**(n - 1) * n**3 + sum_solution(n - 1)
6
7
8  def explicit_solution(n):
9      return -int(
10         (1.0 / 8.0) *
11         ((-1)**n) *
12         (4 * n**3 + 6 * n**2 - 1) +
13         1.0 / 8.0)
14
15
16 def main():
17     for n in range(1, 40):
18         print(sum_solution(n), explicit_solution(n), sum_solution(n) -
19             explicit_solution(n))
20
21 if __name__ == "__main__":
22     main()

```

Homework #4

Analysis of Algorithms

Illya Starikov

Due Date: September 26th, 2017

Question #1

```
1 def sum_of_two_elements(S, x):
2     S = sorted(S)
3     i, j = 0, len(S) - 1
4     while i <= j:
5         if S[i] + S[j] == x:
6             return True
7         elif S[i] + S[j] < x:
8             i += 1
9         else:
10            j -= 1
11
12     return False
```


Table 1: Output for sum_of_two_elements

List	x	$n \lg n$ solution	n^2 solution
[219, 526, 87, 829, 59, 384, 849, 65, 463, 934]	449	True	True
[393, 983, 584, 160, 421, 652, 41, 719, 686, 181]	1338	True	True
[491, 154, 345, 508, 208, 50, 11, 183, 723, 994]	931	True	True
[798, 810, 263, 786, 177, 314, 211, 708, 300, 286]	1061	True	True
[262, 494, 533, 38, 105, 937, 383, 625, 733, 428]	645	True	True
[730, 371, 356, 591, 126, 416, 732, 84, 505, 165]	717	True	True
[469, 821, 559, 492, 479, 91, 621, 55, 550, 507]	605	True	True
[822, 407, 527, 858, 498, 360, 551, 532, 503, 274]	1054	True	True
[278, 265, 997, 628, 563, 536, 783, 817, 725, 124]	1380	True	True
[455, 745, 422, 274, 335, 781, 909, 867, 669, 681]	943	True	True
[209, 836, 717, 858, 446, 773, 507, 693, 907, 59]	268	True	True
[18, 639, 711, 738, 583, 69, 714, 503, 597, 280]	1452	True	True
[690, 867, 901, 558, 367, 927, 439, 590, 651, 447]	1459	True	True
[156, 129, 61, 363, 948, 347, 874, 914, 775, 73]	503	True	True
[781, 985, 385, 523, 753, 804, 740, 7, 155, 441]	596	True	True
[897, 799, 83, 402, 144, 820, 621, 22, 640, 660]	1420	True	True
[478, 364, 216, 907, 638, 576, 835, 487, 571, 883]	1051	True	True
[193, 116, 483, 17, 363, 276, 14, 534, 145, 636]	162	True	True
[626, 446, 185, 716, 514, 225, 953, 826, 758, 809]	1072	True	True
[268, 778, 934, 880, 347, 306, 90, 767, 626, 230]	1712	True	True
[170, 591, 380, 744, 868, 242, 736, 756, 45, 798]	624	False	False
[85, 354, 59, 185, 916, 42, 567, 532, 106, 285]	704	False	False
[938, 218, 584, 440, 574, 748, 450, 931, 955, 588]	909	False	False
[930, 337, 292, 491, 668, 486, 630, 320, 91, 797]	17	False	False
[277, 754, 116, 486, 75, 868, 788, 346, 326, 188]	847	False	False
[951, 37, 80, 72, 515, 45, 925, 533, 626, 767]	346	False	False
[529, 286, 483, 105, 283, 224, 461, 245, 447, 861]	482	False	False
[563, 577, 795, 911, 692, 462, 755, 311, 898, 268]	706	False	False
[798, 617, 27, 746, 149, 763, 546, 752, 692, 279]	816	False	False
[334, 339, 184, 870, 776, 716, 375, 752, 414, 453]	522	False	False
[683, 969, 687, 274, 870, 566, 139, 664, 699, 325]	17	False	False
[641, 411, 4, 735, 742, 205, 264, 548, 331, 617]	604	False	False
[876, 939, 653, 487, 148, 433, 807, 238, 848, 556]	872	False	False
[385, 21, 172, 598, 463, 721, 187, 670, 328, 917]	510	False	False
[3, 246, 181, 435, 937, 974, 817, 109, 841, 383]	518	False	False
[889, 542, 73, 455, 946, 307, 189, 988, 440, 349]	58	False	False
[383, 691, 490, 938, 233, 139, 818, 231, 825, 720]	170	False	False
[772, 543, 127, 538, 273, 508, 924, 793, 495, 150]	229	False	False
[311, 843, 307, 863, 305, 602, 696, 351, 910, 776]	42	False	False

Question #2

1. As follows, the inversions (i, j) are:

a) (1, 5)

b) (2, 5)

- c) (3, 4)
 - d) (3, 5)
 - e) (4, 5)
- (A[i], A[j]) are:

- a) (2, 1)
- b) (3, 5)
- c) (8, 6)
- d) (8, 1)
- e) (6, 1)

2. For a reverse sorted array, there would $\binom{n}{2} = \frac{n(n-1)}{2}$ total inversions.
3. Divide the array recursively into half and count number of inversions in sub-arrays (an $\lg n$ algorithm). To count all inversions, it takes n steps. $n - \text{operations} \times \lg n - \text{recursive steps}$. Combined, $\Theta(n \lg n)$ algorithm.

Question #3

We recall that the geometric series, and its derivatives, state that:

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \quad \text{and} \quad \sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2} \quad (1)$$

Next, we know that our sum is as follows:

$$\sum_{k=0}^{\infty} \frac{(k-1)}{2^k}$$

Breaking these up we have the following:

$$\sum_{k=0}^{\infty} k \left(\frac{1}{2}\right)^k - \sum_{k=0}^{\infty} \left(\frac{1}{2}\right)^k \quad (2)$$

Replacing Equation 2 with Equation 1, we get the following:

$$\begin{aligned} \sum_{k=0}^{\infty} k \left(\frac{1}{2}\right)^k - \sum_{k=0}^{\infty} \left(\frac{1}{2}\right)^k &= \frac{1/2}{(1-1/2)^2} - \frac{1}{1-1/2} \\ &= 2 - 2 \\ &= 0 \end{aligned}$$

Question #4

Because k^3 is monotonically increasing, we have the following bounds upper bound:

$$\begin{aligned}\sum_{k=1}^n k^3 &\leq \int_1^{n+1} x^3 dx \\ &= \left. \frac{x^4}{4} \right|_1^{n+1} \\ &= \frac{(n+1)^4 - 1}{4}\end{aligned}$$

For the lower bound,

$$\begin{aligned}\sum_{k=1}^n k^3 &\geq \int_0^n x^3 dx \\ &= \left. \frac{x^4}{4} \right|_0^n \\ &= \frac{(n)^4}{4}\end{aligned}$$

So, the bounds are as follows:

$$\frac{(n)^4}{4} \leq \sum_{k=1}^n k^3 \leq \frac{(n+1)^4 - 1}{4}$$

Question #5

We use integral approximations for all of the following.

Problem #5.1

$$\begin{aligned}\int_0^n x^r dx &\leq \sum_{k=1}^n k^r \leq \int_1^{n+1} x^r dx \\ \frac{n^{r+1}}{r+1} &\leq \sum_{k=1}^n k^r \leq \frac{(n+1)^{r+1} - 1}{r+1}\end{aligned}$$

Therefore, the bound is $\Theta(n^{r+1})$.

Problem #5.2

For the following, we take s for 1,

$$\begin{aligned} \int_0^n \lg^1 x \, dx &\leq \sum_{k=1}^n \lg^1 k \leq \int_1^{n+1} \lg^1 x \, dx \\ n(\lg(n) - 1) &\leq \sum_{k=1}^n \lg^1 k \leq (n+1)(\lg(n+1) - 1) + 1 \end{aligned}$$

Therefore, the bound is $\Theta(n \lg^1 n)$.

For the following, we take s for 2,

$$\begin{aligned} \int_0^n \lg^2 x \, dx &\leq \sum_{k=1}^n \lg^2 k \leq \int_1^{n+1} \lg^2 x \, dx \\ n((\lg(n) - 2)\lg(n) + 2) &\leq \sum_{k=1}^n \lg^2 k \leq (n+1)((\lg(n+1) - 2)\lg(n+1) + 2) - 2 \end{aligned}$$

Therefore, the bound is $\Theta(n \lg^2 n)$.

For the following, we take s for 3,

$$\int_0^n \lg^3 x \, dx \leq \sum_{k=1}^n \lg^3 k \leq \int_1^{n+1} \lg^3 x \, dx$$

From this, we get,

$$\begin{aligned} &n(\lg(n)((\lg(n) - 3)\lg(n) + 6) - 6) \\ &\leq \sum_{k=1}^n \lg^3 k \\ &\leq (n+1)(\lg(n+1)((\lg(n+1) - 3)\lg(n+1) + 6) - 6) + 6 \end{aligned}$$

Therefore, the bound is $\Theta(n \lg^3 n)$.

In general, the integral requires s integration by parts, multiplying $\lg n$ s times. So, for $\sum_{k=1}^n \lg^s k$, the bound would be $\Theta(n \lg^s n)$.

Problem #5.3

In general, we know that the lower-bound integral we have will have the form

$$\int_0^n x^r \lg^s x \, dx$$

And likewise for the upper bound. Using the same process as before (omitted due to tedious mathematics), we get the tight bound $\Theta(n^{r+1} \lg^s n)$.

Homework #5

Analysis of Algorithms

Illya Starikov

Due Date: October 4th, 2017

Question #1

Theorem 1. *If $L \geq 2$, then every binary tree with L leaves contains a subtree having between $\frac{L}{3}$ and $\frac{2L}{3}$ leaves, inclusive.*

Proof. Suppose not. That is, suppose that there exists a binary tree with more than 1 leaves with all subtrees containing leaves in the range $\frac{L}{3} > n > \frac{2L}{3}$. The root contains L leaves, and all bottommost leaves contain 1 leaves.

This implies that two subtrees with size less than $\frac{L}{3}$ (because our hypothesis states all leaves must in the range $\frac{L}{3} > n > \frac{2L}{3}$) combined to make a tree of size greater than $\frac{2L}{3}$. This is a contradiction

$$\frac{L}{3} + \frac{L}{3} \not> \frac{2L}{3}$$

Because we have reached a contradiction, our hypothesis does not hold. Therefore, binary tree with L leaves contains a subtree having between $\frac{L}{3}$ and $\frac{2L}{3}$ leaves, inclusive. $\square$

Question #2

Theorem 2. *For every binary tree T , let us associate a “weight” $w(q) = 2^{-\text{depth}(q)}$. For all leaves $q \in T$,*

$$\sum_q w(q) \leq 1$$

Proof. We prove so by induction.

(Base Case) For the root,

$$\sum_{x \in T} w(x) = 2^{-0} = 1$$

(Inductive Hypothesis) Suppose for any tree of n nodes, $\sum_{x \in T} w(x) \leq 1$.

(Inductive Step) Suppose the binary tree T_0 to have $n - 1$ nodes. We must show that $\sum_{x \in T_0} w(x) \leq 1$. Suppose we take the left subtree T_L or the right subtree T_R . By the inductive hypothesis, the sum of the weights are less than or equal to 1; however, since the depth of a node in T_0 is one greater than the depth of T_L or T_R , the respective weights of every leaf in T_0 is halved. Thus,

$$\sum_{x \in T_L} w(x) \leq 1/2 \quad \sum_{x \in T_R} w(x) \leq 1/2$$

Because both of these are bounded by $1/2$, their sum forms T_0 , which is bounded by 1. By the principle of mathematical induction, $\sum_q w(q) \leq 1$ for any tree. $\square$

Question #3

Theorem 3. *Any planar graph can be colored with six or fewer colors.*

Proof. We prove so by induction. We take n to be an arbitrary number of nodes in an arbitrary graph G .

(Base Case) For n vertices, where $1 \leq n \leq 6$, we can color the graph G with 6 or less colors (for every vertices maps to a single, unique color).

(Inductive Hypothesis) Suppose for $n - 1$ vertices, where $n > 1$, the graph G can be colored with 6 or less colors.

(Inductive Step) We now prove that G can be colored with n vertices can be colored with 6 or fewer colors. Recall all connected, simple planar graph contains a maximum of 5 degrees. Suppose that the any vertex $v \in G$ has this max degree 5.

We remove this vertex, name it v_0 , with all incident edges. We now have less than n vertices, and by our inductive hypothesis, this graph now can be colored with 6 or less colors. Adding this vertex back, we see that there are 5 incident vertexes, hence 5 colors. We use the 6th color for v_0 .

Hence, the inductive step holds. By the principle of mathematical induction, any planar graph can be colored with six or fewer colors. $\square$

Question #4

Theorem 4. *Any point x in a graph is a cut-point iff there exists two vertices in the graph, a and b , such that every path between a and b has to pass through x .*

Proof. Assume G to be a connected graph, and $\exists$ vertices $a, b \in G$, where every path $a \longleftrightarrow b$ path passes through x . Also assume there exists no path $a \longleftrightarrow b \in G - \{x\}$. Therefore, $G - \{x\}$ is disconnected, for a and b are in two different components of $G - \{x\}$. This proves that x is a cut-point. $\square$

Question #5

Euler Path A path that uses every edge of a graph exactly once.

Hamiltonian Cycle A path through a graph that starts and ends at the same vertex and includes every other vertex exactly once.

Yes, all **cyclic graphs** meet both these conditions.

Theorem 1. Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the non-negative numbers by the recurrence

$$T(n) = aT(n/b) + f(n) \quad (1)$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if

$$af(n/b) < cf(n)$$

for some constant $c < 1$ and sufficiently large n , then $T(n) = \Theta(f(n))$.

Question #1

Problem #1.1

From Equation 1, we have the following:

$$a = 2 \quad b = 4 \quad f(n) = 1$$

From this,

$$\begin{aligned} f(n) &= \mathcal{O}(n^{\log_b a - \epsilon}) \\ 1 &= \mathcal{O}(n^{\log_4 2 - \epsilon}) \\ &= \mathcal{O}(n^{1/2 - \epsilon}) \\ &= \mathcal{O}(n^{1/2 - 1/2}) \\ &= \mathcal{O}(1) \end{aligned}$$

Therefore, the tight asymptotic bound for the recurrence is

$$\begin{aligned} T(n) &\in \Theta(n^{\log_b a}) \\ &\in \Theta(n^{\log_4 2}) \\ &\in \Theta(n^{1/2}) \\ &\in \Theta(\sqrt{n}) \end{aligned}$$

Therefore, the bound is $\Theta(\sqrt{n})$.

Problem #1.2

From Equation 1, we have the following:

$$a = 2 \quad b = 4 \quad f(n) = \sqrt{n}$$

From this,

$$\sqrt{n} = \Theta(n^{\log_4 2})$$

Therefore, the tight asymptotic bound for the the recurrence is

$$T(n) \in \Theta(\sqrt{n} \lg n)$$

Problem #1.3

From Equation 1, we have the following:

$$a = 2 \quad b = 4 \quad f(n) = n$$

From this,

$$n = \Omega(n^{\log_4 2 + 1/2})$$

Therefore, the tight asymptotic bound for the the recurrence is

$$T(n) \in \Theta(n)$$

Problem #1.4

From Equation 1, we have the following:

$$a = 2 \quad b = 4 \quad f(n) = n^2$$

From this,

$$n^2 = \Omega(n^{\log_4 2 + 3/2})$$

Therefore, the tight asymptotic bound for the recurrence is

$$T(n) \in \Theta(n^2)$$

Question #2

Recall Strassen's algorithm belongs to the complexity class $\Theta(n^{\lg 7})$. By using the Case 1 of the Master Theorem, we must find a that solves the inequality $\log_4 a < \lg 7$. The integer that solves the equation $\log_4 a = \lg 7$ is 49; therefore,

$$a = 48$$

Question #3

Because the process is identical to Problem , the work is omitted; only answers are provided.

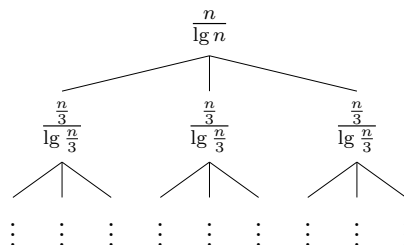
1. Master Theorem (*Case #3*), $T(n) \in \Theta(n^4)$
2. Master Theorem (*Case #3*), $T(n) \in \Theta(n)$
3. Master Theorem (*Case #2*), $T(n) \in \Theta(n^2 \lg n)$
4. Master Theorem (*Case #3*), $T(n) \in \Theta(n^2)$
5. Master Theorem (*Case #1*), $T(n) \in \Theta(n^{\log_2 7})$
6. Master Theorem (*Case #2*), $T(n) \in \Theta(\sqrt{n} \lg n)$
7. Because $(n - 2) \in \mathcal{O}(n)$, we can write an equivalent recurrence relation,

$$T(n) = T(n/1) + n^2$$

Because $\log_1 1$ is undefined, we cannot use the Master Theorem; but we do not need it. We have n^2 work to do n times. This implies that $T(n) \in \Theta(n^3)$.

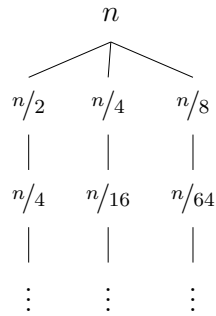
Question #4

1. Master Theorem (*Case #1*), $T(n) \in \Theta(n^{\log_3 4})$
2. Using a recursion tree,



We see that the $T(n) \in \Theta(n \lg \lg n)$.

3. Master Theorem (*Case #3*), $T(n) \in \Theta(n^{2.5})$.
4. We augment the equation to be $3T(n/3) + n/2$, where we can use Master Theorem (*Case #2*), $T(n) \in \Theta(n \lg n)$.
5. With similar reasoning as in Step #2, we use a recursion tree to get the results $T(n) \in \Theta(n \lg \lg n)$
6. We solve so via the substitution method. We use a recursion tree to get our initial guess.



Guessing $T(n) \in \Theta(n)$, we get the following:

$$\begin{aligned}
 T(n) &= T(n/2) + T(n/4) + T(n/8) + n \\
 &= c^{n/2} + c^{n/4} + c^{n/8} + n \\
 &= \frac{7}{8}n + n \\
 &\leq cn
 \end{aligned}$$

We see this inequality holds for $c \geq 8$. Therefore, $T(n) \in \Theta(n)$.

7. Using a recursion tree,



We see that $T(n) \in \Theta(n \lg \lg n)$.

8. Using a recursion tree,

$$\begin{array}{c} \lg n \\ | \\ \lg(n-1) \\ | \\ \lg(n-2) \\ | \\ \vdots \end{array}$$

We see that $T(n) \in \Theta(n \lg n)$.

9. Using a recursion tree,

$$\begin{array}{c} \frac{1}{\lg n} \\ | \\ \frac{1}{\lg(n-2)} \\ | \\ \frac{1}{\lg(n-4)} \\ | \\ \vdots \end{array}$$

We see that $T(n) \in \Theta(\lg \lg n)$.

10. We guess

$$T(n) \in \Theta(n \lg \lg n)$$

$$\begin{aligned} T(n) &= \sqrt{n} T(\sqrt{n}) + n \\ &= \sqrt{n} T(c\sqrt{n} \lg \lg \sqrt{n}) + n \\ &= c n \lg \lg n - c n + n \\ &\leq c n \lg \lg n \end{aligned}$$

We see this inequality for $c \in \mathbb{R}^+$. Therefore $T(n) \in \Theta(n \lg \lg n)$.

Question #5

Options c and e .

Question #6

1. We choose a root among the vertices, call it v_0 . If we choose the k th smallest element, the smaller subtree will have $i - 1$ vertices and the larger will have $n - i$ vertices. Summing over all possibilities, we get the following form for b_n :

$$\begin{aligned} b_n &= \sum_{k=1}^n b_{k-1} b_{n-k} \\ &= \sum_{k=0}^{n-1} b_k b_{n-k-1} \end{aligned}$$

- 2.

$$\begin{aligned} B(x) &= \sum_{n=0}^{\infty} b_n x^n \\ &= 1 + \sum_{n=1}^{\infty} b_n x^n \\ &= 1 + \sum_{n=1}^{\infty} \sum_{k=0}^{n-1} b_k b_{n-k-1} x^{n-k-1} \\ &= 1 + x \sum_{n=1}^{\infty} \sum_{k=0}^{n-1} b_k x^k b_{n-k-1} x^{n-k-1} \\ &= 1 + x \sum_{n=0}^{\infty} \sum_{k=0}^n b_k x^k b_{n-k} x^{n-k} \\ &= 1 + x B(x)^2 \end{aligned}$$

3. We use the Taylor expansion of $\sqrt{1 - 4x}$, we have the following

$$\begin{aligned}
B(x) &= \frac{1}{2x} \left(1 - \sum_{n=0}^{\infty} \frac{1}{1-2n} \binom{2n}{n} x^n \right) \\
&= -\frac{1}{2x} \sum_{n=1}^{\infty} \sum_{n=1}^{\infty} \frac{1}{1-2n} \binom{2n}{n} x^n \\
&= \frac{1}{2} \sum_{n=1}^{\infty} \frac{1}{2n-1} \binom{2n}{n} x^{n-1} \\
&= \frac{1}{2} \sum_{n=0}^{\infty} \frac{1}{2n+1} \binom{2n+2}{n+1} x^n
\end{aligned}$$

Therefore $b_n = \frac{1}{n+1} \binom{2n}{n}$.

4.

$$\begin{aligned}
b_n &= \frac{1}{n+1} \frac{(2n)!}{n!n!} \\
&\approx \frac{1}{n+1} \frac{\sqrt{4\pi n} (2n/e)^{2n}}{2\pi n (n/e)^{2n}} \\
&= \frac{1}{n+1} \frac{4^n}{\sqrt{\pi n}} \\
&= \left(\frac{1}{n} + \left(\frac{1}{n+1} - \frac{1}{n} \right) \right) \frac{4^n}{\sqrt{\pi n}} \\
&= \left(\frac{1}{n} - \frac{1}{n^2+n} \right) \frac{4^n}{\sqrt{\pi n}} \\
&= \frac{1}{n} \left(1 - \frac{1}{n+1} \right) \frac{4^n}{\sqrt{\pi n}} \\
&= \frac{4^n}{\sqrt{\pi n}^{3/2}} (1 + O(1/n))
\end{aligned}$$

Homework #7

Analysis of Algorithms

Illya Starikov

Due Date: October 20th 2017

```
1 class Node:
2     value = -1
3     left_child, right_child = None, None
4
5     def __init__(self, value):
6         self.value = value
7         self.left_child, self.right_child = None, None
8
9     @property
10    def number_of_children(self):
11        if self.left_child is not None and self.right_child is not None:
12            return 2
13        elif self.left_child is not None or self.right_child is not None:
14            return 1
15        else:
16            return 0
17
18    @property
19    def valid_children(self):
20        number_of_children = self.number_of_children
21
22        if number_of_children == 0:
23            return None
24        elif number_of_children == 1:
25            if self.left_child is not None:
26                return self.left_child
27            else:
28                return self.right_child
29        else:
30            return (self.left_child, self.right_child)
31
32    def __str__(self):
33        return str(self.value)
34
35
36 class BinarySearchTree:
```

```

37     root = -1
38
39     def __init__(self):
40         self.root = None
41
42     def insert(self, value):
43         if self.root is None:
44             self.root = Node(value)
45         else:
46             self.__insert_node(self.root, value)
47
48     def exists(self, value):
49         return False if self.__find_node(self.root, value) is None else
True
50
51     def delete(self, value):
52         self.root = self.__delete_node(self.root, value)
53
54     def print_tree(self):
55         if not self.root:
56             return
57
58         current_level = [self.root]
59
60         while current_level:
61
62             print(' '.join(str(node) for node in current_level))
63
64             next_level = list()
65             for n in current_level:
66                 if n.left_child:
67                     next_level.append(n.left_child)
68                 if n.right_child:
69                     next_level.append(n.right_child)
70             current_level = next_level
71
72     # MARK: Private Methods
73     def __insert_node(self, current_node, value):
74         # for the values <= `value`, we put on the left side of the tree
75         if value <= current_node.value:
76             if current_node.left_child is None:
77                 current_node.left_child = Node(value)
78             else:
79                 self.__insert_node(current_node.left_child, value)
80
81         # for values > `value`, we put on the right side of the tree
82     else:
83         if current_node.right_child is None:
84             current_node.right_child = Node(value)
85         else:
86             self.__insert_node(current_node.right_child, value)
87

```



```

88     def __find_node(self, current_node, value):
89         if current_node is None:
90             return None
91         elif value < current_node.value:
92             return self.__find_node(current_node.left_child, value)
93         elif value > current_node.value:
94             return self.__find_node(current_node.right_child, value)
95         else:
96             return current_node
97
98     def __delete_node(self, current_node, key):
99         if not current_node:
100             return current_node
101
102         if current_node.value > key:
103             current_node.left_child = self.__delete_node(current_node.
left_child, key)
104         elif current_node.value < key:
105             current_node.right_child = self.__delete_node(current_node.
right_child, key)
106         else:
107             if not current_node.left_child:
108                 right_child = current_node.right_child
109                 del current_node
110                 return right_child
111
112             elif not current_node.right_child:
113                 left_child = current_node.left_child
114                 del current_node
115                 return left_child
116
117             else:
118                 successor = current_node.right_child
119                 while successor.left_child:
120                     successor = successor.left_child
121
122                 current_node.value = successor.value
123                 current_node.right_child = self.__delete_node(current_node
.right_child, successor.value)
124
125         return current_node

```

Question #1

With the following elements,

42	2555	4830	7516
104	2718	4883	7604
321	2849	4961	7632
578	2954	5016	7706
600	2974	5119	7754
734	3063	5136	7989
929	3159	5246	8106
1004	3262	5253	8126
1120	3397	5502	8135
1128	3405	5520	8204
1149	3476	5550	8223
1213	3694	5644	8338
1323	3739	5675	8359
1347	3902	6082	8565
1512	4310	6306	8611
1522	4466	6425	8740
1730	4475	6562	8890
1831	4494	6570	8961
1853	4584	6653	9300
1886	4644	6712	9304
1899	4699	6940	9509
2040	4750	6966	9614
2082	4769	7164	9762
2164	4783	7324	9971
2253	4786	7499	9985

Problem #1.1

Determining if elements in the binary search tree are there. All these values should be true.

```

1 Element 3739 Found: True
2 Element 5550 Found: True
3 Element 600 Found: True
4 Element 3262 Found: True
5 Element 8611 Found: True
6 Element 4750 Found: True
7 Element 5246 Found: True
8 Element 6712 Found: True
9 Element 1323 Found: True
10 Element 1831 Found: True
11 Element 5520 Found: True
12 Element 3063 Found: True
13 Element 9300 Found: True
14 Element 1004 Found: True
15 Element 7164 Found: True
16 Element 4466 Found: True
17 Element 8890 Found: True

```

18	Element	8338	Found: True
19	Element	1853	Found: True
20	Element	8223	Found: True
21	Element	3159	Found: True
22	Element	4644	Found: True
23	Element	7706	Found: True
24	Element	8740	Found: True
25	Element	1120	Found: True
26	Element	2040	Found: True
27	Element	6653	Found: True
28	Element	578	Found: True
29	Element	9762	Found: True
30	Element	8565	Found: True
31	Element	7604	Found: True
32	Element	9985	Found: True
33	Element	4961	Found: True
34	Element	929	Found: True
35	Element	2082	Found: True
36	Element	5253	Found: True
37	Element	3405	Found: True
38	Element	7516	Found: True
39	Element	6306	Found: True
40	Element	8106	Found: True
41	Element	1347	Found: True
42	Element	4310	Found: True
43	Element	4584	Found: True
44	Element	2954	Found: True
45	Element	9614	Found: True
46	Element	2253	Found: True
47	Element	1522	Found: True
48	Element	8135	Found: True
49	Element	4475	Found: True
50	Element	1512	Found: True
51	Element	9304	Found: True
52	Element	6966	Found: True
53	Element	321	Found: True
54	Element	2974	Found: True
55	Element	6940	Found: True
56	Element	7754	Found: True
57	Element	1213	Found: True
58	Element	1149	Found: True
59	Element	8961	Found: True
60	Element	8359	Found: True
61	Element	4883	Found: True
62	Element	7499	Found: True
63	Element	4786	Found: True
64	Element	8126	Found: True
65	Element	42	Found: True
66	Element	4699	Found: True
67	Element	3902	Found: True
68	Element	1730	Found: True
69	Element	4494	Found: True

70	Element	5502	Found:	True
71	Element	104	Found:	True
72	Element	4830	Found:	True
73	Element	6082	Found:	True
74	Element	1899	Found:	True
75	Element	5136	Found:	True
76	Element	5644	Found:	True
77	Element	2849	Found:	True
78	Element	5675	Found:	True
79	Element	8204	Found:	True
80	Element	7632	Found:	True
81	Element	1886	Found:	True
82	Element	5119	Found:	True
83	Element	5016	Found:	True
84	Element	6562	Found:	True
85	Element	3397	Found:	True
86	Element	9971	Found:	True
87	Element	3694	Found:	True
88	Element	4783	Found:	True
89	Element	2718	Found:	True
90	Element	734	Found:	True
91	Element	7989	Found:	True
92	Element	7324	Found:	True
93	Element	1128	Found:	True
94	Element	6425	Found:	True
95	Element	4769	Found:	True
96	Element	2555	Found:	True
97	Element	3476	Found:	True
98	Element	9509	Found:	True
99	Element	2164	Found:	True
100	Element	6570	Found:	True

Problem #1.2

Determining if elements **not** in binary search tree are found. These values should be false.

1	Element	9170	Found:	False
2	Element	1059	Found:	False
3	Element	6807	Found:	False
4	Element	7990	Found:	False
5	Element	9177	Found:	False
6	Element	9933	Found:	False
7	Element	9290	Found:	False
8	Element	5479	Found:	False
9	Element	611	Found:	False
10	Element	5458	Found:	False
11	Element	1563	Found:	False
12	Element	5870	Found:	False
13	Element	3134	Found:	False
14	Element	178	Found:	False
15	Element	3890	Found:	False

16	Element	7225	Found: False
17	Element	9820	Found: False
18	Element	2292	Found: False
19	Element	5313	Found: False
20	Element	4489	Found: False
21	Element	3835	Found: False
22	Element	2543	Found: False
23	Element	5679	Found: False
24	Element	46	Found: False
25	Element	1812	Found: False
26	Element	4628	Found: False
27	Element	7310	Found: False
28	Element	6839	Found: False
29	Element	5411	Found: False
30	Element	2929	Found: False
31	Element	2351	Found: False
32	Element	9107	Found: False
33	Element	7498	Found: False
34	Element	5810	Found: False
35	Element	6692	Found: False
36	Element	3986	Found: False
37	Element	2941	Found: False
38	Element	9506	Found: False
39	Element	3485	Found: False
40	Element	1113	Found: False
41	Element	4268	Found: False
42	Element	9848	Found: False
43	Element	1525	Found: False
44	Element	7304	Found: False
45	Element	1792	Found: False
46	Element	2707	Found: False
47	Element	4111	Found: False
48	Element	7696	Found: False
49	Element	4897	Found: False
50	Element	120	Found: False
51	Element	9588	Found: False
52	Element	819	Found: False
53	Element	5893	Found: False
54	Element	9990	Found: False
55	Element	6444	Found: False
56	Element	2619	Found: False
57	Element	3091	Found: False
58	Element	8431	Found: False
59	Element	7965	Found: False
60	Element	4863	Found: False
61	Element	9778	Found: False
62	Element	9601	Found: False
63	Element	8084	Found: False
64	Element	7459	Found: False
65	Element	7299	Found: False
66	Element	1	Found: False
67	Element	6819	Found: False

```

68 Element 9085 Found: False
69 Element 5574 Found: False
70 Element 8748 Found: False
71 Element 8790 Found: False
72 Element 6042 Found: False
73 Element 5852 Found: False
74 Element 5341 Found: False
75 Element 7671 Found: False
76 Element 8456 Found: False
77 Element 1818 Found: False
78 Element 6918 Found: False
79 Element 9723 Found: False
80 Element 3980 Found: False
81 Element 6496 Found: False
82 Element 5121 Found: False
83 Element 2553 Found: False
84 Element 7316 Found: False
85 Element 1461 Found: False
86 Element 3896 Found: False
87 Element 9382 Found: False
88 Element 8610 Found: False
89 Element 5181 Found: False
90 Element 3305 Found: False
91 Element 549 Found: False
92 Element 7306 Found: False
93 Element 4612 Found: False
94 Element 9525 Found: False
95 Element 4847 Found: False
96 Element 6420 Found: False
97 Element 2643 Found: False
98 Element 4538 Found: False
99 Element 5934 Found: False
100 Element 7969 Found: False

```

Problem #1.3

Deleting entire tree. This output is ≈ 2000 , so please refer to output for entire tree.

```

1 Current Tree Before Deleting Element 4769
2 3739
3 2974 7632
4 1149 3405 4783 9509
5 734 1899 3063 3694 4475 6562 8890 9985
6 42 1120 1512 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
7 104 1004 1128 1347 1522 2849 3159 3397 4310 4494 4750 5016 5675 7324 7604
   8565 8961 9614
8 321 929 1213 1853 2040 4466 4644 4769 4830 5520 6082 6653 7516 8126 8611
   9300 9762
9 578 1323 1831 1886 2718 4699 4786 4883 5502 5550 6306 6570 7164 7989 8223
10 600 1730 2253 4961 5253 6425 6940 7706 8106 8135 8338
11 2164 2555 5136 6712 6966 7754 8204 8359
12 2082 5119 5246

```

```

13
14 Current Tree After Deleting Element 4769
15 3739
16 2974 7632
17 1149 3405 4783 9509
18 734 1899 3063 3694 4475 6562 8890 9985
19 42 1120 1512 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
20 104 1004 1128 1347 1522 2849 3159 3397 4310 4494 4750 5016 5675 7324 7604
    8565 8961 9614
21 321 929 1213 1853 2040 4466 4644 4830 5520 6082 6653 7516 8126 8611 9300
    9762
22 578 1323 1831 1886 2718 4699 4786 4883 5502 5550 6306 6570 7164 7989 8223
23 600 1730 2253 4961 5253 6425 6940 7706 8106 8135 8338
24 2164 2555 5136 6712 6966 7754 8204 8359
25 2082 5119 5246
26 -----

27 Current Tree Before Deleting Element 1853
28 3739
29 2974 7632
30 1149 3405 4783 9509
31 734 1899 3063 3694 4475 6562 8890 9985
32 42 1120 1512 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
33 104 1004 1128 1347 1522 2849 3159 3397 4310 4494 4750 5016 5675 7324 7604
    8565 8961 9614
34 321 929 1213 1853 2040 4466 4644 4830 5520 6082 6653 7516 8126 8611 9300
    9762
35 578 1323 1831 1886 2718 4699 4786 4883 5502 5550 6306 6570 7164 7989 8223
36 600 1730 2253 4961 5253 6425 6940 7706 8106 8135 8338
37 2164 2555 5136 6712 6966 7754 8204 8359
38 2082 5119 5246
39
40 Current Tree After Deleting Element 1853
41 3739
42 2974 7632
43 1149 3405 4783 9509
44 734 1899 3063 3694 4475 6562 8890 9985
45 42 1120 1512 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
46 104 1004 1128 1347 1522 2849 3159 3397 4310 4494 4750 5016 5675 7324 7604
    8565 8961 9614
47 321 929 1213 1886 2040 4466 4644 4830 5520 6082 6653 7516 8126 8611 9300
    9762
48 578 1323 1831 2718 4699 4786 4883 5502 5550 6306 6570 7164 7989 8223
49 600 1730 2253 4961 5253 6425 6940 7706 8106 8135 8338
50 2164 2555 5136 6712 6966 7754 8204 8359
51 2082 5119 5246
52 -----

53 Current Tree Before Deleting Element 4699
54 3739
55 2974 7632
56 1149 3405 4783 9509

```

```

57 734 1899 3063 3694 4475 6562 8890 9985
58 42 1120 1512 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
59 104 1004 1128 1347 1522 2849 3159 3397 4310 4494 4750 5016 5675 7324 7604
   8565 8961 9614
60 321 929 1213 1886 2040 4466 4644 4830 5520 6082 6653 7516 8126 8611 9300
   9762
61 578 1323 1831 2718 4699 4786 4883 5502 5550 6306 6570 7164 7989 8223
62 600 1730 2253 4961 5253 6425 6940 7706 8106 8135 8338
63 2164 2555 5136 6712 6966 7754 8204 8359
64 2082 5119 5246
65
66 Current Tree After Deleting Element 4699
67 3739
68 2974 7632
69 1149 3405 4783 9509
70 734 1899 3063 3694 4475 6562 8890 9985
71 42 1120 1512 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
72 104 1004 1128 1347 1522 2849 3159 3397 4310 4494 4750 5016 5675 7324 7604
   8565 8961 9614
73 321 929 1213 1886 2040 4466 4644 4830 5520 6082 6653 7516 8126 8611 9300
   9762
74 578 1323 1831 2718 4786 4883 5502 5550 6306 6570 7164 7989 8223
75 600 1730 2253 4961 5253 6425 6940 7706 8106 8135 8338
76 2164 2555 5136 6712 6966 7754 8204 8359
77 2082 5119 5246
78 -----

79 Current Tree Before Deleting Element 2082
80 3739
81 2974 7632
82 1149 3405 4783 9509
83 734 1899 3063 3694 4475 6562 8890 9985
84 42 1120 1512 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
85 104 1004 1128 1347 1522 2849 3159 3397 4310 4494 4750 5016 5675 7324 7604
   8565 8961 9614
86 321 929 1213 1886 2040 4466 4644 4830 5520 6082 6653 7516 8126 8611 9300
   9762
87 578 1323 1831 2718 4786 4883 5502 5550 6306 6570 7164 7989 8223
88 600 1730 2253 4961 5253 6425 6940 7706 8106 8135 8338
89 2164 2555 5136 6712 6966 7754 8204 8359
90 2082 5119 5246
91
92 Current Tree After Deleting Element 2082
93 3739
94 2974 7632
95 1149 3405 4783 9509
96 734 1899 3063 3694 4475 6562 8890 9985
97 42 1120 1512 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
98 104 1004 1128 1347 1522 2849 3159 3397 4310 4494 4750 5016 5675 7324 7604
   8565 8961 9614
99 321 929 1213 1886 2040 4466 4644 4830 5520 6082 6653 7516 8126 8611 9300
   9762

```



```

100 578 1323 1831 2718 4786 4883 5502 5550 6306 6570 7164 7989 8223
101 600 1730 2253 4961 5253 6425 6940 7706 8106 8135 8338
102 2164 2555 5136 6712 6966 7754 8204 8359
103 5119 5246
104 -----

105 Current Tree Before Deleting Element 4961
106 3739
107 2974 7632
108 1149 3405 4783 9509
109 734 1899 3063 3694 4475 6562 8890 9985
110 42 1120 1512 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
111 104 1004 1128 1347 1522 2849 3159 3397 4310 4494 4750 5016 5675 7324 7604
    8565 8961 9614
112 321 929 1213 1886 2040 4466 4644 4830 5520 6082 6653 7516 8126 8611 9300
    9762
113 578 1323 1831 2718 4786 4883 5502 5550 6306 6570 7164 7989 8223
114 600 1730 2253 4961 5253 6425 6940 7706 8106 8135 8338
115 2164 2555 5136 6712 6966 7754 8204 8359
116 5119 5246
117
118 Current Tree After Deleting Element 4961
119 3739
120 2974 7632
121 1149 3405 4783 9509
122 734 1899 3063 3694 4475 6562 8890 9985
123 42 1120 1512 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
124 104 1004 1128 1347 1522 2849 3159 3397 4310 4494 4750 5016 5675 7324 7604
    8565 8961 9614
125 321 929 1213 1886 2040 4466 4644 4830 5520 6082 6653 7516 8126 8611 9300
    9762
126 578 1323 1831 2718 4786 4883 5502 5550 6306 6570 7164 7989 8223
127 600 1730 2253 5253 6425 6940 7706 8106 8135 8338
128 2164 2555 5136 6712 6966 7754 8204 8359
129 5119 5246
130 -----

131 Current Tree Before Deleting Element 1512
132 3739
133 2974 7632
134 1149 3405 4783 9509
135 734 1899 3063 3694 4475 6562 8890 9985
136 42 1120 1512 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
137 104 1004 1128 1347 1522 2849 3159 3397 4310 4494 4750 5016 5675 7324 7604
    8565 8961 9614
138 321 929 1213 1886 2040 4466 4644 4830 5520 6082 6653 7516 8126 8611 9300
    9762
139 578 1323 1831 2718 4786 4883 5502 5550 6306 6570 7164 7989 8223
140 600 1730 2253 5253 6425 6940 7706 8106 8135 8338
141 2164 2555 5136 6712 6966 7754 8204 8359
142 5119 5246
143

```

```

144 Current Tree After Deleting Element 1512
145 3739
146 2974 7632
147 1149 3405 4783 9509
148 734 1899 3063 3694 4475 6562 8890 9985
149 42 1120 1522 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
150 104 1004 1128 1347 1886 2849 3159 3397 4310 4494 4750 5016 5675 7324 7604
      8565 8961 9614
151 321 929 1213 1831 2040 4466 4644 4830 5520 6082 6653 7516 8126 8611 9300
      9762
152 578 1323 1730 2718 4786 4883 5502 5550 6306 6570 7164 7989 8223
153 600 2253 5253 6425 6940 7706 8106 8135 8338
154 2164 2555 5136 6712 6966 7754 8204 8359
155 5119 5246
156 -----

157 Current Tree Before Deleting Element 1730
158 3739
159 2974 7632
160 1149 3405 4783 9509
161 734 1899 3063 3694 4475 6562 8890 9985
162 42 1120 1522 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
163 104 1004 1128 1347 1886 2849 3159 3397 4310 4494 4750 5016 5675 7324 7604
      8565 8961 9614
164 321 929 1213 1831 2040 4466 4644 4830 5520 6082 6653 7516 8126 8611 9300
      9762
165 578 1323 1730 2718 4786 4883 5502 5550 6306 6570 7164 7989 8223
166 600 2253 5253 6425 6940 7706 8106 8135 8338
167 2164 2555 5136 6712 6966 7754 8204 8359
168 5119 5246
169
170 Current Tree After Deleting Element 1730
171 3739
172 2974 7632
173 1149 3405 4783 9509
174 734 1899 3063 3694 4475 6562 8890 9985
175 42 1120 1522 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
176 104 1004 1128 1347 1886 2849 3159 3397 4310 4494 4750 5016 5675 7324 7604
      8565 8961 9614
177 321 929 1213 1831 2040 4466 4644 4830 5520 6082 6653 7516 8126 8611 9300
      9762
178 578 1323 2718 4786 4883 5502 5550 6306 6570 7164 7989 8223
179 600 2253 5253 6425 6940 7706 8106 8135 8338
180 2164 2555 5136 6712 6966 7754 8204 8359
181 5119 5246
182 -----

183 Current Tree Before Deleting Element 2849
184 3739
185 2974 7632
186 1149 3405 4783 9509
187 734 1899 3063 3694 4475 6562 8890 9985

```

```

188 42 1120 1522 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
189 104 1004 1128 1347 1886 2849 3159 3397 4310 4494 4750 5016 5675 7324 7604
      8565 8961 9614
190 321 929 1213 1831 2040 4466 4644 4830 5520 6082 6653 7516 8126 8611 9300
      9762
191 578 1323 2718 4786 4883 5502 5550 6306 6570 7164 7989 8223
192 600 2253 5253 6425 6940 7706 8106 8135 8338
193 2164 2555 5136 6712 6966 7754 8204 8359
194 5119 5246
195
196 Current Tree After Deleting Element 2849
197 3739
198 2974 7632
199 1149 3405 4783 9509
200 734 1899 3063 3694 4475 6562 8890 9985
201 42 1120 1522 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
202 104 1004 1128 1347 1886 2040 3159 3397 4310 4494 4750 5016 5675 7324 7604
      8565 8961 9614
203 321 929 1213 1831 2718 4466 4644 4830 5520 6082 6653 7516 8126 8611 9300
      9762
204 578 1323 2253 4786 4883 5502 5550 6306 6570 7164 7989 8223
205 600 2164 2555 5253 6425 6940 7706 8106 8135 8338
206 5136 6712 6966 7754 8204 8359
207 5119 5246
208 -----

209 Current Tree Before Deleting Element 5520
210 3739
211 2974 7632
212 1149 3405 4783 9509
213 734 1899 3063 3694 4475 6562 8890 9985
214 42 1120 1522 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
215 104 1004 1128 1347 1886 2040 3159 3397 4310 4494 4750 5016 5675 7324 7604
      8565 8961 9614
216 321 929 1213 1831 2718 4466 4644 4830 5520 6082 6653 7516 8126 8611 9300
      9762
217 578 1323 2253 4786 4883 5502 5550 6306 6570 7164 7989 8223
218 600 2164 2555 5253 6425 6940 7706 8106 8135 8338
219 5136 6712 6966 7754 8204 8359
220 5119 5246
221
222 Current Tree After Deleting Element 5520
223 3739
224 2974 7632
225 1149 3405 4783 9509
226 734 1899 3063 3694 4475 6562 8890 9985
227 42 1120 1522 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
228 104 1004 1128 1347 1886 2040 3159 3397 4310 4494 4750 5016 5675 7324 7604
      8565 8961 9614
229 321 929 1213 1831 2718 4466 4644 4830 5550 6082 6653 7516 8126 8611 9300
      9762
230 578 1323 2253 4786 4883 5502 6306 6570 7164 7989 8223

```

```

231 600 2164 2555 5253 6425 6940 7706 8106 8135 8338
232 5136 6712 6966 7754 8204 8359
233 5119 5246
234 -----

235 Current Tree Before Deleting Element 8890
236 3739
237 2974 7632
238 1149 3405 4783 9509
239 734 1899 3063 3694 4475 6562 8890 9985
240 42 1120 1522 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
241 104 1004 1128 1347 1886 2040 3159 3397 4310 4494 4750 5016 5675 7324 7604
    8565 8961 9614
242 321 929 1213 1831 2718 4466 4644 4830 5550 6082 6653 7516 8126 8611 9300
    9762
243 578 1323 2253 4786 4883 5502 6306 6570 7164 7989 8223
244 600 2164 2555 5253 6425 6940 7706 8106 8135 8338
245 5136 6712 6966 7754 8204 8359
246 5119 5246
247
248 Current Tree After Deleting Element 8890
249 3739
250 2974 7632
251 1149 3405 4783 9509
252 734 1899 3063 3694 4475 6562 8961 9985
253 42 1120 1522 2954 3262 3476 3902 4584 5644 7499 8740 9304 9971
254 104 1004 1128 1347 1886 2040 3159 3397 4310 4494 4750 5016 5675 7324 7604
    8565 9300 9614
255 321 929 1213 1831 2718 4466 4644 4830 5550 6082 6653 7516 8126 8611 9762
256 578 1323 2253 4786 4883 5502 6306 6570 7164 7989 8223
257 600 2164 2555 5253 6425 6940 7706 8106 8135 8338
258 5136 6712 6966 7754 8204 8359
259 5119 5246

```

Question #2

```

1 class Graph:
2     adjacency_list = {}
3     is_directed = False
4
5     def __init__(self, is_directed=False):
6         self.adjacency_list = {}
7         self.is_directed = is_directed
8
9     def add_edge(self, start, destination):
10        self.adjacency_list[start].add(destination)
11
12        if self.is_directed:
13            self.adjacency_list[destination].add(start)
14

```

```

15     def add_node(self, node):
16         if node not in self.adjacency_list:
17             self.adjacency_list[node] = set()
18
19     def print_graph(self):
20         max_row, max_column = self.max_dimensions
21         list_version = [[0 for i in range(max_column)] for j in range(
max_row)]
22
23         for source, destination_set in self.adjacency_list.items():
24             for destination in destination_set:
25                 list_version[source][destination] = 1
26
27         print(matrix(list_version))
28
29     @property
30     def max_dimensions(self):
31         max_element = max(self.adjacency_list.keys())
32
33         for _, value in self.adjacency_list.items():
34             value = list(value)
35
36             if value:
37                 if max(value) > max_element:
38                     max_element = max(value)
39
40         return max_element + 1, max_element + 1
41
42     def depth_first_search(self, start_node, preserve_order=True):
43         if preserve_order:
44             colors = {}
45             for vertex in self.adjacency_list:
46                 colors[vertex] = "white"
47
48             return self.__depth_first_search_preserve_order(start_node,
colors, None)
49         else:
50             return self.__depth_first_search(start_node, None)
51
52     def breadth_first_search(self, start_node, preserve_order=True):
53         if preserve_order:
54             return self.__breadth_first_search_preserve_order(start_node)
55         else:
56             return self.__breadth_first_search(start_node)
57
58     def __depth_first_search(self, start_node, visited_nodes):
59         if visited_nodes is None:
60             visited_nodes = set()
61
62         visited_nodes.add(start_node)
63         for unvisited_node in self.adjacency_list[start_node] -
visited_nodes:

```

```

64         self.__depth_first_search(unvisited_node, visited_nodes)
65
66     return visited_nodes
67
68     def __depth_first_search_preserve_order(self, start_node, colors,
visited_nodes):
69         if visited_nodes is None:
70             visited_nodes = []
71
72         if colors[start_node] is "white":
73             colors[start_node] = "grey"
74             visited_nodes += [start_node]
75
76         for unvisited_node in [x for x in self.adjacency_list[start_node]
if x not in visited_nodes]:
77             if colors[unvisited_node] is "white":
78                 self.__depth_first_search_preserve_order(unvisited_node,
colors, visited_nodes)
79
80         colors[start_node] = "black"
81
82     return visited_nodes
83
84     def __breadth_first_search(self, start):
85         visited, queue = set(), [start]
86
87         while queue:
88             vertex = queue.pop(0)
89
90             if vertex not in visited:
91                 visited.add(vertex)
92                 queue.extend(self.adjacency_list[vertex] - visited)
93
94         return visited
95
96     def __breadth_first_search_preserve_order(self, start):
97         colors = {}
98
99         for vertex in self.adjacency_list:
100             colors[vertex] = "white"
101
102         queue = [start]
103         visited_nodes = [start]
104
105         while queue:
106             node = queue.pop(0)
107
108             for unvisited_node in list(self.adjacency_list[node]):
109                 if colors[unvisited_node] is "white":
110                     colors[unvisited_node] = "grey"
111                     queue.insert(0, unvisited_node)
112                     visited_nodes += [unvisited_node]

```

```

113
114         colors[node] = "black"
115
116     return visited_nodes

```

We test examples with the following adjacency list.

$$\begin{bmatrix}
 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\
 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0
 \end{bmatrix}$$

Problem #2.1

For Depth-First Search, we get the following ordering (with the root taken at random).

$3 \rightarrow 4 \rightarrow 2 \rightarrow 8 \rightarrow 0 \rightarrow 9 \rightarrow 1 \rightarrow 6 \rightarrow 5 \rightarrow 7$
 $0 \rightarrow 9 \rightarrow 1 \rightarrow 2 \rightarrow 8 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 5 \rightarrow 7$
 $8 \rightarrow 0 \rightarrow 9 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 5 \rightarrow 7$
 $5 \rightarrow 4 \rightarrow 2 \rightarrow 8 \rightarrow 0 \rightarrow 9 \rightarrow 1 \rightarrow 3 \rightarrow 6 \rightarrow 7$
 $0 \rightarrow 9 \rightarrow 1 \rightarrow 2 \rightarrow 8 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 5 \rightarrow 7$
 $6 \rightarrow 0 \rightarrow 9 \rightarrow 1 \rightarrow 2 \rightarrow 8 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7$
 $3 \rightarrow 4 \rightarrow 2 \rightarrow 8 \rightarrow 0 \rightarrow 9 \rightarrow 1 \rightarrow 6 \rightarrow 5 \rightarrow 7$
 $5 \rightarrow 4 \rightarrow 2 \rightarrow 8 \rightarrow 0 \rightarrow 9 \rightarrow 1 \rightarrow 3 \rightarrow 6 \rightarrow 7$
 $7 \rightarrow 8 \rightarrow 0 \rightarrow 9 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 5$
 $9 \rightarrow 1 \rightarrow 2 \rightarrow 8 \rightarrow 0 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 5 \rightarrow 7$
 $3 \rightarrow 4 \rightarrow 2 \rightarrow 8 \rightarrow 0 \rightarrow 9 \rightarrow 1 \rightarrow 6 \rightarrow 5 \rightarrow 7$
 $8 \rightarrow 0 \rightarrow 9 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 5 \rightarrow 7$
 $5 \rightarrow 4 \rightarrow 2 \rightarrow 8 \rightarrow 0 \rightarrow 9 \rightarrow 1 \rightarrow 3 \rightarrow 6 \rightarrow 7$
 $0 \rightarrow 9 \rightarrow 1 \rightarrow 2 \rightarrow 8 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 5 \rightarrow 7$
 $2 \rightarrow 8 \rightarrow 0 \rightarrow 9 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 5 \rightarrow 7$

Problem #2.2

For Breadth-First Search, we get the following ordering (with the root taken at random).

$3 \rightarrow 4 \rightarrow 6 \rightarrow 0 \rightarrow 8 \rightarrow 5 \rightarrow 7 \rightarrow 9 \rightarrow 1 \rightarrow 2$
 $0 \rightarrow 9 \rightarrow 3 \rightarrow 4 \rightarrow 1 \rightarrow 2 \rightarrow 8 \rightarrow 5 \rightarrow 6 \rightarrow 7$
 $8 \rightarrow 0 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 9 \rightarrow 1 \rightarrow 4 \rightarrow 2$
 $5 \rightarrow 4 \rightarrow 2 \rightarrow 6 \rightarrow 0 \rightarrow 8 \rightarrow 3 \rightarrow 7 \rightarrow 9 \rightarrow 1$
 $0 \rightarrow 9 \rightarrow 3 \rightarrow 4 \rightarrow 1 \rightarrow 2 \rightarrow 8 \rightarrow 5 \rightarrow 6 \rightarrow 7$
 $6 \rightarrow 0 \rightarrow 8 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 9 \rightarrow 1 \rightarrow 4 \rightarrow 2$
 $3 \rightarrow 4 \rightarrow 6 \rightarrow 0 \rightarrow 8 \rightarrow 5 \rightarrow 7 \rightarrow 9 \rightarrow 1 \rightarrow 2$
 $5 \rightarrow 4 \rightarrow 2 \rightarrow 6 \rightarrow 0 \rightarrow 8 \rightarrow 3 \rightarrow 7 \rightarrow 9 \rightarrow 1$
 $7 \rightarrow 8 \rightarrow 0 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 9 \rightarrow 1 \rightarrow 4 \rightarrow 2$
 $9 \rightarrow 1 \rightarrow 4 \rightarrow 6 \rightarrow 0 \rightarrow 8 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 2$
 $3 \rightarrow 4 \rightarrow 6 \rightarrow 0 \rightarrow 8 \rightarrow 5 \rightarrow 7 \rightarrow 9 \rightarrow 1 \rightarrow 2$
 $8 \rightarrow 0 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 9 \rightarrow 1 \rightarrow 4 \rightarrow 2$
 $5 \rightarrow 4 \rightarrow 2 \rightarrow 6 \rightarrow 0 \rightarrow 8 \rightarrow 3 \rightarrow 7 \rightarrow 9 \rightarrow 1$
 $0 \rightarrow 9 \rightarrow 3 \rightarrow 4 \rightarrow 1 \rightarrow 2 \rightarrow 8 \rightarrow 5 \rightarrow 6 \rightarrow 7$
 $2 \rightarrow 8 \rightarrow 0 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 9 \rightarrow 1 \rightarrow 4$

Question #3

Our graph is exactly the same as the previous version; however, this one has the boolean property (`is_directed` set to false). We test examples with the following adjacency list.

$$\begin{bmatrix}
 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0
 \end{bmatrix}$$

Problem #3.1

For Depth-First Search, we get the following ordering (with the root taken at random).

$3 \rightarrow 9 \rightarrow 0 \rightarrow 8 \rightarrow 1 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 7 \rightarrow 6$
 $9 \rightarrow 0 \rightarrow 8 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 7 \rightarrow 6$
 $2 \rightarrow 0 \rightarrow 8 \rightarrow 1 \rightarrow 3 \rightarrow 9 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$
 $4 \rightarrow 9 \rightarrow 0 \rightarrow 8 \rightarrow 1 \rightarrow 3 \rightarrow 6 \rightarrow 7 \rightarrow 2 \rightarrow 5$
 $9 \rightarrow 0 \rightarrow 8 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 7 \rightarrow 6$
 $0 \rightarrow 8 \rightarrow 1 \rightarrow 3 \rightarrow 9 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 7 \rightarrow 6$
 $4 \rightarrow 9 \rightarrow 0 \rightarrow 8 \rightarrow 1 \rightarrow 3 \rightarrow 6 \rightarrow 7 \rightarrow 2 \rightarrow 5$
 $2 \rightarrow 0 \rightarrow 8 \rightarrow 1 \rightarrow 3 \rightarrow 9 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$
 $4 \rightarrow 9 \rightarrow 0 \rightarrow 8 \rightarrow 1 \rightarrow 3 \rightarrow 6 \rightarrow 7 \rightarrow 2 \rightarrow 5$
 $1 \rightarrow 8 \rightarrow 0 \rightarrow 9 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 7 \rightarrow 6$
 $0 \rightarrow 8 \rightarrow 1 \rightarrow 3 \rightarrow 9 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 7 \rightarrow 6$
 $7 \rightarrow 8 \rightarrow 0 \rightarrow 9 \rightarrow 3 \rightarrow 1 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 6$
 $7 \rightarrow 8 \rightarrow 0 \rightarrow 9 \rightarrow 3 \rightarrow 1 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 6$
 $5 \rightarrow 0 \rightarrow 8 \rightarrow 1 \rightarrow 3 \rightarrow 9 \rightarrow 4 \rightarrow 2 \rightarrow 7 \rightarrow 6$
 $4 \rightarrow 9 \rightarrow 0 \rightarrow 8 \rightarrow 1 \rightarrow 3 \rightarrow 6 \rightarrow 7 \rightarrow 2 \rightarrow 5$

Problem #3.2

For Breadth-First Search, we get the following ordering (with the root taken at random).

3 → 9 → 1 → 4 → 6 → 7 → 8 → 2 → 0 → 5
 9 → 0 → 3 → 4 → 2 → 5 → 1 → 8 → 7 → 6
 2 → 0 → 8 → 4 → 5 → 7 → 1 → 3 → 9 → 6
 4 → 9 → 2 → 3 → 5 → 1 → 8 → 7 → 0 → 6
 9 → 0 → 3 → 4 → 2 → 5 → 1 → 8 → 7 → 6
 0 → 8 → 9 → 2 → 5 → 4 → 3 → 1 → 7 → 6
 4 → 9 → 2 → 3 → 5 → 1 → 8 → 7 → 0 → 6
 2 → 0 → 8 → 4 → 5 → 7 → 1 → 3 → 9 → 6
 4 → 9 → 2 → 3 → 5 → 1 → 8 → 7 → 0 → 6
 1 → 8 → 3 → 4 → 7 → 2 → 0 → 5 → 9 → 6
 0 → 8 → 9 → 2 → 5 → 4 → 3 → 1 → 7 → 6
 7 → 8 → 1 → 2 → 3 → 9 → 4 → 6 → 5 → 0
 7 → 8 → 1 → 2 → 3 → 9 → 4 → 6 → 5 → 0
 5 → 0 → 2 → 4 → 9 → 3 → 1 → 8 → 7 → 6
 4 → 9 → 2 → 3 → 5 → 1 → 8 → 7 → 0 → 6

Question #4

```

1 class ParentheticalOrder(Graph):
2     def print_parenthetical_order(self, start_node):
3         colors = {}
4         for vertex in self.adjacency_list:
5             colors[vertex] = "white"
6
7         for node in sorted(self.adjacency_list):
8             if colors[node] is "white":
9                 self.__print_parenthetical_order(node, colors)
10
11     def __print_parenthetical_order(self, node, colors):
12         colors[node] = "grey"
13         print("{ } ".format(node), end='')
14         for neighbor in sorted(self.adjacency_list[node]):
15             if colors[neighbor] is "white":
16                 self.__print_parenthetical_order(neighbor, colors)
17         colors[node] = 'black'
18         print(" { } ".format(node), end='')

```

We get the following string:

(u (v (y (x x) y) v) u) (w (z z) w)

Question #5

Question #6

```

1 class WrestleMania(Graph):
2     def determine_valid_rivalry(self):
3         rivalry, not_visited = {}, list(self.adjacency_list.keys())
4
5         for vertex in self.adjacency_list:
6             rivalry[vertex] = "none"
7
8         while "none" in rivalry.values():
9             current_depth, start = 0, not_visited[-1]
10            queue = [start]
11
12            while queue:
13                current_depth += 1
14                node = queue.pop(0)
15
16                for unvisited_node in list(self.adjacency_list[node]):
17                    if rivalry[unvisited_node] is "none":
18                        if current_depth % 2 == 0:
19                            rivalry[unvisited_node] = "good guy"
20                        else:
21                            rivalry[unvisited_node] = "bad guy"
22
23                    not_visited.remove(unvisited_node)
24                    queue.insert(0, unvisited_node)
25
26            for wrestler, adjacency_wrestlers in self.adjacency_list.items():
27                for adjacent_wrestler in adjacency_wrestlers:
28                    if rivalry[wrestler] == rivalry[adjacent_wrestler]:
29                        return False
30
31            return True

```

For the following sample inputs

$$X_1 = \begin{bmatrix} & x & y & z \\ x & 1 & 0 & 1 \\ y & 0 & 0 & 1 \\ z & 1 & 1 & 0 \end{bmatrix} \quad X_2 = \begin{bmatrix} & w & x & y & z \\ w & 0 & 1 & 0 & 1 \\ x & 1 & 0 & 1 & 0 \\ y & 0 & 1 & 0 & 1 \\ z & 1 & 0 & 1 & 0 \end{bmatrix} \quad X_3 = \begin{bmatrix} & u & v & w & x & y & z \\ u & 0 & 1 & 0 & 0 & 0 & 0 \\ v & 1 & 0 & 0 & 0 & 0 & 0 \\ w & 0 & 0 & 0 & 1 & 0 & 0 \\ x & 0 & 0 & 1 & 0 & 0 & 0 \\ y & 0 & 0 & 0 & 0 & 0 & 1 \\ z & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

```

1 No configuration could be found.
2 =====
3 Wrestler y Is A Baby Face

```

```

4 Wrestler x Is A Heel
5 Wrestler z Is A Heel
6 Wrestler w Is A Baby Face
7 =====
8 Wrestler u Is A Heel
9 Wrestler w Is A Heel
10 Wrestler v Is A Baby Face
11 Wrestler y Is A Heel
12 Wrestler x Is A Baby Face
13 Wrestler z Is A Baby Face

```

Question #7

Problem #7.1

For this problem we have three cases.

No Children If the root has no children it has no edges so it cannot disconnect the graph.

One Child The only edge it breaks is paths to itself so it cannot disconnect graph.

≥ 2 *Children* Because there are no cross edges between the children, and because the children nodes are in the subtrees of the root, no path exists between them. Deleting the root will delete the link between the two or more children; thus, disconnecting the graph and making the root G_π an articulation point.

Problem #7.2

Suppose there is a vertex $v \in G_\pi$, where v is a non-root. Furthermore, suppose v has a child s such that there is no back edge from or any descendent of s to ancestor of v . Thus, the removal of v will lead the disconnection of the sub-tree rooted at v from the graph G_π .

Therefore, the absence of an edge between the descendants of s and ancestors of v resulted in the disconnection of the graph after removal of non-root vertex v . This implies v is an articulation point.

Problem #7.3

Because v is discovered before all of its descendants, the only edges that could affect the minimum are ancestors of v . Thus, we can do an augmented depth-first search to determine all of the low values.

```

1 class ArticulateGraph(Graph):
2     time = 0
3     times, colors, lows = {}, {}, {}
4
5     def find_minimum(self, start_node, visited_nodes=None):

```

```

6         if visited_nodes is None:
7             self.times, self.colors, self.lows = {}, {}, {}
8             self.time = 0
9
10            for vertex in self.adjacency_list:
11                self.colors[vertex] = "white"
12
13            visited_nodes = []
14
15            for unvisited_node in [x for x in self.adjacency_list[start_node]
16 if x not in visited_nodes]:
17                if self.colors[unvisited_node] is "white":
18                    self.colors[start_node] = "grey"
19                    self.time += 1
20                    visited_nodes += [start_node]
21
22                    self.times[start_node] = self.time
23                    self.lows[start_node] = self.times[start_node]
24
25                    for adjacent_node in self.adjacency_list[start_node]:
26                        self.find_minimum(adjacent_node, visited_nodes)
27
28                        if self.colors[adjacent_node] is "white":
29                            if self.time[start_node] < self.lows[self.
adjacency_list]:
30                                self.lows[start_node] = self.times[
adjacent_node]
31
32                    self.colors[start_node] = "black"
33                    self.time += 1
34
35            return self.lows

```

With the following graph,

$$\begin{bmatrix}
 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
 \end{bmatrix}$$

We get the following:

$8 \rightarrow 3$

$2 \rightarrow 2$

$3 \rightarrow 1$

$4 \rightarrow 4$

$7 \rightarrow 9$

Problem #7.4

After applying the algorithm mentioned above for all $v \in V$, we check to see if $v.low = v.d$. If it is, no descendants of v has a back edge to a proper ancestor of v , implying v is not a articulation point.

Problem #7.5

If there is a cycle from $u \rightarrow v \rightarrow w$, where u and v have unique partitions, then removing any edge of w does not disconnect the graph. Therefore, the edges from w to u and v cannot be bridges.

Problem #7.6

A simple circuit contains an edge (u, v) if and only if

- Both of its endpoints are articulation points.
- One of its endpoints is an articulation point and the other is a vertex of degree 1.

Using our previous algorithm above, we can compute this by running `find_minimum(self, v)`, and deciding if one of the two criteria are met.

Problem #7.7

Because we have already stated that any two edges of any bi-connected components lie on a common cycle, we know all edges are lying inside a component of a graph will not be bridges (via Part E). Therefore, all bi-connected components of a graph partition the non-bridges of said graph.

Problem #7.8

Locate all bridge edges using the algorithm described in Part F. Remove each bridge, the connected components are all biconnected components.

Homework #8

Analysis of Algorithms

Illya Starikov

Due Date: November 27th, 2017

Contents

Listings	1
Question #1	2
Question #2	3
Question #3	4
Question #4	7
Question #5	8
Question #6	9
Question #7	12
Question #8	13
Question #9	19

Listings

1	Problem #8.2	4
2	Problem #5	9
3	Problem #8.1	13
4	Problem #8.2	15
5	Problem #9	19

Question #1

Let the samples space $S = \{X_{\text{free}} \implies \text{prisoner } X \text{ is going free, } Y_{\text{free}} \implies \text{prisoner } Y \text{ is going free, } Z_{\text{free}} \implies \text{prisoner } Z \text{ is going free, } \}$. Furthermore, let y be event that Y is to be executed. Using Bayes Theorem, we get the following:

$$\begin{aligned}\Pr(X_{\text{free}}|y) &= \frac{\Pr(y|X_{\text{free}}) \times \Pr(X_{\text{free}})}{\Pr(y \cap X_{\text{free}}) + \Pr(y \cap Y_{\text{free}}) + \Pr(y \cap Z_{\text{free}})} \\ &= \frac{\frac{1}{3} \times \frac{1}{2}}{\frac{1}{3} \times \frac{1}{2} + 0 + \frac{1}{3} \times 1} \\ &= \frac{1}{3}\end{aligned}$$

We see that the prisoner X still has a probability of $\frac{1}{3}$.

Question #2

$$\text{minimum} = 2^h \qquad \text{maximum} = 2^{h+1} - 1$$

Question #3

A simple way to achieve an $\mathcal{O}(n \lg k)$ algorithm.

1. Sort all of the arrays in ascending order.
2. Create a minimum heap with all of the minimum elements in the k individual lists. Remove these elements from the respective arrays.
3. Remove the minimum element (name it $d_{\min}$) from the heap mentioned in Step 2, and replace it with the minimum from it's respective list. Add $d_{\min}$ to the solution.
4. While the heap is not empty, repeat Step 3.

Listing 1: Problem #8.2

```
1  #!/usr/local/bin/python3
2  #
3  # problem-3.py
4  # source
5  #
6  # Created by Illya Starikov on 11/16/17.
7  # Copyright 2017. Illya Starikov. All rights reserved.
8  #
9
10 from collections import defaultdict
11
12
13 class MinHeap():
14     """A Min-Max Heap Implementation"""
15     __heap_list = []
16     __current_size = 0
17
18     def __init__(self):
19         self.__heaplist = [0]
20         self.__current_size = 0
21
22     def heapify(self, list_):
23         i = len(list_) // 2
24         self.__current_size = len(list_)
25         self.__heap_list = [0] + list_[:]
26         while (i > 0):
27             self.percolate_down(i)
28             i = i - 1
29
30     def percolate_down(self, i):
31         while (i * 2) <= self.__current_size:
32             minimum_child = self.min_child(i)
33             if self.__heap_list[i] > self.__heap_list[minimum_child]:
34                 tmp = self.__heap_list[i]
35                 self.__heap_list[i] = self.__heap_list[minimum_child]
```

```

36         self.__heap_list[minimum_child] = tmp
37         i = minimum_child
38
39     def percolate_up(self, i):
40         while i // 2 > 0:
41             if self.__heap_list[i] < self.__heap_list[i // 2]:
42                 tmp = self.__heap_list[i // 2]
43                 self.__heap_list[i // 2] = self.__heap_list[i]
44                 self.__heap_list[i] = tmp
45             i = i // 2
46
47     def insert(self, k):
48         self.__heap_list.append(k)
49         self.__current_size = self.__current_size + 1
50         self.percolate_up(self.__current_size)
51
52     def min_child(self, i):
53         if i * 2 + 1 > self.__current_size:
54             return i * 2
55         else:
56             if self.__heap_list[i*2] < self.__heap_list[i*2+1]:
57                 return i * 2
58             else:
59                 return i * 2 + 1
60
61     def delete_min(self):
62         to_return = self.__heap_list[1]
63
64         self.__heap_list[1] = self.__heap_list[self.__current_size]
65         self.__current_size = self.__current_size - 1
66         self.__heap_list.pop()
67         self.percolate_down(1)
68
69         return to_return
70
71     def print_heap(self):
72         print(self.__heap_list)
73
74     def __len__(self):
75         return self.__current_size
76
77     def empty_tree(input_list):
78         """Recursively iterate through values in nested lists."""
79         for item in input_list:
80             if not isinstance(item, list) or not empty_tree(item):
81                 return False
82         return True
83
84     def k_way_merge(enumerables):
85         heap_values = defaultdict(list)
86         heap = MinHeap()
87         solution = []

```

```

88
89     for list_ in enumerables:
90         list_ = sorted(list_)
91         min_value = list_.pop(0)
92
93         heap_values[min_value].append(list_)
94
95     heap.heapify(list(heap_values.keys()))
96
97     while len(heap) > 0:
98         minimum = heap.delete_min()
99         solution.append(minimum)
100
101         minimum_list = [] if heap_values[minimum] == [] else
102         heap_values[minimum][-1]
103         del heap_values[minimum]
104
105         if minimum_list != []:
106             new_minimum = minimum_list.pop(0)
107             heap_values[new_minimum].append(minimum_list)
108
109             heap.insert(new_minimum)
110
111     return solution
112
113 def main():
114     lists = [[3, 2, 1, 0], [5, 4], [9, 8, 7, 6]]
115     print(k_way_merge(lists))
116
117 if __name__ == "__main__":
118     main()

```

Question #4

Recall [Stirling's Approximation](#) for factorials:

$$n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \quad (1)$$

We skip the $\frac{1}{2}$ and $\frac{1}{n}$ case; for if we can't prove $\frac{1}{2^n} \times n! \in \Omega(n)$, then neither are $\frac{1}{2}$ and $\frac{1}{n}$.

Theorem 1.

$$\forall n \in \mathbb{R}^+, \frac{1}{2^n} \times n! \notin \Omega(n)$$

Proof. Suppose not. That is, suppose $\exists c \in \mathbb{R}, \frac{1}{2^n} \times n! \leq c n$. Therefore, $\frac{1}{2^n} \times n!$ must be in the same growth class as n . By definition, the limit of the two functions must be convergent to some number M .

Because we cannot directly take the limit, we use Equation 1. From this we get the result

$$\lim_{n \rightarrow \infty} \frac{\frac{1}{2^n} \times n!}{c n} = \frac{\frac{1}{2^n} \times \sqrt{2\pi n} \left(\frac{n}{e}\right)^n}{c n} = \infty \quad \forall c \in \mathbb{R}^+$$

This has led us to a contradiction. □

Question #5

Proof. Suppose we have n integers of length L (we can make this assumption because we can always pad the leading digits with 0s).

Base Case Solving with $L = 1$ is trivial. Because it assumed our sorting algorithm is correct, this will simply sort the digits.

Inductive Hypothesis Suppose that it is true for $n - 1$; we will show it to be true for n .

When comparing the n th digit, call it d_1 and d_2 from the two respective lists, it appears something like such:

...	...	d_1	...	...
...	...	d_2	...	...

From this, there are three cases:

$d_1 < d_2$ Then d_1 is placed before d_2 .

$d_1 > d_2$ Then d_1 is placed after d_1 .

$d_1 = d_2$ Then the numbers go unchanged. **Because our sorting algorithm is stable**, the order is preserved from the lower digits. This is shown below.

...	...	42	64	...
...	...	42	16	...

□

Question #6

We can accomplish $n + \lg n - 2$ runtime as such:

1. Split all elements into pairwise elements $(S \implies (s_1, s_2), (s_3, s_4), \dots, (s_{n-1}, s_n))$.
2. Compare all of the elements with respect to the smallest element in the pair. This will get the smallest. This requires n iterations.
3. The second smallest value must have been an element that lost to the original minimum, so we cache all of the lost contenders. We then run the previous step on this to get the second minimum. This requires at most $\lg n - 1$ iterations.

We first the following sample input $n = 10$:

342	133	40	365	796	716	354	38	256	614

We get the following output:

$$\begin{aligned}\text{Number of Comparison} &\in \Omega(n + \lceil \lg n \rceil - 2) \\ &\leq 10 + \lg 10 - 2 \\ &\leq 12 \\ &= 7\end{aligned}$$

Listing 2: Problem #5

```
1  #!/usr/local/bin/python3
2  #
3  # problem-5.py
4  # source
5  #
6  # Created by Illya Starikov on 11/24/17.
7  # Copyright 2017. Illya Starikov. All rights reserved.
8  #
9
10 from random import sample
11 from math import log2, ceil
12
13
14 def static_vars(**kwargs):
15     def decorate(func):
16         for k in kwargs:
17             setattr(func, k, kwargs[k])
18         return func
19     return decorate
20
```

```

21
22 def pairwise(iterable):
23     """s -> (s0, s1), (s2, s3), (s4, s5), ..."""
24
25     a = iter(iterable)
26     return list(zip(a, a))
27
28
29 @static_vars(comparison_counter=0)
30 def find_smallest_value(iterable, pairwised=False):
31
32     if iterable != []:
33         if pairwised is False:
34             iterabale_pairwise = pairwise(iterable)
35         else:
36             iterabale_pairwise = iterable
37
38         minimum = iterabale_pairwise.pop(0)
39
40         for element in iterabale_pairwise:
41             find_smallest_value.comparison_counter += 1
42
43             if min(element) < min(minimum):
44                 minimum = element
45
46         return min(minimum), max(minimum), [x for x in iterable if x
47 not in minimum], find_smallest_value.comparison_counter
48
49     return None, None, None, None
50
51 def find_second_smallest_element(iterable):
52     """Returns second smallest of two inputs"""
53     smallest, pair_of_smallest, new_iterable, _ = find_smallest_value(
54     iterable)
55     smallest_second_iteration, _, _, comparison_counter =
56     find_smallest_value(new_iterable)
57
58     second_smallest = pair_of_smallest if smallest_second_iteration is
59     None else min(pair_of_smallest, smallest_second_iteration)
60     return second_smallest, comparison_counter
61
62 def generate_input(of_length):
63     """Generates random input for problem #7. Returns (iterable,
64     second_smallest_element)"""
65     input_ = sample(range(of_length * 100), of_length)
66
67     smallest = min(input_)
68     second_smallest = min([x for x in input_ if smallest != x])
69
70     return input_, second_smallest

```



```

68
69
70 def main():
71     length_of_input = 10
72
73     input_iterable, second_smallest_from_input = generate_input(
74         length_of_input)
75     second_smallest, comparisons = find_second_smallest_element(
76         input_iterable)
77     print("Input: {}".format(input_iterable))
78
79     print("Length: {}\nElements {} = {} => {}\nNumber of Comparisons
80         ({} ) <= n + lg(n) - 2 ({} ) => {}".format(
81             length_of_input,
82             second_smallest_from_input, second_smallest,
83             second_smallest_from_input == second_smallest,
84             comparisons, length_of_input + ceil(log2(length_of_input)) - 2,
85             (comparisons <= length_of_input + ceil(log2(length_of_input)) - 2)
86         ))
87
88 if __name__ == "__main__":
89     main()

```

Question #7

Theorem 2. *The Set-Partition Problem is NP-Complete.*

Proof. It is trivial to prove that The Set-Partition problem is NP. Given two candidate solutions A and $\bar{A}$, verify that following things:

$$\sum_{x \in A} x - \sum_{x \in \bar{A}} x = 0 \quad \text{and} \quad A \cup \bar{A} = S$$

Recall The Subset Sum problem as follows:

Given a set of natural numbers S and a natural number t , is there a subset of S that sums to t

We will prove that The Set-Partition problem reduces down to The Subset Sum problem.

Suppose we have $p, t \in \mathbb{R}$ and a set P , where t is a particular number and p is $p = \sum_{x \in P} x$. We give input to Set-Partition $P^* = P \cup s - 2t$. From this, we have two cases.

1. If there exists $t \in P$, then remaining numbers in P sum to $s - t$. This satisfies the Set-Partition problem.
2. There exists a partition of X^* into two sets Q, Q^* such that the $\sum_{x \in Q} = \sum_{x \in Q^*} = s - t$. This implies $s - 2t \in Q * \cup Q$. Removing this number, one set sums to t , and we can use the previous case.

Because we have proved The Set-Partition Problem is NP and it reduces to an NP-Complete problem, we conclude that the Set-Partition problem is NP Complete. $\square$

Question #8

Problem #8.1

There exists an algorithm. Because there are only two denominations, we can enumerate all possible solutions. By producing pairs of all values in amount of x and the amount of y , we get a solution that's $\mathcal{O}(\text{amount of } x \times \text{amount of } y) = \mathcal{O}(n^2)$.

Listing 3: Problem #8.1

```
1  #!/usr/local/bin/python3
2  #
3  # problem-8-1.py
4  # source
5  #
6  # Created by Illya Starikov on 11/26/17.
7  # Copyright 2017. Illya Starikov. All rights reserved.
8  #
9
10
11 def all_combinations(list_one, list_two):
12     """Returns all two-element combinations of both lists"""
13
14     all_combinations = []
15     for element_one in list_one:
16         for element_two in list_two:
17             all_combinations += [[element_one, element_two]]
18
19     return all_combinations
20
21
22 def bonnie_and_clyde(x_worth, y_worth, x_amount, y_amount):
23     """Literally a brute force solution"""
24     all_combinations_of_x_y = all_combinations(range(x_amount), range(
25         y_amount))
26
27     for candidate_x, candidate_y in all_combinations_of_x_y:
28         if (candidate_x * x_worth + candidate_y * y_worth) == (x_worth
29             * (x_amount - candidate_x) + y_worth * (y_amount - candidate_y)):
30             return (True, candidate_x, candidate_y)
31
32     return (False, None, None)
33
34 def main():
35     # Impossible Case
36     x_worth, y_worth = 100, 7
37     x_amount, y_amount = 1, 2
38
39     print(bonnie_and_clyde(x_worth, y_worth, x_amount, y_amount))
40
41     # Possible Case (Literally one of each)
```

```
41     x_worth, y_worth = 100, 7
42     x_amount, y_amount = 2, 2
43
44     print(bonnie_and_clyde(x_worth, y_worth, x_amount, y_amount))
45
46 if __name__ == "__main__":
47     main()
```

Problem #8.2

There exists an algorithm. And it works like such:

1. Sort the coins in ascending order.
2. Pop off the largest coin, give it to Bonnie.
3. Keep taking smallest coins and assign them to Clyde. If finished, and still have coins to assign, go to Step 2.
4. If at any there is not enough coins to match Bonnie when assigning to Clyde in Step 3, there exists no such configuration.

Listing 4: Problem #8.2

```
1  #!/usr/local/bin/python3
2  #
3  # problem-8-2.py
4  # source
5  #
6  # Created by Illya Starikov on 11/26/17.
7  # Copyright 2017. Illya Starikov. All rights reserved.
8  #
9
10
11 def bonnie_and_clyde(coins):
12     coins_sorted = sorted(coins)
13
14     bonnie_coins, clyde_coins = [], []
15
16     while coins_sorted != []:
17         bonnie_coins += [coins_sorted.pop()]
18
19         while sum(bonnie_coins) != sum(clyde_coins):
20             if not coins_sorted:
21                 return [None, None]
22
23             clyde_coins += [coins_sorted.pop(0)]
24
25     return [bonnie_coins, clyde_coins]
26
27
28 def main():
29     # possible configuration
30     coins = [1, 1, 2, 4, 8]
31     print(bonnie_and_clyde(coins))
32
33     # impossible configuration
34     coins = [1, 2, 4, 64]
35     print(bonnie_and_clyde(coins))
36
```

```
37
38 if __name__ == "__main__":
39     main()
```

Problem #8.3

There exists no polynomial algorithm.

Proof. Note to Reader: We shall call this the Bonnie-Clyde Check problem, and the solution set of checks S_{Bonnie} for Bonnie and S_{Clyde} for Clyde.

It is trivial to prove that this Bonnie-Clyde check problem is NP. Given a solution S_{Bonnie} and S_{Clyde} , we must verify

$$\sum_{x \in S_{\text{Bonnie}}} x\text{'s worth} = \sum_{x \in S_{\text{Clyde}}} x\text{'s worth} \quad |S_{\text{Bonnie}}| + |S_{\text{Clyde}}| = n$$

where $|A|$ is the cardinality of A ¹. This is clearly a polynomial algorithm.

We will now show that the Bonnie-Clyde Check problem reduces down to the Partition Problem.

Assign every checks value as it's representation in the input set S . Feed S into The Partition problem. All output from The Partition problem is also valid output for the Bonnie Clyde problem.

Because we have proved that the Bonnie-Clyde problem is NP and reduces to an NP Complete problem, we conclude that the Bonnie-Clyde problem is NP complete. $\square$

¹Cardinality meaning the amount of elements in the set

Problem #8.4

The problem is also NP Complete.

Proof. Note to Reader: We shall call this the Bonnie-Clyde 100 Check problem, and the solution set of checks S_{Bonnie} for Bonnie and S_{Clyde} for Clyde.

If the minimum check value is greater than 100\$, the proof is identical to the previous.

If the minimum check is less than 100\$, first we must prove it is in NP. Given a solution S_{Bonnie} and S_{Clyde} , we must verify

$$\left| \sum_{x \in S_{\text{Bonnie}}} x\text{'s worth} - \sum_{x \in S_{\text{Clyde}}} x\text{'s worth} \right| = 100 \quad |S_{\text{Bonnie}}| + |S_{\text{Clyde}}| = n$$

where $|A|$ is the cardinality of A ². This is clearly a polynomial algorithm.

We will now show that the Bonnie-Clyde 100 Check problem reduces to the partition problem.

If there is a valid solution via Partition problem, we know there to be a solution. If not, we append a 1 to the distribution of checks, and rerun Set-Partition. Does this until either:

- We reach a valid solution.
- We reach 100.

Either way, this problem reduces to the Partition Problem.

Because we have proved that the Bonnie-Clyde problem is NP and reduces to an NP Complete problem, we conclude that the Bonnie-Clyde problem is NP complete. $\square$

²Cardinality meaning the amount of elements in the set

Question #9

A greedy algorithm is as follows:

1. Pick two arbitrary vertices u and v .
2. Add both u and v to the vertex cover.
3. Delete all incident edges to u and v .
4. If the adjacency matrix still has edges, go to Step 1.

For the adjacency matrix

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

We get the minimum vertex cover:

$$\{1, 2, 3, 4, 5\}$$

Listing 5: Problem #9

```
1  #!/usr/local/bin/python3
2  #
3  # problem-9.py
4  # source
5  #
6  # Created by Illya Starikov on 11/27/17.
7  # Copyright 2017. Illya Starikov. All rights reserved.
8  #
9
10 import random
11
12
13 def vertex_cover(adjacency_matrix):
14     if sum([item for sublist in adjacency_matrix for item in sublist])
15         != 0:
16         random_verties = random.sample(range(len(adjacency_matrix[0])),
17             2)
18         for i in range(len(adjacency_matrix[0])):
19             adjacency_matrix[i][random_verties[0]] = 0
20             adjacency_matrix[random_verties[0]][i] = 0
21
22             adjacency_matrix[i][random_verties[1]] = 0
23             adjacency_matrix[random_verties[1]][i] = 0
```

```

24         return vertex_cover(adjacency_matrix).union(set(random_verties)
25     )
26     else:
27         return set()
28
29
30 def main():
31     matrix = [[0, 0, 1, 0, 1],
32               [1, 0, 0, 1, 0],
33               [0, 1, 0, 0, 1],
34               [1, 0, 1, 0, 0],
35               [1, 0, 0, 0, 1]]
36
37     print(vertex_cover(matrix))
38
39
40 if __name__ == "__main__":
41     main()

```

Homework #8

Analysis of Algorithms

Illya Starikov

Due Date: November 27th, 2017

Contents

Listings	1
Question #1	2
Question #2	5
Question #3	6
Question #4	9
Question #5	11
Question #6	12
Question #7	13
Question #8	15

Listings

Question #1

Theorem 1. We take the definition of $\mathcal{O}$ as such.

For $f(n) \in \mathcal{O}(um_1)$, there must exist constants c , where $c > 0$, and there must exist $n_0 \in \mathbb{N}$, where $n_0 \geq 1$, such that

$$|f(n)| \leq c|g(n)| \quad \forall n \geq n_0$$

Problem #1.1

Problem Statement. *Without using limits, but only the definition of $\mathcal{O}$ prove that $81n^3 + 1300n^2 + 300n \in \mathcal{O}(um_1)$, but that $n^5 - 15000n^4 - 10n^3 \notin \mathcal{O}(um_1)$. Show all work.*

Proof. In this instance, $f(n) = 81n^3 + 1300n^2 + 300n$ and $g(n) \in \mathcal{O}(um_1)$. To prove that $f(n) \in \mathcal{O}(um_1)$, first we must find constants c and n_0 .

Take the following, $c = 1$, $n_0 = 15\,000$. Then, for all $n \geq 15\,000$,

$$81n^3 + 1300n^2 + 300n \ll n^5 - 15000n^4 - 10n^3$$

We see this is true, because

$$\begin{aligned} f(n) &= 81n^3 + 1300n^2 + 300n \\ &\leq 81n^3 + 1300n^3 + 300n^3 \\ &\leq 1681n^3 \\ g(n) &= n^5 - 15000n^4 - 10n^3 \\ &\geq n^3 - 15000^3 - 10n^3 \\ &\geq -14991n^3 \end{aligned}$$

From this we, we see that

$$|f(n)| \leq c|g(n)| \quad \forall n \geq n_0$$

Therefore, $f(n) \in \mathcal{O}(um_1)$, and $81n^3 + 1300n^2 + 300n \in \mathcal{O}(um_1)$. □

Proof. Suppose not. That is, suppose that $\exists c, n_0$ such that

$$|f(n)| \leq c|g(n)| \quad \forall n \geq n_0$$

Therefore, c, n_0 must satisfy the equation

$$\begin{aligned}
81n^3 + 1300n^2 + 300n &\geq c \times (n^5 - 15000n^4 - 10n^3) \\
n^5 &\leq 15000n^4 - 10n^3 + c \times (81n^3 + 1300n^2 + 300n) \\
&\leq \frac{15000}{n} + \frac{10}{n^2} + c \times \left(\frac{81}{n^2} + \frac{1300}{n^3} + \frac{300}{n^4} \right) \\
&\approx 1 + 3.6 \times 10^{-7}c
\end{aligned}$$

From this, the inequality must be satisfied:

$$n \leq \max(n_0, \delta + 1 + 3.6 \times 10^{-7}c)$$

As we see, there is no values of n_0 and c that will make this inequality hold for all n . This has led us to our contradiction. Therefore

$$n^5 - 15000n^4 - 10n^3 \notin \mathcal{O}(um_1)$$

□

Problem #1.2

Problem Statement. Let $f(x) = 2\sin^3(x) - 4\sin^2(x)$ and $g(x) = 2\cos^4(x) + 5\cos(x)$. Determine whether $f(x) \in \mathcal{O}(um_1)$ or $g(x) \in \mathcal{O}(um_1)$. You must show all work and base it directly on the definition of $\mathcal{O}$.

Proof. Because both $\sin x$ and $\cos x$ are sinusoidal, we cannot compare them directly. For the purposes of this problem, we will use a Taylor Series expansions (Equation 1).

$$\sum_{n=1}^{\infty} \frac{f^{(n)}(a)}{n!} (x-a)^n \quad (1)$$

For $f(x)$ and $g(x)$, we get the following expansions.

$$\begin{aligned} \sum_{n=1}^{\infty} \frac{f^{(n)}(a)}{n!} (x-a)^n &= -4x^2 + 2x^3 + \frac{4x^4}{3} - x^5 - \frac{8x^6}{45} + \frac{13x^7}{60} + \frac{4x^8}{315} + \mathcal{O}(um_1) \\ \sum_{n=1}^{\infty} \frac{g^{(n)}(a)}{n!} (x-a)^n &= 7 - \frac{13x^2}{2} + \frac{85x^4}{24} - \frac{1093x^6}{720} + \frac{3329x^8}{8064} - \frac{263173x^{10}}{3628800} + \frac{839681x^{12}}{95800320} + \mathcal{O}(um_1) \end{aligned}$$

From this, we can clearly see that starting at the third term, $g(x)$ grows much more rapidly than $f(x)$. Furthermore, we see that the exponents of $g(x)$ grow by increments of 2, while $f(x)$ only grows by an increments of 1.

Therefore, choosing $c = 1$ and $n_0 = 1$,

$$\begin{aligned} |f(x)| \leq c|g(x)| \quad \forall n \geq n_0 &\implies f(x) \in \mathcal{O}(um_1) \\ &\implies 2\sin^3(x) - 4\sin^2(x) \in \mathcal{O}(um_1) \end{aligned}$$

□

Question #2

For the proceeding problems, the source code is as follows.

Question #3

Problem #3.1

Problem Statement. *Let function q be as below:*

```
def q(n):  
    if n <= 0:  
        return 1  
    elif n < 2:  
        return 7  
    else:  
        return q(n - 1) + q(n - 2)
```

Let function sq be as below:

```
def sq(n):  
    if n < 0:  
        return 0  
    else:  
        return sq(n - 1) + q(n)
```

Conjecture a very simple linear relationship between q and sq .

After careful inspection of the sequences, it became quite apparent that there was a relationship between the two sequences is a constant and two terms in the sequence. In other words,

$$s(q) = q(n + 2) - 7 \tag{2}$$

These findings can be summarized by Table [1](#).

Table 1: The values of $q(n)$, $sq(n)$, and $q(n+2) - 7$

$q(n)$	$sq(n)$	$q(n+2) - 7$
1	1	1
7	8	8
8	16	16
15	31	31
23	54	54
38	92	92
61	153	153
99	252	252
160	412	412
259	671	671
419	1090	1090
678	1768	1768
1097	2865	2865
1775	4640	4640
2872	7512	7512

Problem #3.2

Problem Statement. *Prove, using induction, that the relationship that you conjectured in the previous part is correct. Be sure to set your proof up correctly and to list explicitly the steps of a proof by induction.*

Proof. We will prove the following with induction. To prove this, first we must take into consideration that:

$$sq(n) = \sum_{i=0}^{n+1} q(n) = q(n+2) - 7 \quad (3)$$

Define The Problem For this problem, we wish to prove that $p \stackrel{um_1}{\sim} q$ by the relation modeled by Equation 2. We map this relationship $\forall n \in \mathbb{Z}^+$.

Check Base Case Two Other Values Refer to Table 1 for the first three values, along with 12 more values.

Prove for all $n > s$, that if $P(n-1)$ is true, then $P(n)$ is true Assume the following inductive hypothesis:

$$\sum_{i=0}^{n-1} q(n) = q(n+1) - 7$$

Then we are going to prove

$$\sum_{i=0}^n q(n) = q(n+2) - 7$$

We prove so by such:

$$\begin{aligned} \sum_{i=0}^n q(n) &= \sum_{i=0}^{n-1} q(n) + q(n) \\ &= (q(n+1) - 7) + q(n) \\ &= q(n+2) - 7 \end{aligned}$$

Conclude The proof Because we have proved the base case and the inductive step, we use induction to conclude $sq(n) = q(n+2) - 7$.

□

Question #4

Problem Statement. Let Vec_n be the set of all vectors of length n each component of which comes from $range(n)$. For example, $(3, 0, 2, 2) \in Vec_4$. If $v \in Vec_n$, a quirk is defined as a pair (i, j) such that $0 \leq i < j \leq n - 1$, but $v[i] > v[j]$.

Problem #4.1

Problem Statement. Create a sample space consisting of the elements of Vec_3 . List all the elements of this space. Turn it into a probability space by using the uniform distribution. Finally, compute the average number of quirks in members of Vec_3 .

The sample space S as follows,

$$S = (0, 0, 0), (0, 0, 1), (0, 0, 2), (0, 0, 3),$$
$$(0, 1, 0), (0, 1, 1), (0, 1, 2), (0, 1, 3), (0, 2, 0),$$
$$(0, 2, 1), (0, 2, 2), (0, 2, 3), (0, 3, 0), (0, 3, 1),$$
$$(0, 3, 2), (0, 3, 3), (1, 0, 0), (1, 0, 1), (1, 0, 2),$$
$$(1, 0, 3), (1, 1, 0), (1, 1, 1), (1, 1, 2), (1, 1, 3),$$
$$(1, 2, 0), (1, 2, 1), (1, 2, 2), (1, 2, 3), (1, 3, 0),$$
$$(1, 3, 1), (1, 3, 2), (1, 3, 3), (2, 0, 0), (2, 0, 1),$$
$$(2, 0, 2), (2, 0, 3), (2, 1, 0), (2, 1, 1), (2, 1, 2),$$
$$(2, 1, 3), (2, 2, 0), (2, 2, 1), (2, 2, 2), (2, 2, 3),$$
$$(2, 3, 0), (2, 3, 1), (2, 3, 2), (2, 3, 3), (3, 0, 0),$$
$$(3, 0, 1), (3, 0, 2), (3, 0, 3), (3, 1, 0), (3, 1, 1),$$
$$(3, 1, 2), (3, 1, 3), (3, 2, 0), (3, 2, 1), (3, 2, 2),$$
$$(3, 2, 3), (3, 3, 0), (3, 3, 1), (3, 3, 2), (3, 3, 3)$$

The average number of quirks for $Vec_3 = 1$.

Problem #4.2

Problem Statement. *Generalize the results of Part (a) to determine the average number of quirks in members of Vec_n . You do not need to list the elements of Vec_n .*

For Vec_n , we have the following:

$$\text{Number of } quirks \in Vec_n = 0.25(n-1)^2$$

Question #5

Problem Statement. Let G be defined by the following equations: $G(0) = 5$, $G(1) = 15$, $G(2) = 40$, and for $n > 2$, $G(n) = G(n-1) + G(n-2) + G(n-3)$. Write a Python program that implements G directly from the definition. Submit this program as a `.py` in your ZIP file. Try to compute $G(500)$ with this program. If you can't compute $G(500)$ directly show how to use dynamic programming to write a more efficient program. Submit this program in your ZIP file.

Let H be defined by the following equations: $H(0) = 6$, $H(1) = 7$, $H(2) = 8$, and for $n > 2$, $H(n) = H(n-1) - H(n-2) + H(n-3)$. Write a Python program that implements H directly from the definition. Submit this program as a `.py` in your ZIP file. Try to compute $H(500)$ with this program. If you can't compute $H(500)$ directly show how to use dynamic programming to write a more efficient program. Once you compute $H(500)$ see if you can discover a more efficient program. Submit all of these programs in your ZIP file.

For the sample input, the following is generated.

```
G(500) = 213 546 417 395 738 934 772 794 111 784 493 777 375 698 990 926 394 537 758 958 705 709
        75 006 915 873 346 065 610 819 405 691 957 713 899 246 806 099 708 361 322 154 885 470 915
H(500) = 6
```

For a $\mathcal{O}(num_1)$ solution to $H(n)$, the following was produced:

$$H(n) = \begin{cases} 6 & \iff n \bmod 4 = 0 \\ 7 & \iff n \bmod 2 \neq 0 \\ 8 & \iff (n-2) \bmod 4 = 0 \end{cases}$$

Question #6

Problem Statement. Suppose you are dealing with a dynamic table that follows the following rules:

1. The table size doubles when the table is full and another element is added.
2. The table contracts to $2/3$ of its size when its load factor falls below $1/3$.

Using the potential function

$$\Phi(T) = |2 \times T.num - T.size|$$

show that the amortized cost of a TABLE-DELETE for this strategy is bounded above by a constant. For the definition of all terms, consult your textbook.

Proof. We know the amortized cost of the i th operation to be

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} \tag{4}$$

Assuming the i th operation does not trigger a contraction, using Equation 4,

$$\begin{aligned} \hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= 1 + (2 \times T.num_i - T.size_i) - (2 \times T.num_{i-1} - T.size_{i-1}) \\ &= 1 + (2 \times T.num_i - T.size_i) - (2 \times T.num_i - T.size_i) + 2 \\ &= 3 \end{aligned}$$

However, if the i th operation does trigger a contraction,

$$\begin{aligned} \hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= T.num_i + 1 + (2 \times T.num_i - T.size_i) - (2 \times T.num_{i-1} - T.size_{i-1}) \\ &= T.num_i + 1 + \left(\frac{2}{3} T.size_{i-1} - 2 \times T.num_i \right) - (T.size_{i-1} - 2 \times T.num_i - 2) \\ &= 2 \end{aligned}$$

We see in either situations, the amortized cost of a TABLE-DELETE operation is bounded by $2 \leq \hat{c}_i \leq 3$.

□

Question #7

Problem #7.1

Problem Statement. Consider the following two sets of coin denominations: $\{1 \text{ cent}, 8 \text{ cents}, 20 \text{ cents}\}$, $\{1 \text{ cent}, 6 \text{ cents}, 18 \text{ cents}\}$. Describe a greedy algorithm that will express any sum given in pennies in terms of the denominations given in a set. Determine whether the greedy algorithm always produces the optimal solution for these two sets or not. Give a convincing reason for your conclusion.

A greedy algorithm that always produces a solution could be described as follows:

1. Take $S = \emptyset$ to be the solution set, *coins* to be the input of coins, and T to be the target to reach.
2. Sort *coins* in ascending order.
3. While $\sum_{x \in S} x \neq T$
 - a) Take the difference δ to be $T - \sum_{x \in S} x$.
 - b) Pick the largest coin c such that $c \leq \delta$ ¹.
 - c) Add this coin to S .

Although this always produces a solution, it does not always produce an optimal solution. For example, suppose our target $T = 24$. With the algorithm described above, then it would take 5 coins ($1 \times 20\text{¢} + 4 \times 1\text{¢}$), while the optimal solution would take 4 coins ($4 \times 8\text{¢}$).

¹Because 1 is a valid coin, there will always be a coin to choose from

Problem #7.2

Problem Statement. *Suppose we have an unlimited number of rooms and a finite number of activities, each of which can be staged in any of the rooms. Give an efficient algorithm that can schedule all the activities using the smallest number of rooms.*

A greedy algorithm that always produces solution is as follows:

1. Sort all of the activities in respect to their finish times (i.e., the job that finishes first is the first element). Call this sorted set of activities A .
2. Take a particular room that is not preoccupied R_i and map a particular schedule S_i to it.
3. While not at the end of A :
 - a) Pick the first activity as just the first element in the A . Remove this element from A and add it the schedule S_i .
 - b) Search A from beginning, finding the first activity who's start time is after the finish time of the previous job. Add this to schedule S_i , remove it from A .
 - c) Repeat the last step.
4. If A is empty, the algorithm is finished. If not, repeat Step 2.

Question #8

Problem #8.1

Problem Statement. *Consider the following problem: given a graph, determine whether it can be colored using exactly 4 colors. Prove that this coloring problem is NP-Complete.*

Proof. To prove the Four Coloring Problem is NP-Complete, first we must prove that the Four Color Problem is NP. To check a solution in polynomial time, we simply iterate through all edges, and check:

- Make sure that the graph is colored by ≤ 4 colors.
- Iterate through all edges, ensuring that every edge's neighbor is colored by a different color.

Because this is a $\mathcal{O}(nm_1)$ algorithm, we can check a solution in polynomial time.

To prove that this is NP-Complete, we will reduce it to the three coloring problem, as follows.

Supposing we have a graph G , we wish to create a new graph G' , such that if G is colorable in three colors, G' can be colored in four. To make G' , then we simply add a new vertex and attach it to all of the vertices in G . Therefore, if G is three colorable, then we know G' to be four colorable.

Because we know the Four Coloring Problem is NP and is reducible from the Three Coloring Problem, we conclude that the Four Coloring Problem is NP-Complete. $\square$

Problem #8.2

Problem Statement. *Prove that for all $k > 4$, determining whether a graph can be colored using exactly k colors is NP-Complete.*

Proof. To prove the Four Coloring Problem is NP-Complete, first we must prove that the k -Color Problem is NP. To check a solution in polynomial time, we simply iterate through all edges, and check:

- Make sure that the graph is colored by $\leq k$ colors.
- Iterate through all edges, ensuring that every edge's neighbor is colored by a different color.

Because this is a $\mathcal{O}(nm_1)$ algorithm, we can check a solution in polynomial time.

To prove that this is NP-Complete, we will reduce it to the three coloring problem, as follows.

Supposing we have a graph G , we wish to create a new graph G' , such that if G is colorable in three colors, G' can be colored in k . To make G' , then we simply add attach k new vertices and attach it to all of the vertices in G . Therefore, if G is three colorable, then we know G' to be k colorable.

Because we know the k -Coloring Problem is NP and is reducible from the Three Coloring Problem, we conclude that the k -Coloring Problem is NP-Complete.

□

CS5400

Artificial Intelligence



Exercise #1

Introduction To Artificial Intelligence

Illya Starikov

Due Date: January 22nd, 2018

We defined a rational agent as follows:

For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

We also have the following assumptions:

1. The performance measure awards one point for each clean square at each time step, over a “lifetime” of 1000 time steps.
2. The “geography” of the environment is known a priori (Figure 2.2) but the dirt distribution and the initial location of the agent are not. Clean squares stay clean and sucking cleans the current square. The Left and Right actions move the agent left and right except when this would take the agent outside the environment, in which case the agent remains where it is.
3. The only available actions are Left, Right, and Suck.
4. The agent correctly perceives its location and whether that location contains dirt.

Problem Statement. *Show that the simple vacuum-cleaner agent function described in Figure 2.3 is indeed rational under the assumptions listed on Page 38.*

We couple our assumptions with the fact the agent moves to the adjoining square if there is no dirt, and “Suck”s if there is dirt. Also, we are under the assumption that the environment cannot become dirtier from some external agent within the 1000 time steps.

Under these assumptions, with our performance measure (see Item 1), we can conclude the agent is rational. Because the performance measure cannot decrease, it would always be beneficial for the agent to move to a new square to clean (since there is always a

possibility of increasing the performance measure). By design, our agent always moves to a new square, consequently, always having the possibility of increasing the performance measure.

Problem Statement. *Describe a rational agent function for the case in which each movement costs one point. Does the corresponding agent program require internal state?*

Yes. Because movement now costs one point, it is no longer in the agent's best interest to constantly move. However, because our agent is purely reflexive and we are unsure of the initial conditions, the agent will receive a negative performance measure; quite contrary to our definition of a rational agent.

For our agent to achieve maximum performance measure, the optimal situation would be to keep all clean squares in some sort of memory, and when the entirety of the vacuum world is clean, constantly perform a *NoOp*; a reflexive agent cannot do this.

Problem Statement. *Discuss possible agent designs for the cases in which clean squares can become dirty and the geography of the environment is unknown. Does it make sense for the agent to learn from its experience in these cases? If so, what should it learn? If not, why not?*

Learning would be optimal in this situation, for the following reasons:

Boundaries An agent with no clue about the world boundaries is doomed to constantly bump in the walls. This would waste precious time cycles on simply finding the boundaries.

Obstacles Just as one might bump in the boundaries, an agent might bump into obstacles.

Common Accumulation Some areas might become dirtier faster than others (i.e., a doorway). If an agent could determine where these areas are, it can check those areas more often than areas with less dirt.

In essence, learning some sort of a map would greatly improve the performance measure of the agent.

After learning some sort of map, we use the following agent design:

1. Start traversing to the most common dirty areas on the map in order of closeness.
2. With every step, if the current step is dirty, clean it; if not, *NoOp*.
3. If all common areas are **not** traversed, proceed to Step 1.
4. Proceed to the closest edge of the world boundary.
5. Proceed around the edges of the map.
6. If **not** reached original position, repeat Step 5.

7. Move one unit closer to the inside of the world. Treat the previous movement as the new world boundaries.
8. If **not** reached center of world, repeat Step 5.
9. Proceed to Step 1.

Exercise #2

Introduction To Artificial Intelligence

Illya Starikov

Due Date: January 24th, 2018

For these problems, we split the river into two sections into two section: Left and Right. Left is assumed to be the initial state.

1 Problem Formulation

Problem Statement. *Precisely formulate the problem.*

1. A method of describing a state would be with the tuple: $(C_L, M_L, B_{L/R})$, where:

C_L = The number of Cannibals on the Left

M_L = The number of Missionaries on the Left

$B_{L/R}$ = The location (Left/Right) of the boat

However, we restrict these numbers to as follows:

We do not care about the number of Cannibals/Missionaries on the right, because we can calculate them as follows:

$$C_R = 3 - C_L \quad M_R = 3 - M_L$$

Although these values are not strictly within the state description, from here forth (for readability purposes) we act as if it is, mentally replacing C_R and M_R with these values. The final description is still $(C_L, M_L, B_{L/R})$, and these are merely computed properties of the state.

Note, however, because we want to restrict our state space to only valid states, we place the following restrictions:

$$M_L \geq C_L \quad \text{and} \quad M_R \geq C_R$$

Furthermore,

$$B_{L/R} = \text{Left} \iff M_L + B_L = 6 \quad B_{L/R} = \text{Right} \iff M_R + B_R = 6$$

2. The initial state is as follows:

$$(C_L, M_L, B_{L/R}) = (3, 3, L)$$

3. Our ACTIONS are defined as follow. We use synonyms $L \leftrightarrow$ Left and $R \leftrightarrow$ Right.

Basically, we check to see if we can send one of both **Cannibal** and **Missionary**. If we can, add them to the solution set. Next, we check to see if we can send either 1 **Cannibal** and 1 **Missionary**. After, we check to see if we can send 2 **Cannibal** and 2 **Missionary**. Because there are rules we have to account for (i.e., there must be more **Missionary** than **Cannibal** *unless there's 0 Missionary*).

```

1: function  $\neg$ (direction)
2:   if direction = Left then
3:     return Right
4:   else
5:     return Left
6:   end if
7: end function
8:
9:
10: function ACTIONS( $s$ )
11:   PossibleActions  $\leftarrow \{\}$  ▷ A set of sets of moves
12:
13:   for all direction  $d \in \{\text{Left}, \text{Right}\}$  do
14:     if  $a.M_d > 0$  and  $a.C_d > 0$  then
15:       NewAction  $\leftarrow \{\text{send } \text{Cannibal } d \rightarrow \neg d\}$ 
16:       NewAction  $\leftarrow \{\text{send } \text{Missionary } d \rightarrow \neg d\}$  ▷ Note this has the form
17:        $\{\dots, \dots\}$ 
18:       PossibleActions  $\leftarrow \text{PossibleActions} \cup \{\text{NewAction}\}$  ▷ Note this has the
19:       form  $\{\dots, \{\dots, \dots\}, \dots\}$ 
20:     end if
21:
22:     for  $i \leftarrow 1$  to 2 do
23:       if  $a.M_d - i \geq 0$  then ▷ Is there even enough missionaries to remove?
24:         if  $(a.M_d - i \geq a.C_d$  or  $a.M_d - i = 0)$  and  $a.M_{\neg d} + i \geq a.C_{\neg d}$  then
25:           NewAction  $\leftarrow \{\text{send } i \text{ Missionary } d \rightarrow \neg d\}$ 1
26:           PossibleActions  $\leftarrow \text{PossibleActions} \cup \{\text{NewAction}\}$ 
27:         end if
28:       end if

```

¹Note by saying “send i **Missionary**”, we are just saying “send a missionary” i times. For $i = 2$, we have NewAction $\leftarrow \{\text{send a missionary}, \text{send a missionary}\}$.


```

28:
29:         if  $a.C_d - i \geq 0$  then           ▷ Is there even enough cannibals to remove?
30:             if  $a.C_{\neg d} + i \leq a.M_{\neg d}$  or  $a.M_{\neg d} = 0$  then
31:                 NewAction  $\leftarrow \{\text{send } i \text{ Cannibal } d \rightarrow \neg d\}$ 2
32:                 PossibleActions  $\leftarrow$  PossibleActions  $\cup \{\text{NewAction}\}$ 
33:             end if
34:         end if
35:     end for
36: end for
37:
38:     return PossibleActions
39: end function

```

4. We define RESULTS(s, a) as follows.

```

1: function RESULTS( $s, a$ )
2:      $a' \leftarrow a$ 
3:      $a'.B_{L/R} = \neg a'.B_{L/R}$ 
4:
5:     for all action  $\in a$  do           ▷ We defined an action as a set of steps
6:         if action = send Cannibal Left  $\rightarrow$  Right then
7:              $a'.C_L = a'.C_L - 1$ 
8:         end if
9:
10:        if action = send Cannibal Right  $\rightarrow$  Left then
11:             $a'.C_L = a'.C_L + 1$ 
12:        end if
13:
14:        if action = send Missionary Left  $\rightarrow$  Right then
15:             $a'.M_L = a'.M_L - 1$ 
16:        end if
17:
18:        if action = send Missionary Right  $\rightarrow$  Left then
19:             $a'.M_L = a'.M_L + 1$ 
20:        end if
21:    end for
22:
23:    return  $a'$ 
24: end function

```

5. We define ISGOAL(s) as follows.

```

1: function ISGOAL( $s$ )

```

²See previous footnote.

```
2:   return  $s = (0, 0, \text{Right})$ 
3: end function
```

6. We define $C(s, a, s')$

```
1: function  $C(s, a, s')$ 
2:   return  $|a|$ 3
3: end function
```

2 State Space

Figure 1 provides the complete space for the problem, with **M** representing Missionaries and **C** representing cannibals.

3 Possible Solution

Figure 2 outlines a path for a solution, with **M** representing Missionaries and **C** representing cannibals.

³We define the $|x|$ operator to return the length of x .

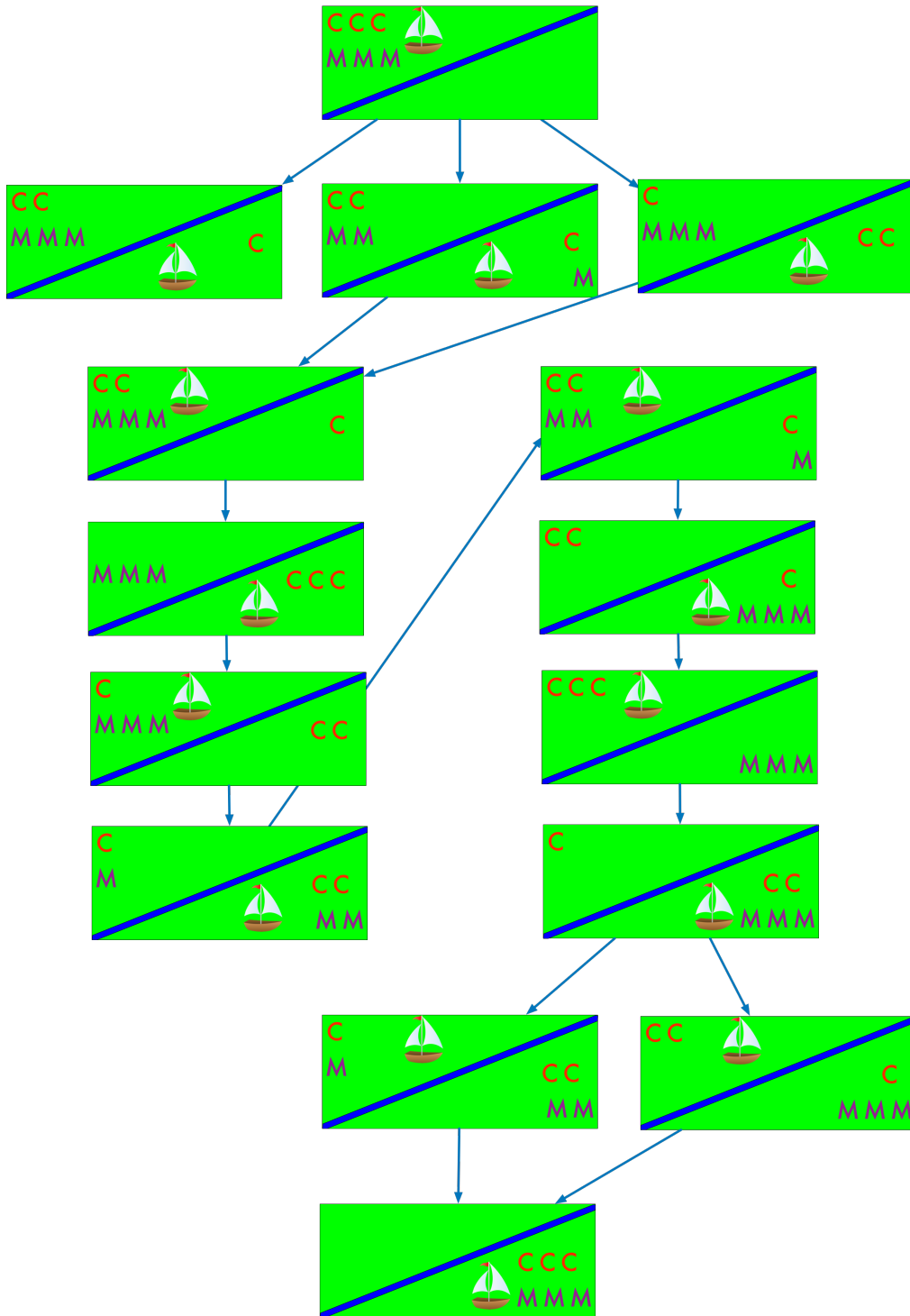


Figure 1: The Complete State Space

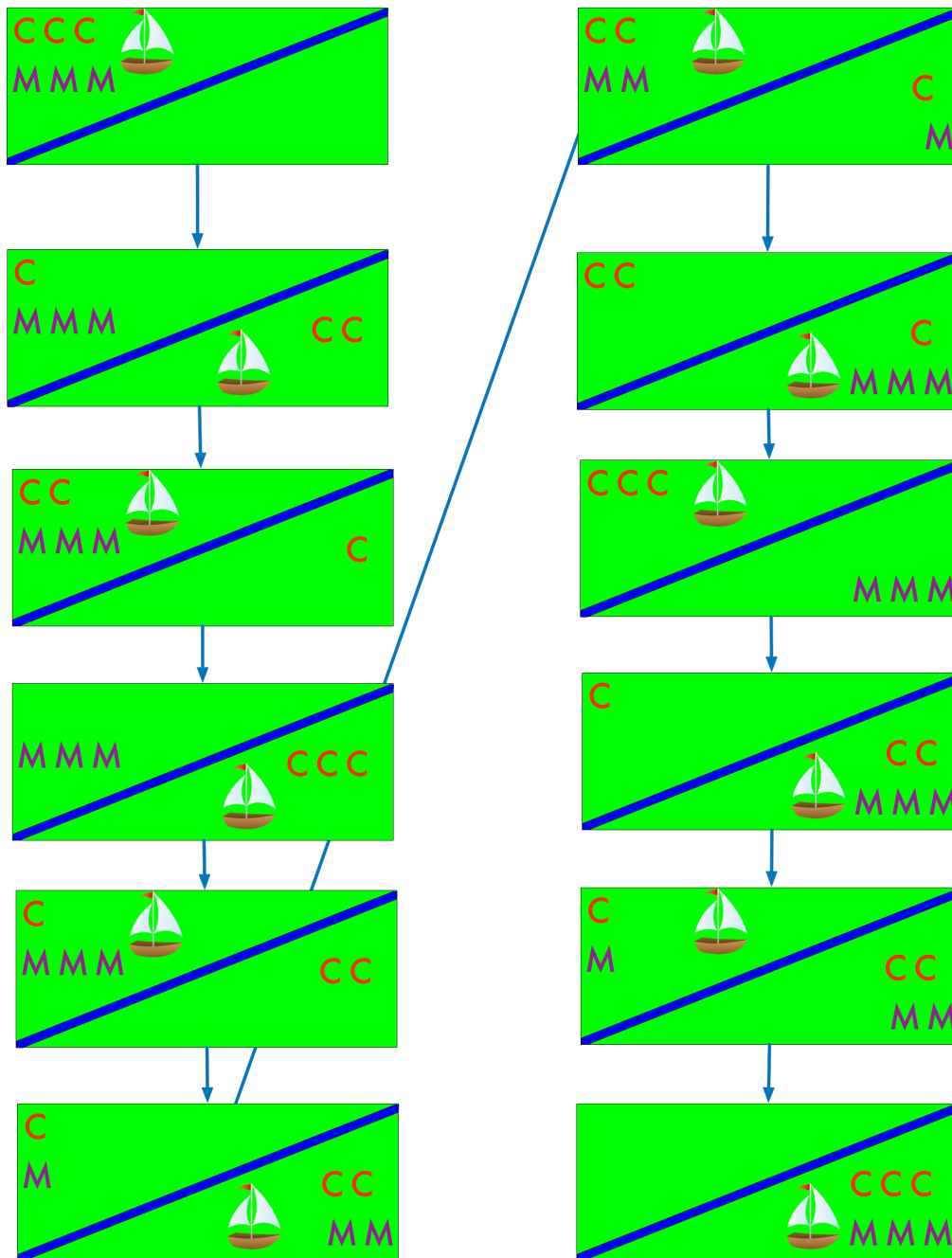


Figure 2: Path To A Solution

Exercise #3

Introduction To Artificial Intelligence

Illya Starikov

Due Date: February 5th, 2018

1 Negative Path Optimality

Problem Statement. *Suppose that actions can have arbitrarily large negative costs; explain why this possibility would force any optimal algorithm to explore the entire state space.*

Even if for any path we have an arbitrary path $P_{\max}$ with largest path cost $N_{\max}$,

$$N_{\max} = \sum_{\text{vertex } v \in P_{\max}} \text{PATH-COST}(v)$$

There might be a singular path P after it,

$$N = \sum_{\text{vertex } v \in P} \text{PATH-COST}(v)$$

Such that

$$N_{\max} + N = \text{Minimum Path Cost}$$

Therefore, any algorithm that is to find the optimal path cost must search the entire state space.

Also, there is the possibility of a negative cycle in a graph, which can only be discovered by exploring the entire state space.

2 Bounds on Negative Path Cost

Problem Statement. *Does it help if we insist that step costs must be greater than or equal to some negative constant c ? Consider both trees and graphs.*

For trees, it can be helpful in limiting the search space, because there can only be so much gained from negative paths.

For graphs, there is the aforementioned issue of negative cycles, where the lowest path cost could be $-\infty$ (i.e., keep repeating the negative cycle infinitely).

3 Loops In Negative Path Cost

Problem Statement. *Suppose that a set of actions forms a loop in the state space such that executing the set in some order results in no net change to the state. If all of these actions have negative cost, what does this imply about the optimal behavior for an agent in such an environment?*

This implies that the optimal behavior in this environment is to loop around the states in the negative cycle.

Exercise #4

Introduction To Artificial Intelligence

Illya Starikov

Due Date: February 12th, 2018

Problem Statement. *On page 90, we mentioned iterative lengthening search, an iterative analog of uniform cost search. The idea is to use increasing limits on path cost. If a node is generated whose path cost exceeds the current limit, it is immediately discarded. For each new iteration, the limit is set to the lowest path cost of any node discarded in the previous iteration.*

1 Iterative Lengthening Optimality

Problem Statement. *Show that this algorithm is optimal for general path costs.*

Because the nodes are discovered in order of path cost, naturally the node with the lowest path cost will be discovered first.

2 Iterative Lengthening Performance

Problem Statement. *Consider a uniform tree with branching factor b , solution depth d , and unit step costs. How many iterations will iterative lengthening require?*

The runtime will be $\mathcal{O}(b^d)$.

3 Iterative Lengthening Performance II

Problem Statement. *Now consider step costs drawn from the continuous range $[\epsilon, 1]$, where $0 < \epsilon < 1$. How many iterations are required in the worst case?*

The runtime will be $\mathcal{O}\left(\frac{d}{\epsilon}\right)$.

Exercise #5

Introduction To Artificial Intelligence

Illya Starikov

Due Date: February 19th, 2018

1 A^* vs. DFS

Problem Statement. *True/False: Depth-first search always expands at least as many nodes as A search with an admissible heuristic.*

False. DFS has the best case of $\Theta(d)$ (for the case where there is no backtracking to find a solution). A^* can make no such guarantee, because it ranges by heuristic.

2 $h(n) = 0$ For 8-Puzzle

Problem Statement. *True/False: $h(n) = 0$ is an admissible heuristic for the 8-puzzle.*

True. For positive path costs, $h(n) = 0$ will always be an admissible heuristic.

$$\forall \epsilon \in \mathbb{R}^+, h(n) = 0 \leq C^*(n) = \epsilon$$

3 A^* In Robotics

Problem Statement. *True/False: A is of no use in robotics because percepts, states, and actions are continuous.*

True. A^* operates on discrete states. However, there is always the possibility to create discrete states from continuous states.

4 Breadth-First Search Completeness

Problem Statement. *True/False: Breadth-first search is complete even if zero step costs are allowed.*

True. Per our definition of completeness:

A search algorithm is complete iff it will find a goal when one exists.

5 Rook Admissibility

Problem Statement. *True/False: Assume that a rook can move on a chessboard any number of squares in a straight line, vertically or horizontally, but cannot jump over other pieces. Manhattan distance is an admissible heuristic for the problem of moving the rook from square A to square B in the smallest number of moves.*

False. Using the chess depicted in Figure 1, the Manhattan Distance from A1 to A8 would be 8; however, a rook could get there in one move.

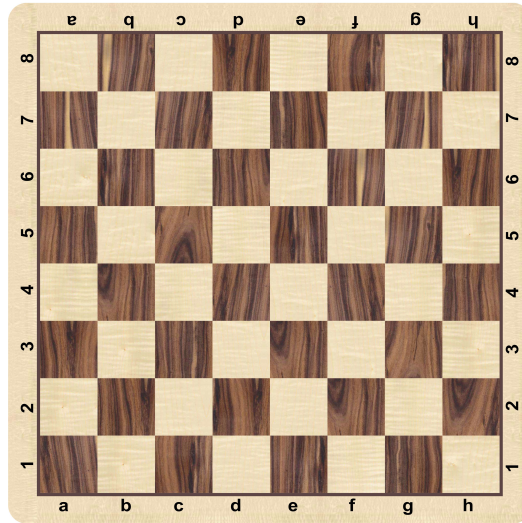


Figure 1: A Standard Chessboard

Exercise #6

Introduction To Artificial Intelligence

Illya Starikov

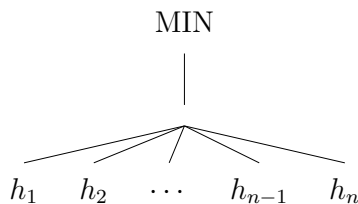
Due Date: March 13th, 2018

1 Suboptimal Min vs Optimal Min

Problem Statement. *Prove the following assertion: For every game tree, the utility obtained by MAX using minimax decisions against a suboptimal MIN will be never be lower than the utility obtained playing against an optimal MIN. Can you come up with a game tree in which MAX can do still better using a suboptimal strategy against a suboptimal MIN?*

We will do so by induction.

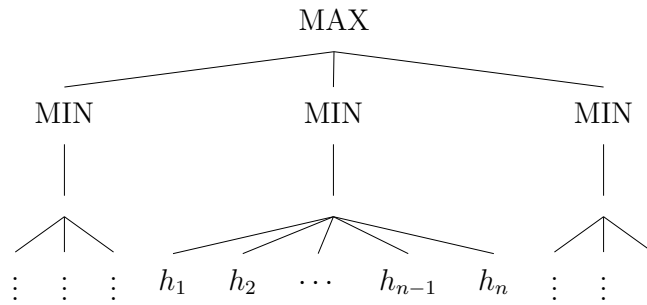
Basis Case The base case is a *MIN* node with all terminals. Suppose we have the following leaf such that for any node h_x , they are ordered in ascending order based on the utility value $h_1 \leq h_2 \leq \dots \leq h_{n-1} \leq h_n$:



An optimal *MIN* would always pick h_1 , while a suboptimal would pick from $\{h_1, h_2, \dots, h_{n-1}, h_n\}$. From this, we can conclude

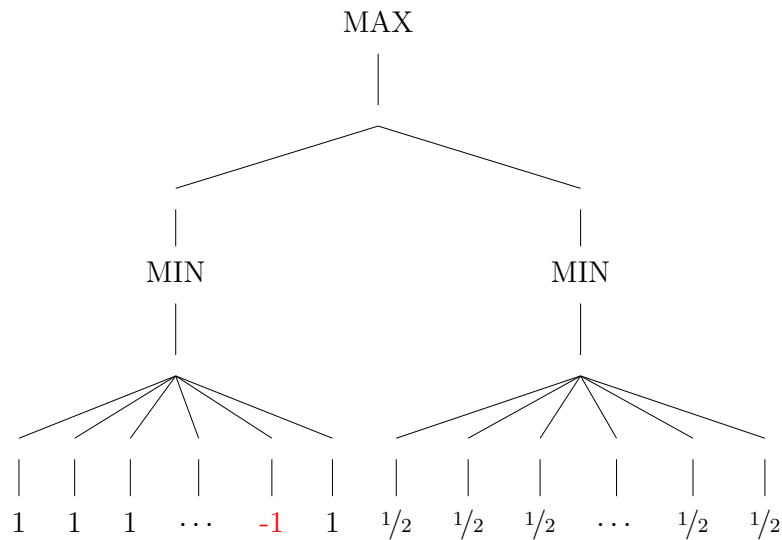
$$\text{Optimal } MIN\text{'s Utility} \leq \text{Suboptimal } MIN \text{ Utility} \quad (1)$$

Inductive Case The recursive case is for *MIN* with parents and children.



However, because the game tree can be reduced to a *MIN* node with all terminals (such as the base case), our inductive step holds.

For a game tree of a suboptimal *MAX* vs a *MIN*, we can produce the following chess game:



An optimal *MAX* would pick the rightmost *MIN*, which would result in only a draw; however, a suboptimal *MIN* would only pick the leftmost *MIN*, resulting in in almost all wins, with only one possibility of losing.

Exercise #7

Introduction To Artificial Intelligence

Illya Starikov

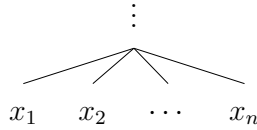
Due Date: March 9th, 2018

1 x To $ax + b$ Leaf Node Transformation

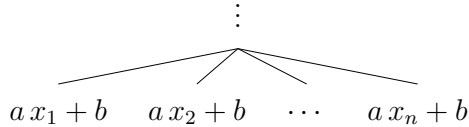
Problem Statement. *Prove that with a positive linear transformation of leaf values (i.e., transforming a value x to $ax + b$ where $a > 0$), the choice of move remains unchanged in a game tree, even when there are chance nodes.*

Proof. We will prove the following via proof of mathematical induction.

Base Case We will take the base case with a lead nodes, so the transformation would be as follows:



would transform into:



Recalling the definition of MINIMAX, we get the following:

$$\text{MINIMAX}(s) = \begin{cases} \text{MAX'S UTILITY}(s) & \leftrightarrow \text{TERMINAL-TEST}(s) \\ \max_{a \in \text{ACTIONS}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \leftrightarrow \text{PLAYER}(s) = \text{MAX} \\ \min_{a \in \text{ACTIONS}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \leftrightarrow \text{PLAYER}(s) = \text{MIN} \end{cases} \quad (1)$$

Recalling that ACTIONS returns a set of values (let us name them $x_1, x_2, \dots, x_n$) our min () and max () values can be rewritten as:

$$\begin{aligned} \min(x_1, x_2, \dots, x_n) &= x_p \\ \max(x_1, x_2, \dots, x_n) &= x_p \end{aligned}$$

where x_p is the chosen node. For the probability, we get

$$p(x_1) + p(x_2) + \cdots + p(x_n)$$

Applying the transformation, we get the following:

$$\begin{aligned} \min(a x_1 + b, a x_2 + b, \dots, a x_n + b) &= a x_p + b \\ \max(a x_1 + b, a x_2 + b, \dots, a x_n + b) &= a x_p + b \\ p_1(a x_1 + b) + p_2(a x_2 + b) + \cdots + p_n(a x_n + b) \end{aligned}$$

From this, we factor out a and b :

$$\begin{aligned} a \min(x_1, x_2, \dots, x_n) + b &= a x_p + b \\ a \max(x_1, x_2, \dots, x_n) + b &= a x_p + b \\ a(p_1 x_1 + p_2 x_2 + \cdots + p_n x_n) + b \end{aligned}$$

Upon which we can clearly see a ($\forall a \in \mathbb{R}^+$) and b have no choice on which node gets chosen.

Inductive Step As we scale our problem up for MINIMAX (Equation 1), we see that if the decision of the terminal node is unchanged, the decision at the root will be changed. The same terminal value will be returned; and as we go up the game tree, each ply acts as a terminal.

□

Exercise #8

Introduction To Artificial Intelligence

Illya Starikov

Due Date: April 22nd, 2018

1 State-Space Search Algorithms

Problem Statement. *Give the name of the algorithm that results from each of the following special cases.*

1. Local beam search with $k = 1$. **Steepest-Ascent Hill-Climbing.**
2. Local beam search with one initial state and no limit on the number of states retained. **Breadth First Tree Search.**
3. Simulated annealing with $T = 0$ at all times (and omitting the termination test). **Steepest-Ascent Hill-Climbing.**
4. Simulated annealing with $T = \infty$ at all times. **Random Walk.**
5. Genetic algorithm with population size $N = 1$. **Random Walk.**

Bonus #1

Illya Starikov

Due Date: Sunday, February 18th, 2018

We know the definition of branching factor b^* to be defined as

$$N = b^* + (b^*)^2 + \dots + (b^*)^d$$

With N being the total number of nodes being generated, and d being the solution depth. From this, we can get an estimate for the effective branching factor as follows:

$$b^* = N^{\frac{1}{d}}$$

Note this is not a perfect equation, and has the possibility for high error tolerances; for our purposes, it should be okay. The two heuristics for this assignment are as follows:

$$\begin{aligned} h_1 &= \text{Score Difference} \\ &= \frac{\text{Quota} - \text{Current Score}}{\text{Max Swaps} - \text{Current Swaps}} \\ h_2 &= \text{Homogeneous Board} \\ &= \text{Standard Deviation of the} \\ &= \text{Number of Devices Types on the Board} \end{aligned}$$

Results can be found in Table 1. Essentially, the branching factors are similar, but the Homogeneous Board heuristic has a tendency to explore more of the board.

Table 1: The computations for the effective branching factor.

Input	Heuristic	Nodes Generated (N)	Solution Depth (d)	Branching Factor (b^*)
Puzzle #1	Score Difference	4	1	4
	Homogeneous Board	4	1	4
Puzzle #2	Score Difference	99	11	1.52
	Homogeneous Board	144	10	1.64
Puzzle #3	Score Difference	94	12	1.46
	Homogeneous Board	890	25	1.31

CS5402

Data Mining



Test #1

CS5402 — Intro To Data Mining

Illya Starikov

Due Date: July 15th, 2018

Multiple Choice

1. e. None of the above
2. c. Remove any attribute that has missing values.
3. b. $\frac{1}{2}$
4. b. wt
5. d. Spearman's rank correlation coefficient
6. c. Healthland
7. b. slice for Time = Q1
8. d. roll up on Location = Beijing or Tokyo (i.e., from city to country)
9. c. drill down on Time = Q1 (i.e., from quarter to month)
10. a. dice for (location = Beijing or Tokyo) and (product = Chain or bracelet) and (time = Q1 or Q2)

11 Short Answer

Method #1 is the most accurate, because the true positive (y -axis) correctly identified the values, while the false positive (x -axis) incorrectly identified the values. Method #1 had the fastest growing function (with respect to y).

12 1-R Method

Attribute	Attribute Value	# Rows With Attribute Value	Most Frequent Value For sportPref	Errors	Total Errors
ageGroup	youngAdult	3	football (2)	1	3
	middleAge	3	football/hockey/baseball (1/1/1)	2	
	senior	2	baseball (2)	0	
gender	M	5	baseball/football (2/2)	3	5
	F	3	football/hockey/baseball (1/1/1)	2	
petPreference	dog	5	football (3)	2	3
	cat	3	baseball (2)	1	

The rules are as follows:

ageGroup = **youngAdult** $\implies$ football
 ageGroup = **middleAge** $\implies$ football
 ageGroup = **senior** $\implies$ baseball

13 Prism

For football, we get the following table:

gender	pet	drink	sport
M	dog	beer	football
F	dog	beer	football

For our P and T values:

	T	P	T/P
gender = M	3	1	1/3
gender = F	4	1	1/4
pet = dog	3	2	2/3
drink = beer	3	2	3/4

Seeing as not T/P values are 1, we must add a clause. We choose pet = dog as the base.

	T	P	T/P
gender = M	1	0	0
gender = F	1	0	0
drink = beer	2	2	1

pet = **dog** and drink = **beer** $\implies$ football

14 Statistical Modeling

The likelihood would be as follows:

$$\text{likelihood} = 4/9 \times 2/9 \times 6/9 \times 3/9 \times 9/14$$

15 Entropy

(a) entropyBeforeSplit would be as follows:

$$-1/6 \log_2 (1/6) - 2/6 \log_2 (2/6) - 3/6 \log_2 (3/6)$$

(b) entropyPoor would be as follows:

$$-2/4 \log_2 (2/4) - 2/4 \log_2 (2/4)$$

(c) infoGain would be determined as follows:

$$\begin{aligned} \text{entropyAfterSplit} &= 3/6 \text{entropyShort} + 2/6 \text{entropyMed} + 1/6 \text{entropyLong} \\ \text{infoGain} &= \text{entropyBeforeSplit} - \text{entropyAfterSplit} \end{aligned}$$

16 Rule Induction

(a) The partitions would be as follows:

$$\begin{aligned} \{d\}^* &= \{\{x_1\}, \{x_2, x_3\}, \{x_5\}, \{x_5\}\} \\ \{e\}^* &= \{\{x_1, x_2, x_5\}, \{x_3, x_4\}\} \\ \{d, e\}^* &= \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}\} \end{aligned}$$

(b) The coverings are as follows:

- $\{d\}^*$ would not work, because every block in the partition is not a subset of a block in $\{f\}^*$.
- $\{d, e\}^*$ would work, because every block in the partition is a subset of a block in $\{f\}^*$.
- $\{a, d, e\}^*$ would not work, because although every block in the partition is a subset of a block in $\{f\}^*$, it is not minimal.

(c) The rules would be as follows:

$$d = X \text{ and } e = 4 \implies f = T$$

$$d = S \text{ and } e = 4 \implies f = T$$

$$d = S \text{ and } e = 3 \implies f = F$$

$$d = H \text{ and } e = 3 \implies f = F$$

$$d = M \text{ and } e = 4 \implies f = F$$

17 KD-Tree

Sorting, we get the following: $[(2, 10), (4, 20), (6, 10), (8, 20), (10, 30)]$.
With a median of 6...

- $x < 6$ group: $[(2, 10), (4, 20)]$
- $x \geq 6$ group: $[(6, 10), (8, 20), (10, 30)]$

Sorting, we get the following: $[(2, 10), (4, 20)] [(6, 10), (8, 20), (10, 30)]$
With a median of 15 for the first group:

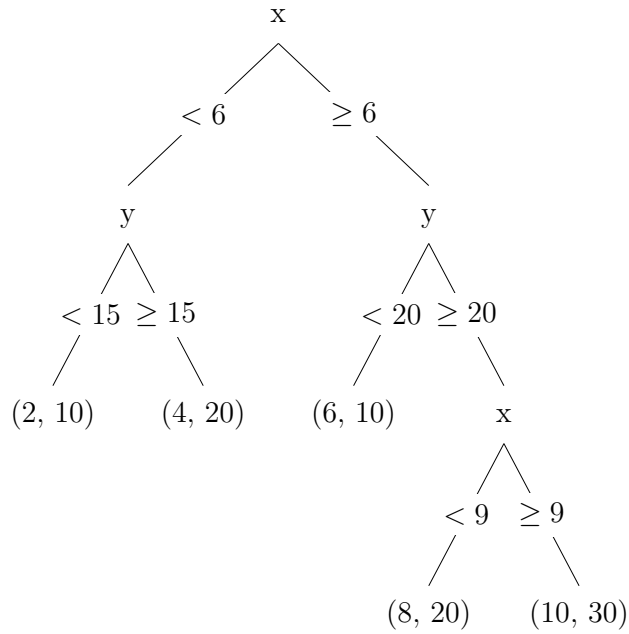
- $y < 15$ group: $[(2, 10)]$
- $y \geq 15$ group: $[(4, 20)]$

With a median of 20 for the second group:

- $y < 20$ group: $(6, 10)$
- $y \geq 20$ group: $[(8, 20), (10, 30)]$

(Using a shortcut for the final block), Sorting, and using a median of 9, our last block looks like as follows:

- $x < 9$ group: $[(8, 20)]$
- $y \geq 9$ group: $[(10, 30)]$



18 Clustering

x	y	distance to (2, 4)	distance to (5, 6)	distance to (8, 1)
2	4	0	5	9
5	6	5	0	8
8	1	9	8	0
7	3	6	5	3
4	10	8	5	13
3	0	5	8	6
9	8	11	6	8

Our clusters would be as follows:

Cluster Center (2, 4) (2, 4), (3, 0)

Cluster Center (5, 6) (5, 6), (4, 10), (9, 8)

Cluster Center (8, 1) (8, 1), (7, 3)

With means as follows:

Cluster Mean of (2, 4), (3, 0) $(2.5, 2) \approx (3, 2)$

Cluster Mean of (5, 6), (4, 10), (9, 8) $(6, 8)$

Cluster Center of (8, 1), (7, 3) $(7.5, 2) \approx (8, 2)$

x	y	distance to (3, 2)	distance to (6, 8)	distance to (8, 2)
2	4	3	8	8
5	6	6	3	7
8	1	6	9	1
7	3	5	6	2
4	10	9	4	12
3	0	2	11	7
9	8	12	3	7

Cluster Center (3, 2) (2, 4), (3, 0)

Cluster Center (6, 8) (5, 6), (4, 10), (9, 8)

Cluster Center (8, 2) (8, 1), (7, 3)

Clusters haven't changed! Final cluster centers and instances are as follows:

Cluster Center (3, 2) (2, 4, 11, yes), (3, 0, 3, yes)

Cluster Center (6, 8) (5, 6, 5, no), (4, 10, 8, yes), (9, 8, 1, no)

Cluster Center (8, 2) (8, 1, 7, no), (7, 3, 4, yes)

19 Confusion Table

- (a) For a randomly produced results, there were 8 values that we predicted to be B, when they were actually G.
- (b) For a classifier produced results, there were 30 values that we predicted to be B, and were actually B.
- (c) The non-random classifier, 90 were predicted correctly. For the random classifier, 39 were predicted correctly. Therefore, 51 more were predicted correctly.
- (d) Kappa Statistic would be

$$\frac{\text{Non-Random Correct} - \text{Random Correct}}{\text{Total}}$$

Which would be as follows:

$$\frac{90 - 39}{100}$$

Test #2

CS5402 — Intro To Data Mining

Illya Starikov

Due Date: July 29th, 2018

1 C4.5

$$Split = 120$$

$$Entropy_{LT} = -\frac{3}{4} \log_2 \left(\frac{3}{4} \right) - \frac{1}{4} \log_2 \left(\frac{1}{4} \right)$$

$$Entropy_{Gt} = -\frac{2}{5} \log_2 \left(\frac{2}{5} \right) - \frac{3}{5} \log_2 \left(\frac{3}{5} \right)$$

$$Info\ Gain = X - \left(\frac{4}{9} \times Entropy_{LT} + \frac{5}{9} \times Entropy_{Gt} \right)$$

$$Split = 140$$

$$Entropy_{LT} = -\frac{4}{6} \log_2 \left(\frac{4}{6} \right) - \frac{2}{6} \log_2 \left(\frac{2}{6} \right)$$

$$Entropy_{Gt} = -\frac{2}{3} \log_2 \left(\frac{2}{3} \right) - \frac{1}{3} \log_2 \left(\frac{1}{3} \right)$$

$$Info\ Gain = X - \left(\frac{6}{9} \times Entropy_{LT} + \frac{3}{9} \times Entropy_{Gt} \right)$$

We choose the split that we calculated previously based on it's associated **highest information gain**. Values before it should be assigned *Less Than Split* and values after it should be assigned *Greater Than Split*.

2 Grouping Or Splitting

$$\{\{SciFiction, Mystery\}, \{NonFiction\}\}$$

$$SciMystery = -\frac{3}{6} \log_2 \left(\frac{3}{6} \right) - \frac{2}{6} \log_2 \left(\frac{2}{6} \right) - \frac{1}{6} \log_2 \left(\frac{1}{6} \right)$$

$$NonFiction = -0 - \frac{1}{6} \log_2 \left(\frac{1}{6} \right) - \frac{1}{6} \log_2 \left(\frac{1}{6} \right)$$

$$Info\ Gain = 1.5575 - \left(\frac{6}{8} \times SciMystery + \frac{2}{8} \times NonFiction \right)$$

$$\{\{SciFiction, NonFiction\}, \{Mystery\}\}$$

$$SciNonFiction = -\frac{2}{4} \log_2 \left(\frac{2}{4} \right) - \frac{1}{4} \log_2 \left(\frac{1}{4} \right) - \frac{1}{4} \log_2 \left(\frac{1}{4} \right)$$

$$NonFiction = -\frac{1}{4} \log_2 \left(\frac{1}{4} \right) - \frac{2}{4} \log_2 \left(\frac{2}{4} \right) - \frac{1}{4} \log_2 \left(\frac{1}{4} \right)$$

$$Info\ Gain = 1.5575 - \left(\frac{4}{8} \times SciNonFiction + \frac{4}{8} \times Mystery \right)$$

$$\{\{Mystery, NonFiction\}, \{NonFiction\}\}$$

$$\text{SciNonFiction} = -1/6 \log_2 (1/6) - 3/6 \log_2 (3/6) - 2/6 \log_2 (2/6)$$

$$\text{NonFiction} = -2/2 \log_2 (2/2) - 0 - 0$$

$$\text{Info Gain} = 1.5575 - (6/8 \times \text{SciNonFiction} + 2/8 \times \text{NonFiction})$$

To determine if any attributes are to be grouped, take the **highest information gain**, and compare it to the Entropy Before Split for music preference. If information gain is higher, then the grouping was better; if not, the grouping was worse.

3 Support Vectors

The equations would reduce down to:

$$10 \alpha_1 + 10 \alpha_2 = -1$$

$$10 \alpha_1 + 11 \alpha_2 = 1$$

For which we get the following solutions:

$$\alpha_1 = -\frac{21}{10} \quad \alpha_2 = 2$$

For the discriminating 2D hyperplane, we get as follows:

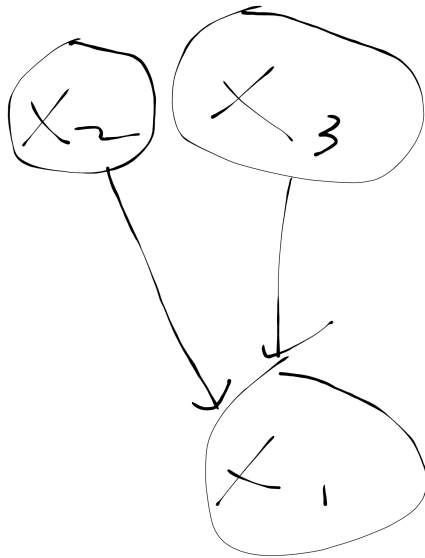
$$\begin{aligned} w &= -\frac{21}{10} \times s_1 + 2 \times s_2 \\ &= \left\langle -\frac{3}{10}, 2, -\frac{41}{10} \right\rangle \end{aligned}$$

Meaning our equation would be as follows:

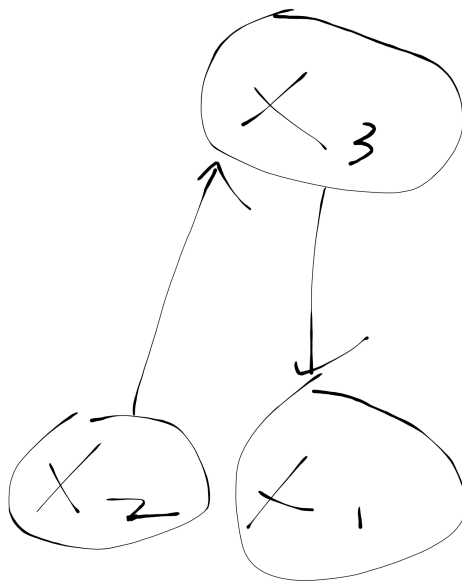
$$-3/10 x - 2 y - 41/10 = 0$$

4 Bayesian Network

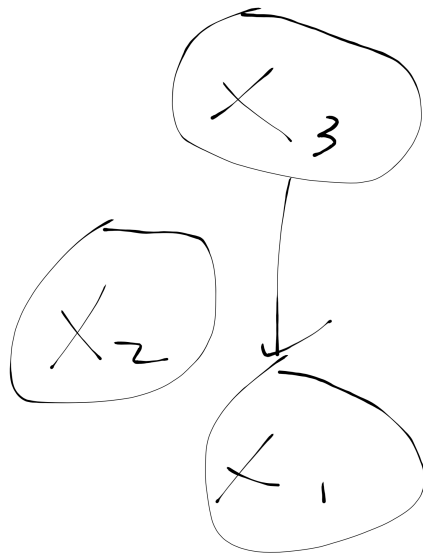
The algorithm will enumerate the possible options (in the original ordering) for a parent of X2. It does so by calculating the probability of the particular attribute being a parent of X2, and the one with the greatest probability is the node that becomes the parents.



1.



2.



3.

5 DBScan

	density	designation
A1	2	Border
A2	2	Border
A3	3	Core
A4	1	Noise
A5	2	Border
A6	2	Border
A7	2	Noise
A8	1	Noise
A9	2	Noise
A10	3	Core

The clusters would be as follows: $\{A1, A2, A3\}$ and $\{A5, A6, A10\}$. Points with have are dense (density > number of points) are designated as core points. Points that are around a core points (within an ϵ of a core point) are designated as border point. Points that fall in neither of these categories are designated as noise.

6 Linkage

The single linkage would be as follows:

$$|(1, 4) - (3, 5)| = 3$$

The complete linkage would be as follows:

$$|(1, 2) - (4, 5)| = 6$$

The centroid linkage would be as follows:

$$|(1, 11/4) - (14/4, 5)| = 19/4$$

The average linkage would be as follows:

$$\text{Average of all distances between points} = 5$$

7 Ensemble Classifier

Sum	-2	4	-6	2
Class	-1	1	-1	1

8 Bayes Network

For **worker** = **T**, the likelihood would be as follows:

$$0.8 \times 0.02 \times 0.1 \times 0.1 \times 0.01$$

For **worker** = **F**, the likelihood would be as follows:

$$0.8 \times 0.02 \times 0.1 \times 0.1 \times 0.99$$

The probability for **worker** = **T** would be:

$$\frac{0.8 \times 0.02 \times 0.1 \times 0.1 \times 0.01}{0.8 \times 0.02 \times 0.1 \times 0.1 \times 0.01 + 0.8 \times 0.02 \times 0.1 \times 0.1 \times 0.99}$$

And the probability for **worker** = **F** would be:

$$\frac{0.8 \times 0.02 \times 0.1 \times 0.1 \times 0.99}{0.8 \times 0.02 \times 0.1 \times 0.1 \times 0.01 + 0.8 \times 0.02 \times 0.1 \times 0.1 \times 0.99}$$

Multiple Choice

9. d. repeatedly sampling from the original dataset according to a uniform probability distribution
10. c. increased, decreased

11. a. bias, variance
12. b. false
13. c. 180 instances reached this point in the decision tree, but 22 of those were not classified as tested_negative in the training dataset
14. d. $\text{age} > 34$
15. b. compute the predicted error rate of the rule with one condition (C1, C2, C3) deleted and no conditions deleted, and, from those, use the version of the rule with the lowest rate
16. c. each instance in the dataset will have a probability of being in a particular cluster
17. b. supervised

Homework #1

CS5402 — Intro To Data Mining

Illya Starikov

Due Date: July 6th, 2018

1 1-Rule Method

Problem Statement. Use the 1-rule (1R) method to find the best single attribute to determine restaurant. In order to demonstrate that you actually know how this method works (and aren't just guessing at which attribute is best), you must fill in ALL of the blank values in the table below; otherwise, you will not receive any credit for this problem.

Attribute	Attribute Value	# Rows	Most Frequent Value	Errors	Total Errors
Meal Preference	Hamburger	3	McDonalds (3)	0	2
	Fish	2	Burger King (2)	0	
	Chicken	4	Wendy's (2)	2	
Gender	M	5	Burger King/McDonald's (2/2)	3	5
	F	4	McDonald's (2)	2	
Drink Preference	Pepsi	3	Burger King (2)	1	3
	Coke	6	McDonald's (4)	2	

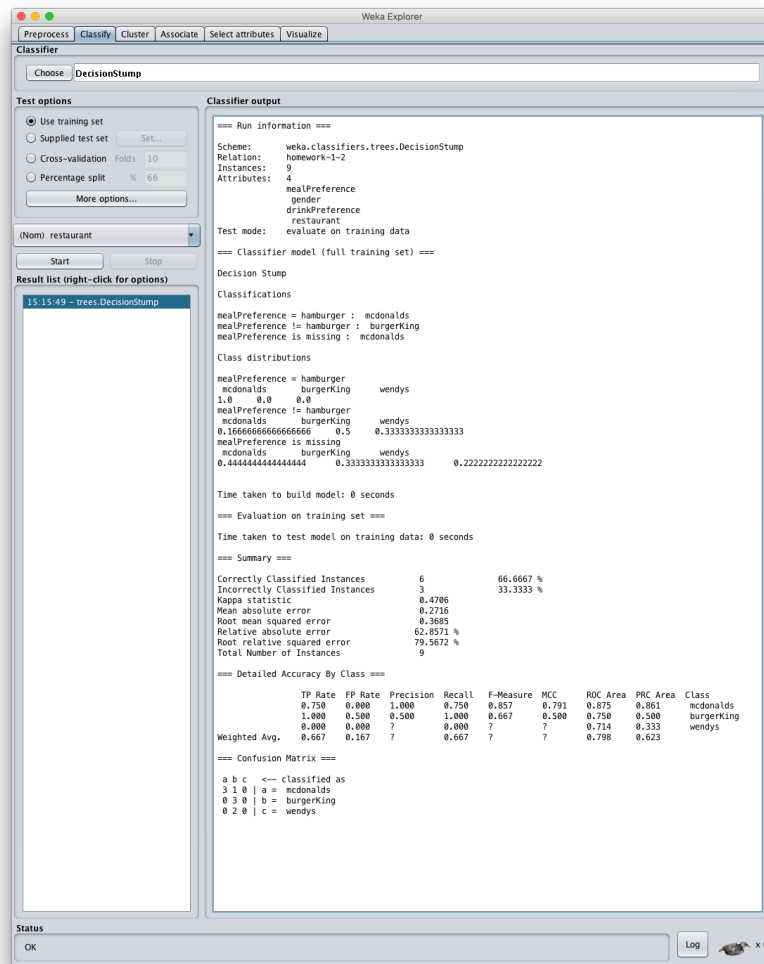
From this, the following rules would be generated:

Meal Preference = *Hamburger* $\implies$ McDonald's
Meal Preference = *Fish* $\implies$ Burger King
Meal Preference = *Chicken* $\implies$ Wendy's

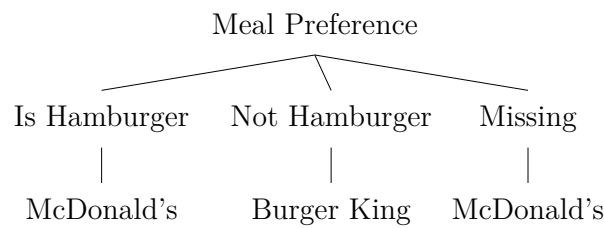
2 Decision Stump

Problem Statement. Create the dataset given in problem 1. as an arff or csv file, and run *DecisionStump* on it in Weka. List the classification rules that are produced by Weka (you can just include a screenshot of your Weka output) AND hand-draw the decision tree that corresponds to those rules.

From Weka's Decision Stump, we get the following output:



For the figure, we get the following decision tree:



3 Statistical Modeling

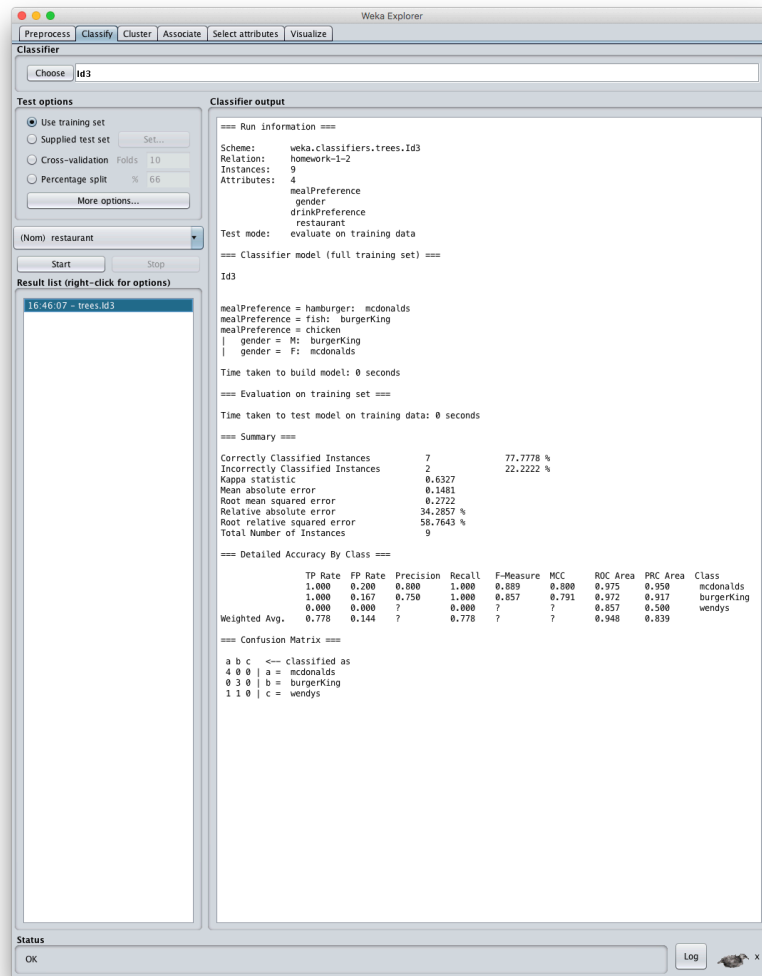
Problem Statement. *Statistical modeling can be used to compute the probability of occurrence of an attribute value. Based on the data given in the table below, if we have a new instance where $ageGroup = youngAdult$, $gender = M$, and $bookPreference = nonFiction$, what is the likelihood that $musicPreference = country$? Just set up the equation to compute this with the appropriate values; you don't have to actually calculate the final answer.*

$$\text{Likelihood of Country} = \frac{2}{3} \times \frac{1}{3} \times \frac{1}{3} \times \frac{3}{8} = \frac{1}{36} \approx 0.02\bar{7}$$

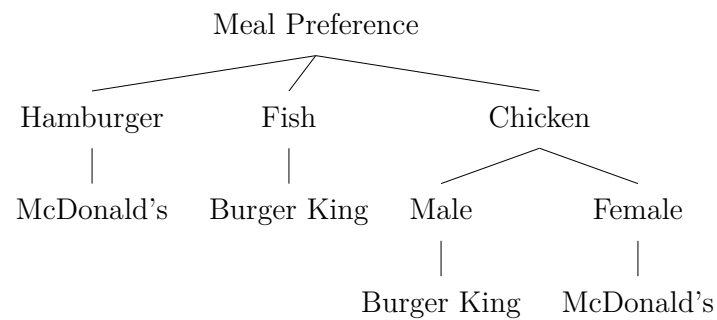
4 Id3

Problem Statement. *Create the dataset given in problem 1. as an arff or csv file, and run Id3 on it in Weka. Include a screenshot of the decision tree output that is produced by Weka AND hand-draw the decision tree. Note: You may have to install the package containing Id3.*

From Weka's Id3, we get the following output:



For the figure, we get the following decision tree:



5 Entropy

Problem Statement. *If we want to make a decision tree for determining restaurant, we must decide which of the three non-decision attributes (mealPreference, gender, or drinkPreference) to use as the root of the tree.*

- *Set up the equation to compute what in lecture we called entropyBeforeSplit for restaurant. You do not have to actually solve (i.e., calculate the terms in) the equation, just set up the equation with the appropriate values.*

$$\text{Entropy Before Split} = -3/10 \log_2 (3/10) - 5/10 \log_2 (5/10) - 2/10 \log_2 (2/10) \approx 1.485$$

- *Set up the equation to compute entropy for mealPreference when its value is chicken. That is, a tree with mealPreference at the root would have three branches (one for hamburger, one for chicken, and one for fish), requiring us to compute entropyHamburger, entropyChicken, and entropyFish; here we only want you to set up the equation to compute entropyChicken. You do not have to actually solve (i.e., calculate the terms in) the equation, just set it up using the appropriate values.*

$$\text{Entropy Chicken} = -1/4 \log_2 (1/4) - 1/4 \log_2 (1/4) - 2/4 \log_2 (2/4) \approx 1.040$$

- *Suppose that instead of considering mealPreference to be the root of this decision tree, we had instead considered drinkPreference. Set up the equation to compute information gain for drinkPreference given the variables specified below.*

$$\text{Information Gain} = X - (7/10 C + 3/10 P)$$

6 isSticky

Problem Statement. *Create the dataset shown below as an arff or csv file and run the Prism algorithm on it in Weka specifying isSticky as the decision attribute. List the classification rules that are produced (you can just include a screenshot of your Weka output). Then work out the Prism algorithm by hand to show what classification rules you would get; who knows, they might be different!*

6.1 Is Sticky = No

consistency	packaging	chocolate	roundShaped	isSticky
soft	individuallyWrapped	yes	no	no
soft	box	yes	no	no
hard	sack	no	yes	no
hard	box	yes	yes	no
hard	individuallyWrapped	no	no	yes
hard	box	no	yes	yes
soft	individuallyWrapped	no	no	yes

Homework #2

CS5402 — Intro To Data Mining

Illya Starikov

Due Date: July 10th, 2018

1 Association Rules

Problem Statement. *Compute the coverage (i.e., support) of each item set listed below.*

drinkPref = pepsi 2

married = no, pet = dog 3

sportPref = football, musicPref = classical, drinkPref = tea 0

Problem Statement. *Write down every association rule that could be generated from the item set listed below, regardless of whether or not there are actually any instances of that rule in our given dataset.*

married = no $\implies$ **pet = dog**

pet = dog $\implies$ **married = no**

— $\implies$ **married = no and pet = dog**

Problem Statement. *Compute the accuracy (i.e., confidence) of each rule listed below. Express accuracy as a fraction (e.g., 2/3, 2/2, etc.), NOT as a decimal number (e.g., 0.67, 1.0, etc.).*

With the case *If pet = dog then income = middle*, we get as follows:

$$\text{Accuracy} = \frac{1}{4}$$

With the case *If married = no and sportPref = football, then pet = dog and musicPref = rock*, we get as follows:

$$\text{Accuracy} = \frac{2}{3}$$

With the case *If* __, *then drinkPref = coke and married = yes*, we get as follows:

$$\text{Accuracy} = 0$$

Problem Statement. *In each rule listed below, specify whether any condition(s) in the antecedent (i.e., the “if” part) can be dropped.*

If married = no and sportPref = football then pet = dog and drinkPref = coke No.

If married = yes and pet = cat then musicPref = country Yes, married = yes can be dropped.

2 RICO

Problem Statement. *Using the Rule Induction from Coverings algorithm discussed in class, find each of the partitions specified below.*

$$\begin{aligned} \{a\}^* &= \{\{x_1, x_5\}, \{x_2, x_3, x_4\}\} \\ \{c\}^* &= \{\{x_1\}, \{x_2, x_3\}, \{x_4\}, \{x_5\}\} \\ \{d\}^* &= \{\{x_1, x_2\}, \{x_3, x_4\}, \{x_5\}\} \\ \{d, e\}^* &= \{\{x_1\}, \{x_2\}, \{x_3, x_4\}, \{x_5\}\} \end{aligned}$$

Problem Statement. *Suppose that we are looking for a covering for decision attribute f which has partition $\{f\}^* = \{\{x_1, x_2\}, \{x_3, x_4, x_5\}\}$.*

- No, the block $\{x_1, x_5\}$ is not a subset of any blocks in $\{f\}^*$.
- No, the block $\{x_2, x_3\}$ is not a subset of any blocks in $\{f\}^*$.
- Yes, all blocks are subsets of all blocks in $\{f\}^*$.
- No, because $\{d, e\}^*$ is not minimal.

Problem Statement. *Find a covering for f . If you found one in part b that worked, just use one of those. Using your covering, give a set of rules for decision attribute f .*

$$\begin{aligned} d = 0 &\implies f = \text{True} \\ d = 1 &\implies f = \text{False} \\ d = 2 &\implies f = \text{False} \end{aligned}$$

3 KD-Tree

Problem Statement. Consider the dataset given below where the decision attribute is the one labeled decision. Build a kd-tree where $k = 3$. No partial credit will be given unless you show your work!

Sorting the tuples by x , we get as follows:

$$[(1, 7, 9), (2, 5, 8), (3, 6, 7), (4, 4, 0), (5, 3, 4), (6, 1, 6), (7, 2, 2)]$$

Seeing a our median = 4, we partition as follows:

$$[(1, 7, 9), (2, 5, 8), (3, 6, 7)] \quad [(4, 4, 0), (5, 3, 4), (6, 1, 6), (7, 2, 2)]$$

Sorting these tuples by y , we get as follows:

$$[(2, 5, 8), (3, 6, 7), (1, 7, 9)] \quad [(6, 1, 6), (7, 2, 2), (5, 3, 4), (4, 4, 0)]$$

With our medians as follows, we get as follows:

$$\begin{array}{ccccccc} \text{median} = 6 & & & & \text{median} = 2.5 & & \\ [(2, 5, 8)] & [(3, 6, 7), (1, 7, 9)] & [(6, 1, 6), (7, 2, 2)] & [(5, 3, 4), (4, 4, 0)] \end{array}$$

Sorting these tuples by z , we get as follows:

$$[(2, 5, 8)] \quad [(3, 6, 7), (1, 7, 9)] \quad [(7, 2, 2), (6, 1, 6)] \quad [(4, 4, 0), (5, 3, 4)]$$

With our medians as follows, we get as follows:

$$\begin{array}{ccccccc} \text{median} = 8 & \text{median} = 8 & & \text{median} = 4 & & \text{median} = 2 & \\ [(2, 5, 8)] & [(3, 6, 7)] & [(1, 7, 9)] & [(7, 2, 2)] & [(6, 1, 6)] & [(4, 4, 0)] & [(5, 3, 4)] \end{array}$$

To produce the following KD-Tree:

Homework #3

CS5402 — Intro To Data Mining

Illya Starikov

Due Date: July 17th, 2018

1 C4.5

Split	Entropy Less Than	Entropy Greater Than	Entropy Before Split	Entropy After Split	Information Gain
8	0	1.55459	1.579	1.435	0.14400
10	1	1.57262	1.579	1.48453	0.09447
15	0.9183	1.52193	1.579	1.38263	0.19637
20	0.81128	1.39215	1.579	1.21342	0.36558
24	0.97095	1.40564	1.579	1.23845	0.34055
28	0.9183	0.98523	1.579	0.95434	0.62466
28	1.37878	1	1.579	1.20396	0.37504
28	1.5	0.97095	1.579	1.29652	0.28248
29	1.53049	1	1.579	1.36726	0.21174
30	1.52193	0.9183	1.579	1.38263	0.19637
30	1.49492	0	1.579	1.26493	0.31407
31	1.55459	0	1.579	1.435	0.14400
33	1.57662	0	1.579	1.57662	0.00238

2 Grouping

Grouping	Entropy	Entropy Before Split	Entropy After Split	Information Gain
Coke-Pepsi	1.5219	1.565	1.2652	0.2998
Mountain Dew	1.5850	1.565		
Coke-Mountain Dew	1.4591	1.565	1.0943	0.4706
Pepsi	0	1.565		
Pepsi-Mountain Dew	1.3710	1.565	0.9623	0.6028
Coke	0.9183	1.565		

No, they should not be grouped; the respective information gains for the groupings are less than the information gain without grouping.

3 J48

For the deepest subtree, `credit_history`, we get the following error for the individual leaves:

$$U_{90\%}(0, 1) + U_{90\%}(0, 1) + 7 \times U_{90\%}(1, 7) + 0 + 2 \times U_{90\%}(0, 2) \approx \\ 0.95 + 0.95 + 7 \times 0.5207 + 0 + 2 \times 0.77639 \approx 3.19709$$

If the subtree would just be replaced by a leaf **good (9/2)**, predicted error would be

$$9 \times U_{90\%}(2, 9) \approx 9 \times 0.47009 \approx 4.23081$$

Therefore, the new tree would be

```
property_magnitude = car
|   personal_status = male div/sep: bad (1.0)
|   personal_status = female div/dep/mar: good (7.0)
|   personal_status = male single: good (9.0/2.0)
|   personal_status = male mar/wid: good (1.0)
|   personal_status = female single: good (0.0)
```

For this subtree, we get the following error for individual leaves:

$$U_{90\%}(0, 1) + 7 \times U_{90\%}(0, 7) + 9 \times U_{90\%}(2, 9) + U_{90\%}(0, 1) + 0 \approx \\ 0.95 + 7 \times 0.348169 \times 0.470090 \approx 7.61793$$

Again, if the subtree would be replaced by a leaf **good (17/2)**, predicted error would be

$$17 \times U_{90\%}(2, 17) \approx 17 \times 0.70420 \approx 5.0286$$

Therefore, the new tree would be

```
property_magnitude = car: good (17/2)
```

4 Missing Value

4.1 Outdoor

4.1.1 Hair = Short

$$\text{entropy} = -\frac{1 + 2/7}{2 + 2/7} \log_2 \frac{1 + 2/7}{2 + 2/7} - \frac{1}{2 + 2/7} \log_2 \frac{1}{2 + 2/7} = 0.9887$$

4.1.2 Hair = Long

$$\text{entropy} = \frac{0}{\vdots} \dots = 0$$

4.2 Indoor

4.2.1 Hair = Short

$$\text{entropy} = -\frac{1}{1+2/7} \log_2 \frac{1}{1+2/7} - \frac{2/7}{1+2/7} \log_2 \frac{2/7}{2+2/7} = 0.7642$$

4.2.2 Hair = Long

$$\text{entropy} = -\frac{1}{1} \log_2 \frac{1}{1} = 0$$

4.3 Both

4.3.1 Hair = Short

$$\text{entropy} = -\frac{1+3/7}{1+3/7} \log_2 \frac{1+3/7}{2+3/7} = 0 = 0$$

4.3.2 Hair = Long

$$\text{entropy} = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1$$

Homework #5

CS5402 — Intro To Data Mining

Illya Starikov

Due Date: July 24th, 2018

1 DBSCAN

a) The table would look as follows:

Point	x	y	Density	Designation
A1	1	4	3	Core Point
A2	5	2	3	Core Point
A3	4	3	3	Core Point
A4	5	6	2	Border Point
A5	2	5	2	Border Point
A6	5	4	4	Core Point
A7	1	2	2	Border Point
A8	3	1	1	Noise

b) The points in the two clusters would be:

- {A1, A5, A7}
- {A2, A3, A4, A6}

The clusters are formed by first identifying the main points; this is done via calculating the ϵ , or the density of the points (i.e., how many points are around those points). After the core points have been identified, the border points are found via observing which points are close (within an ϵ value) to the core points. Finally, the remaining points are simply noise.

2 Bagging Vs Boosting

2.1 Iris

Looking at the confusion matrix of using bagging:


```

a b c <-- classified as
50 0 0 | a = Iris-setosa
0 0 50 | b = Iris-versicolor
0 0 50 | c = Iris-virginica

```

And comparing to the confusion matrix of boosting:

```

a b c <-- classified as
50 0 0 | a = Iris-setosa
0 45 5 | b = Iris-versicolor
0 1 49 | c = Iris-virginica

```

We see that boosting is far more accurate; the kappa statistic agrees, where bagging had a kappa statistic of 0.5 versus the boosting kappa statistic of 0.94.

2.2 Iris

Looking at the confusion matrix of bagging:

```

445 13 | a = benign
7 234 | b = malignant

```

And comparing to the confusion matrix of the boosting:

```

a b <-- classified as
445 13 | a = benign
17 224 | b = malignant

```

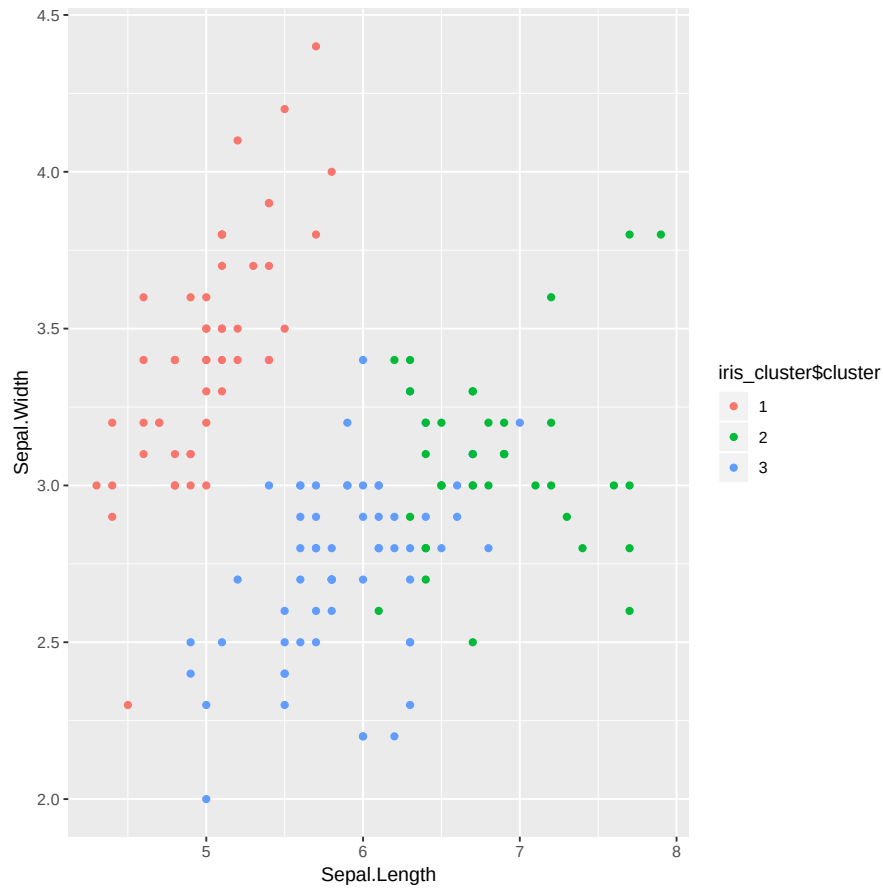
We see there is very little difference in accuracy. The kappa statistic agrees; where bagging had a kappa statistic of 0.937 versus the boosting kappa statistic of 0.9046.

3 KMeans In R

```

1 library(ggplot2)
2
3 iris_copy <- data.frame(iris[1:4])
4 iris_cluster = kmeans(iris_copy, 3)
5
6 table(iris_cluster$cluster, iris$Species)
7
8 # could not get regular plot to work so this is what i got to work
9 # requires packages ggplot2 + labeling (and dependencies)
10 iris_cluster$cluster <- as.factor(iris_cluster$cluster)
11 ggplot(iris, aes(Sepal.Length, Sepal.Width, color = iris_cluster$cluster))
   + geom_point()

```

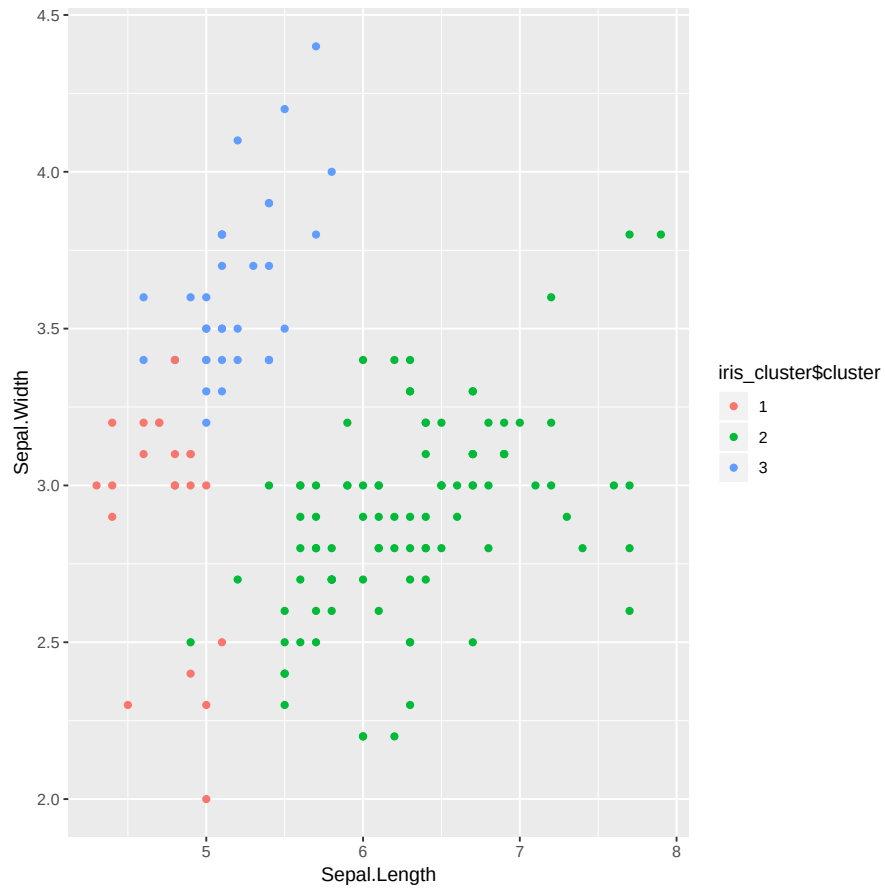


4 DBSCAN In R

```

1 library(fpc)
2 iris_cluster = dbscan(iris_cluster, eps=0.42, MinPts = 5)
3
4 table(iris_cluster$cluster, iris$Species)
5
6 # could not get regular plot to work so this is what i got to work
7 # requires packages ggplot2 + labeling (and dependencies)
8 iris_cluster$cluster <- as.factor(iris_cluster$cluster)
9 ggplot(iris, aes(Sepal.Length, Sepal.Width, color = iris_cluster$cluster))
  + geom_point()

```



5 Frames For Days

```

1 refund = c(
2   "y",
3   "n",
4   "n",
5   "y",
6   "n",
7   "n",
8   "y",
9   "n",
10  "n",
11  "n"
12 )
13 status = c(
14   "single",
15   "married",
16   "single",
17   "married",
18   "divorced",

```

```

19     "married",
20     "divorced",
21     "single",
22     "married",
23     "single"
24 )
25
26 income_k = c(
27     125,
28     100,
29     70,
30     120,
31     95,
32     60,
33     220,
34     85,
35     75,
36     90
37 )
38 class = c(
39     FALSE,
40     FALSE,
41     FALSE,
42     FALSE,
43     TRUE,
44     FALSE,
45     FALSE,
46     TRUE,
47     FALSE,
48     TRUE
49 )
50
51 tax_info = data.frame(refund, status, income_k, class)

```

6 She's So Mean

```

1 tax_info <- data.frame(refund, status, income_k, class)
2 total <- 0
3
4 for (i in 1:nrow(tax_info)) {
5     total <- total + tax_info$income_k[i]
6 }
7
8 average <- total / nrow(tax_info)
9 print(average)

```

7 Animals

```
1 library(party)
2
3 data <- read.csv("C:\\temp\\animal data.csv")
4 data_frame = data.frame(data)
5
6 tree = ctree(Name ~ BloodType + GivesBirth + CanFly + LivesInWater, data=
  data_frame)
7 plot(tree, type="simple")
```

Homework #6

CS5402 — Intro To Data Mining

Illya Starikov

Due Date: July 27th, 2018

1 Support Vectors

```
1 library(kernlab)
2
3 xy = matrix(c(1, 2, 4, 1, 1, 5, 2, 1, 2, 0, 1, 2), nrow=6, ncol=2)
4 z = matrix(c(1, 1, -1, 1, 1, -1), nrow=6, ncol=1)
5
6 svp = ksvm(xy, z, type="C-svc", kernel='vanilladot', C=100, scaled=c())
7
8 print(xmatrix(svp))
9 print(predict(svp, matrix(c(0, 1), nrow=1, ncol=2)))
10 print(predict(svp, matrix(c(4, 1), nrow=1, ncol=2)))
```

The output is as follows:

```
Setting default kernel parameters
[[1]]
  X1 X2
2  2  1
3  4  2

[1] 1
[1] -1
```

2 Bayes Network

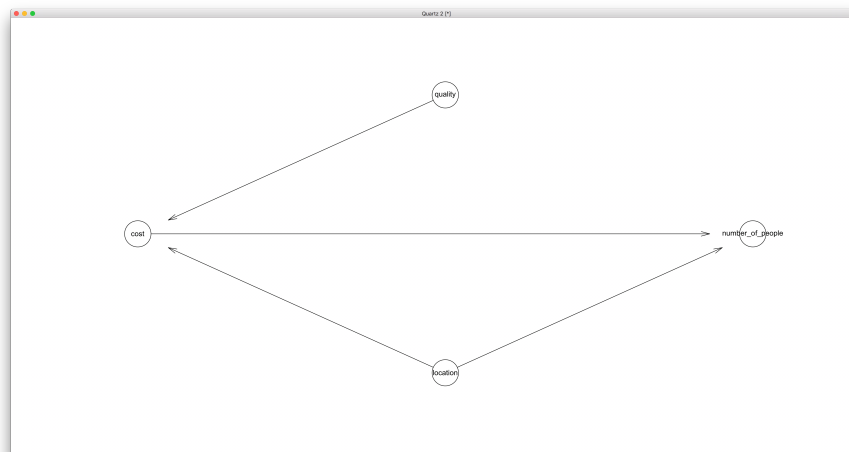
```
1 library(bnlearn)
2 dag = model2network("[location][cost|location:quality][quality][number_of_
   people|location:cost]")
3
4 location.values      = factor(c("Good", "Bad"))
5 quality.values       = factor(c("Good", "Normal", "Bad"))
```

```

6 cost.values = factor(c("High", "Low"))
7 number_of_people.values = factor(c("High", "Low"))
8
9 plot(dag)
10
11 location.prob = array(c(0.6, 0.4), dim=2, dimnames=list(location=
    location.values))
12 quality.prob = array(c(0.3, 0.5, 0.2), dim=3, dimnames=list(
    quality=quality.values))
13 cost.prob = array(c(0.8, 0.2, 0.6, 0.4, 0.1, 0.9, 0.6, 0.4,
    0.6, 0.4, 0.05, 0.95), dim=c(2,3,2), dimnames=list(cost=cost.values,
    quality=quality.values, location=location.values))
14 number_of_people.prob = array(c(0.6, 0.4, 0.8, 0.2, 0.1, 0.9, 0.6, 0.4),
    dim=c(2, 2, 2), dimnames=list(number_of_people=number_of_people.values,
    cost=cost.values, location=location.values))
15
16 conditional_probability_table = list(location=location.prob, quality=
    quality.prob, cost=cost.prob, number_of_people=number_of_people.prob)
17 print(conditional_probability_table)
18
19 bayes_network = custom.fit(dag, conditional_probability_table)
20
21 location = factor(c("Good", "Bad"))
22 quality = factor(c("Normal", "Good"))
23 ccost = factor(c("High", "Low"))
24 number_of_people = factor(c("Low", "High"))
25
26 d_test = data.frame(location, quality, ccost, number_of_people)
27 names(d_test) = c("location", "quality", "cost", "number_of_people")
28 prediction = predict(bayes_network, "quality", d_test, debug=FALSE)
29
30 table(prediction, d_test[, "quality"])
31 print(prediction)

```

The network looks as follows:



The output looks as follows:

```
$location
location
Good  Bad
0.6  0.4
```

```
$quality
quality
  Good Normal  Bad
    0.3    0.5   0.2
```

```
$cost
, , location = Good
```

```
      quality
cost   Good Normal Bad
  High 0.8    0.6 0.1
  Low  0.2    0.4 0.9
```

```
, , location = Bad
```

```
      quality
cost   Good Normal Bad
  High 0.6    0.6 0.05
  Low  0.4    0.4 0.95
```

```
$number_of_people
, , location = Good
```

```
      cost
number_of_people High Low
      High  0.6 0.8
      Low   0.4 0.2
```

```
, , location = Bad
```

```
      cost
number_of_people High Low
      High  0.1 0.6
      Low   0.9 0.4
```



```

prediction Good Normal
  Good      0      0
  Normal    1      1
  Bad       0      0
[1] Normal Normal
Levels: Good Normal Bad

```

3 DBScan

```

1 using RDatasets, Clustering, Distances, Gadfly
2
3 cars = dataset("datasets", "mtcars")
4 x = convert(Array, cars[:,3])
5 y = convert(Array, cars[:,11])
6 distances = pairwise(Euclidean(), x, y)
7
8 cluster = dbscan(distances, 2, 5)
9
10 assignments = assignments(cluster)
11 cluster_plot = plot(x=x, y=y, color=assignments, Geom.point);
12
13 print(cluster)

```

```

Clustering.DbscanResult(
  [1, 5, 7, 12, 13, 14, 15, 16, 17, 22, 23, 24, 25, 29, 31],
  [1, 1, 1, 1, 2, 1, 3, 1, 1, 1, 1, 4, 5, 6, 7, 8, 9, 1, 1, 1,
    1, 10, 11, 12, 13, 1, 1, 1, 14, 1, 15, 1],
  [18, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])

```

Gadfly could not produce a graph.

4 Decision Tree

```

1 using RDatasets, DecisionTree, DataFrames
2
3 cars = dataset("datasets", "mtcars")
4
5 features = convert(Array, cars[:, [12, 3, 5, 11]])
6 classification = convert(Array, cars[:, 2])
7 model = build_tree(classification, features)
8
9 print_tree(model)

```

```

Feature 3, Threshold 118.0
L-> Feature 3, Threshold 96.0
  L-> Feature 3, Threshold 65.5
    L-> Feature 1, Threshold 1.5
      L-> 33.9 : 1/1
      R-> Feature 3, Threshold 57.0
        L-> 30.4 : 1/1
        R-> 24.4 : 1/1
    R-> Feature 3, Threshold 92.0
      L-> Feature 1, Threshold 1.5
        L-> 27.3 : 1/2
        R-> 26.0 : 1/1
      R-> 22.8 : 2/2
  R-> Feature 1, Threshold 3.0
    L-> Feature 3, Threshold 107.0
      L-> Feature 2, Threshold 5.0
        L-> 21.5 : 1/1
        R-> 18.1 : 1/1
      R-> Feature 4, Threshold 4.5
        L-> 21.4 : 2/2
        R-> 30.4 : 1/1
    R-> 21.0 : 2/2
R-> Feature 3, Threshold 192.5
  L-> Feature 2, Threshold 7.0
    L-> Feature 4, Threshold 4.5
      L-> 17.8 : 1/2
      R-> 19.7 : 1/1
    R-> Feature 1, Threshold 2.5
      L-> Feature 3, Threshold 162.5
        L-> 15.2 : 1/2
        R-> 19.2 : 1/2
      R-> 16.4 : 1/3
  R-> Feature 3, Threshold 237.5
    L-> Feature 3, Threshold 222.5
      L-> 10.4 : 2/2
      R-> 14.7 : 1/1
    R-> Feature 3, Threshold 254.5
      L-> 14.3 : 1/2
      R-> Feature 3, Threshold 299.5
        L-> 15.8 : 1/1
        R-> 15.0 : 1/1

```

5 KMeans

```
1 using CSV, DataFrames
2 using Clustering
3 using Gadfly
4
5 data_frame = CSV.read("./problem-5.csv"; types=[Float64, Float64, Float64
6   ])
7 data = convert(Array, data_frame[:, [1, 2]])'
8
9 k_means = kmeans(data, 3; maxiter=200, display=:iter)
10
11 a = assignments(k_means)
12 c = counts(k_means)
13
14 print("Assignments: ")
15 println(a')
16
17 print("Counts: ")
18 println(c')
19
20 plot(x=data[1, :], y = data[2, :], color=a, Geom.point)
```

Iters	objv	objv-change	affected
0	1.000000e+02		
1	4.880000e+01	-5.120000e+01	0
2	4.880000e+01	0.000000e+00	0

K-means converged with 2 iterations (objv = 48.800000000000001)

Assignments: [1 1 1 1 2 1 3]

Counts: [5 1 1]

MATH1214

Calculus I

S&TTM

Calculus I: Single-Variable Calculus

Illya Starikov

August 6, 2025

Contents

0	Functions	2
0.1	Review of Functions	2
0.2	Review of Functions	2
0.3	Representing Functions	3
0.4	Inverse, Exponential, and Logarithmic Functions	5
0.5	Trigonometric Functions and Their Inverses	7
1	Limits	8
1.2	Definitions of Limits	8
1.3	Techniques For Computing Limits	9
1.4	Infinite Limits	10
1.5	Limits at Infinity	11

num₁!

0 Functions

0.1 Review of Functions

A **function** is a rule that assigns to each value x in a set D a unique value denoted $f(x)$. The set D is the **domain** of the function. The **range** is the set of all values of $f(x)$ produced as x varies over the domain.

Vertical Line Test

A graph represents a function if and only if it passes the **vertical line test**. Every vertical line intersects the graph at most once. A graph that fails this test does not represent a function.

Composite Functions

Given two functions f and g , the composite functions $f \circ g$ is defined by $(f \circ g)(x) = f(g(x))$. It is evaluated in two steps: $y = f(u)$, where $u = g(x)$. The domain of $f \circ g$ consists of all x in the domain of g such that $u = g(x)$ is in the domain of f .

Symmetry in Functions

An **even function** has the property that $f(-x) = f(x)$, for all x in the domain. The graph of an even function is symmetric about the y-axis. Polynomials consisting of only even powers of the variable (of the form x^{2n} , where n is a nonnegative integer) are even functions.

An **odd function** f has the property that $f(-x) = -f(x)$, for all x in the domain. The graph of an odd function is symmetric about the origin. Polynomials consisting of only odd powers of the variable (of the form x^{2n+1} , where n is a nonnegative integer) are odd functions.

0.2 Review of Functions

A **function** is a rule that assigns to each value x in a set D a unique value denoted $f(x)$. The set D is the **domain** of the function. The **range** is the

set of all values of $f(x)$ produced as x varies over the domain.

Vertical Line Test

A graph represents a function if and only if it passes the **vertical line test**. Every vertical line intersects the graph at most once. A graph that fails this test does not represent a function.

Composite Functions

Given two functions f and g , the composite functions $f \circ g$ is defined by $(f \circ g)(x) = f(g(x))$. It is evaluated in two steps: $y = f(u)$, where $u = g(x)$. The domain of $f \circ g$ consists of all x in the domain of g such that $u = g(x)$ is in the domain of f .

Symmetry in Functions

An **even function** has the property that $f(-x) = f(x)$, for all x in the domain. The graph of an even function symmetric about the y-axis. Polynomials consisting of only even powers of the variable (of the form x^{2n} , where n is a nonnegative integer) are even functions.

An **odd function** f has the property that $f(-x) = -f(x)$, for all x in the domain. The graph of an odd function is symmetric about the origin. Polynomials consisting of only odd powers of the variable (of the form x^{2n+1} , where n is a nonnegative integer) are odd functions.

0.3 Representing Functions

Some brief families of functions can include

Polynomials are functions of the form

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0,$$

where the **coefficients** $a_0, a_1, \dots, a_n$ are real numbers with $a_n \neq 0$ and the nonnegative integer n is the **degree** of the polynomial. The domain of any polynomial is the set of all real numbers. An n th-degree polynomial can have as many as n real **zeros** or **roots** — values of x .

Rational Functions are ratios of the form $f(x) = p(x)/q(x)$, where p and q are polynomials. Because division by zero is prohibited, the domain of a rational function is the set of all real numbers except those for which the denominator is zero.

Algebraic Functions are constructed using the operations of algebra: addition, subtraction, multiplication, division, and roots. Examples of algebraic functions are $f(x) = \sqrt{2x^3 + 4}$ and $f(x) = x^{1/4}(x^3 + 2)$. In general, if an even root (square root, fourth root, and so forth) appears, then the domain does not contain points at which the quantity under the root is negative (and perhaps other points).

Exponential Functions have the form $f(x) = b^x$, where the base $b \neq 1$ is a positive real number. Closely associated with exponential functions are logarithmic functions of the form $f(x) = \log_b x$, where $b > 0$ and $b \neq 1$. An exponential function has a domain consisting of all real numbers. Logarithmic functions are defined for positive, real numbers. The most important function is the **natural exponential function** $f(x) = e^x$, with base $b = e$, where $e \approx 2.71828\dots$ is one of the fundamental constants of mathematics. Associated with the natural exponential function is the **natural logarithmic function** $f(x) = \ln x$, which also has the base $b = e$.

Trigonometric Functions are $\sin x$, $\cos x$, $\tan x$, $\sec x$, and $\csc x$; they are fundamental to mathematics and many areas of application. Also important are their relatives, the **inverse trigonometric functions**.

Transcendental Functions Trigonometric, exponential, and logarithmic functions are few examples of a large family called transcendental functions.

Transformations

Given the real numbers a , b , c , and d and the function f , the graph of $y = cf(a(x - b)) + d$ is obtained from the graph of $y = f(x)$ in the following steps.

$$\begin{aligned}
y = f(x) & \xrightarrow{\text{horizontal scaling by a factor of } |a|} y = f(ax) \\
& \xrightarrow{\text{horizontal shift by } b \text{ units}} y = f(a(x - b)) \\
& \xrightarrow{\text{vertical scaling by a factor of } |c|} y = cf(a(x - b)) \\
& \xrightarrow{\text{horizontal scaling by a factor of } |a|} y = cf(a(x - b)) + d
\end{aligned}$$

0.4 Inverse, Exponential, and Logarithmic Functions

The Natural Exponential Function

The **natural exponential function** is $f(x) = e^x$, which as the base $e = 2.718281828459\dots$

Inverse Function

Given a function f , its inverse (if it exists) is a function f^{-1} such that whenever $y = f(x)$, then $f^{-1}(y) = x$.

One-to-One Functions and the Horizontal Line Test

A function f is **one-to-one** on a domain D if each value of $f(x)$ corresponds to exactly one value of x in D . More precisely, f is one-to-one on D if $y(x_1) \neq f(x_2)$ whenever $x_1 \neq x_2$ for x_1 and x_2 in D . The **horizontal line test** says that every horizontal line intersects the graph of a one-to-one function at most once.

Existence of Inverse Functions

Let f be one-to-one function on a domain D with a range R . Then f has a unique inverse f^{-1} with domain R and range D such that

$$f^{-1}(f(x)) = x \quad \text{and} \quad f(f^{-1}(y)) = y,$$

where x is in D and y is in R .

Finding an Inverse Function

Suppose f is one-to-one on an interval I . To find f^{-1} :

- Solve $y = f(x)$ for x . If necessary, choose the function that corresponds to I .
- Interchange x and y and write $y = f^{-1}(x)$.

Logarithmic Function Base b

For any base $b > 0$, with $b \neq 1$, the **logarithmic function base b** , denoted $y = \log_b x$, is the inverse of the exponential function $y = b^x$. The inverse of the natural exponential function with base $b = e$ is the **natural logarithm function**, denoted $y = \ln x$.

Inverse Relations For Exponential and Logarithmic Functions

For any base $b > 0$, with $b \neq 1$, the following inverse relations hold:

- $b^{\log_b x} = x$, for $x > 0$
- $\log_b b^x = x$, for any real values of x

Change-of-Base Rules

Let b be a positive real number with $b \neq 1$. Then

$$b^x = e^{x \ln b}, \text{ for all } x \quad \text{and} \quad \log_b x = \frac{\ln x}{\ln b}, \text{ for } x > 0$$

More generally, if c is a positive real number with $c \neq 1$, then

$$b^x = c^{x \log_c b}, \text{ for all } x \quad \text{and} \quad \log_b x = \frac{\log_c x}{\log_c b}, \text{ for } x > 0$$

0.5 Trigonometric Functions and Their Inverses

Let $P(x, y)$ be a point on a circle of radius r associated with the angle θ . Then

$$\sin \theta = \frac{y}{r} \quad \cos \theta = \frac{x}{r} \quad \tan \theta = \frac{y}{x} \quad (1)$$

$$\cot \theta = \frac{x}{y} \quad \sec \theta = \frac{r}{x} \quad \csc \theta = \frac{r}{y} \quad (2)$$

$$(3)$$

Trigonometric Identities

Reciprocal Identities

$$\tan \theta = \frac{\sin \theta}{\cos \theta} \quad \cot \theta = \frac{1}{\tan \theta} = \frac{\cos \theta}{\sin \theta} \quad (4)$$

$$\csc \theta = \frac{1}{\sin \theta} \quad \sec \theta = \frac{1}{\cos \theta} \quad (5)$$

Pythagorean

$$\sin^2 \theta + \cos^2 \theta = 1 \quad 1 + \cot^2 \theta = \csc^2 \theta \quad \tan^2 \theta + 1 = \sec^2 \theta \quad (6)$$

Double- and Half-Angle Formulas

$$\sin 2\theta = 2 \sin \theta \cos \theta \quad \cos 2\theta = \sin^2 \theta - \cos^2 \theta \quad (7)$$

$$\cos^2 \theta = \frac{1 + \cos 2\theta}{2} \quad \sin^2 \theta = \frac{1 - \cos 2\theta}{2} \quad (8)$$

Period of Trigonometric Function

The function $\sin \theta$, $\cos \theta$, $\sec \theta$, and $\csc \theta$ have a period of 2π

$$\sin(\theta + 2\pi) = \sin \theta \quad \cos(\theta + 2\pi) = \cos \theta \quad (9)$$

$$\sec(\theta + 2\pi) = \sec \theta \quad \csc(\theta + 2\pi) = \csc \theta \quad (10)$$

for all θ in the domain.

The functions $\tan \theta$ and $\cot \theta$ have a period of π :

$$\tan(\theta + \pi) = \tan \theta \quad \cot(\theta + \pi) = \cot \theta \quad (11)$$

for all θ in the domain.

Inverse Sine and Cosine

$y = \sin^{-1} x$ is the value of y such that $x = \sin y$, where $-\frac{\pi}{2} \leq y \leq \frac{\pi}{2}$.
 $y = \cos^{-1} x$ is the value of y such that $x = \cos y$, where $0 \leq y \leq \pi$. The domain of both $\sin^{-1} x$ and $\cos^{-1} x$ is $\{x : -1 \leq x \leq 1\}$.

Other Inverse Trigonometric Functions

- $\tan^{-1} x$ is the value of y such that $x = \tan y$, where $-\frac{\pi}{2} < y < \frac{\pi}{2}$.
- $\cot^{-1} x$ is the value of y such that $x = \tan y$, where $0 < y < \pi$.

The domain of both $\tan^{-1} x$ and $\cot^{-1} x$ is $\{x : -\infty < x < \infty\}$

- $\sec^{-1} x$ is the value of y such that $x = \sec y$, where $0 < y < \pi$ with $y \neq \frac{\pi}{2}$
- $\tan^{-1} x$ is the value of y such that $x = \tan y$, where $-\frac{\pi}{2} < y < \frac{\pi}{2}$.

1 Limits

1.2 Definitions of Limits

Limits of a Function (Preliminary)

Suppose the function f is defined for all x near a except possibly at a . If $f(x)$ is arbitrarily close to L (as close to L as we like) for all x sufficiently close (but not equal) to a , we write

$$\lim_{x \rightarrow a} f(x) = L$$

and say the limit of $f(x)$ as x approaches a equals L .

One-Sided Limits

1 Right-sided limits Suppose f is defined for all x near a with $x > a$. If $f(x)$ is arbitrarily close to L for all x sufficiently close to a with $x > a$, we write

$$\lim_{x \rightarrow a^+} f(x) = L \tag{12}$$

and say the limit of $f(x)$ as x approaches a from the right equals L .

2 Left-sided limits Suppose f is defined for all x near a with $x < a$. If $f(x)$ is arbitrarily close to L for all x sufficiently close to a with $x < a$, we write

$$\lim_{x \rightarrow a^-} f(x) = L \quad (13)$$

and say the limit of $f(x)$ as x approaches a from the left equals L .

Relationship Between One-Sided and Two-Sided Limits

Assume f is defined for all x near a except possibly at a . Then $\lim_{x \rightarrow a} f(x) = L$ if and only if $\lim_{x \rightarrow a^-} f(x) = \lim_{x \rightarrow a^+} f(x) = L$.

1.3 Techniques For Computing Limits

Limits of Linear Functions

Let a , b , and m be real numbers. For Linear functions $f(x) = mx + b$,

$$\lim_{x \rightarrow a} f(x) = f(a) = ma + b \quad (14)$$

Limit Laws

Assume $\lim_{x \rightarrow a} f(x)$ and $\lim_{x \rightarrow a} g(x)$ exist. The following properties hold, where c is a real number, and $m > 0$ and $n > 0$ are integers.

1 Sum $\lim_{x \rightarrow a} [f(x) + g(x)] = \lim_{x \rightarrow a} f(x) + \lim_{x \rightarrow a} g(x)$

2 Difference $\lim_{x \rightarrow a} [f(x) - g(x)] = \lim_{x \rightarrow a} f(x) - \lim_{x \rightarrow a} g(x)$

3 Constant Multiple $\lim_{x \rightarrow a} [c f(x)] = c \lim_{x \rightarrow a} f(x)$

4 Product $\lim_{x \rightarrow a} [f(x)g(x)] = \left[\lim_{x \rightarrow a} f(x) \right] \left[\lim_{x \rightarrow a} g(x) \right]$

5 Quotient $\lim_{x \rightarrow a} \left[\frac{f(x)}{g(x)} \right] = \frac{\lim_{x \rightarrow a} f(x)}{\lim_{x \rightarrow a} g(x)}$, provided $\lim_{x \rightarrow a} g(x) \neq 0$

6 Power $\lim_{x \rightarrow a} [f(x)]^n = \left[\lim_{x \rightarrow a} f(x) \right]^n$

7 Fractional Power $\lim_{n \rightarrow \infty} [f(x)]^{n/m} = \left[\lim_{n \rightarrow \infty} f(x) \right]^{n/m}$, provided $f(x) \geq 0$, for x near a , if m is even and n/m is reduced to lowest terms.

Limits of Polynomial and Rational Functions

Assume p and q are polynomials and a is a constant

- Polynomial functions: $\lim_{n \rightarrow \infty} p(x) = p(a)$
- Rational functions: $\lim_{n \rightarrow \infty} \frac{p(x)}{q(x)} = \frac{p(a)}{q(a)}$, provided $q(a) \neq 0$

Limit Laws For One-Sided Limits

Laws 1–6 hold with $\lim_{n \rightarrow \infty}$ replaced by $\lim_{n \rightarrow \infty}$ or $\lim_{n \rightarrow \infty}$. Law 7 is modified as follows, assume $m > 0$ and $n > 0$ are integers.

7 Fractional Power

- $\lim_{n \rightarrow \infty} [f(x)]^{n/m}$, provided $f(x) \geq 0$, for x near a with $x > a$, if m is even and n/m is reduced to lowest terms
- $\lim_{n \rightarrow \infty} [f(x)]^{n/m}$, provided $f(x) \geq 0$, for x near a with $x < a$, if m is even and n/m is reduced to lowest terms

The Squeeze Theorem

Assume the function f , g , and h satisfy $f(x) \leq g(x) \leq h(x)$, for all values of x near a , except possibly at a . If $\lim_{n \rightarrow \infty} f(x) = \lim_{n \rightarrow \infty} h(x) = L$, then $\lim_{n \rightarrow \infty} g(x) = L$.

1.4 Infinite Limits

Suppose f is defined for all x near a . If $f(x)$ grows arbitrarily large for all x sufficiently close (but not equal) to a , we write

$$\lim_{x \rightarrow a} f(x) = \infty \quad (15)$$

We say the limit of $f(x)$ as x approaches a is infinity.

If $f(x)$ is negative and grows arbitrarily large in magnitude for all x sufficiently close (but not equal) to a , we write

$$\lim_{x \rightarrow a} f(x) = -\infty \quad (16)$$

In this case, we say the limit of $f(x)$ as x approaches a is negative infinity. In both cases, *the limit does not exist*.

One-Sided Infinite Limits

Suppose f is defined for all x near a with $x > a$. If $f(x)$ becomes arbitrarily large for all x sufficiently close to a with $x > a$, we write $\lim_{x \rightarrow a^+} f(x) = \infty$. The one-sided infinite limit $\lim_{x \rightarrow a^-} f(x) = -\infty$, $\lim_{x \rightarrow a^+} f(x) = \infty$, and $\lim_{x \rightarrow a^-} f(x) = -\infty$ are defined analogously.

Vertical Asymptote

If $\lim_{x \rightarrow a^-} f(x) = \pm\infty$, $\lim_{x \rightarrow a^+} f(x) = \pm\infty$ or $\lim_{x \rightarrow a} f(x) = \pm\infty$, the line $x = a$ is called a **vertical asymptote** of f .

1.5 Limits at Infinity

If $f(x)$ becomes arbitrarily close to a finite number L for all sufficiently large and positive x , then we write

$$\lim_{x \rightarrow \infty} f(x) = L \quad (17)$$

We say the limit of $f(x)$ as x approaches infinity is L . In this case the line $y = L$ is a **horizontal asymptote** of f . The limit at negative infinity, $\lim_{x \rightarrow -\infty} f(x) = M$, is defined analogously. When the limit exists, the horizontal asymptote is $y = M$.

Infinite Limits at Infinity

If $f(x)$ becomes arbitrarily large as x becomes arbitrary large, then we write

$$\lim_{x \rightarrow \infty} f(x) = \infty \quad (18)$$

The limits $\lim_{x \rightarrow -\infty} f(x) = -\infty$, $\lim_{x \rightarrow -\infty} f(x) = \infty$, and $\lim_{x \rightarrow \infty} f(x) = -\infty$ are defined similarly.

Limit of Infinity at Powers and Polynomials

Let n be a positive integer and let p be the polynomial $p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$, where $a_n \neq 0$.

1. $\lim_{n \rightarrow \infty} x^n = \infty$ when n is even.
2. $\lim_{n \rightarrow \infty} x^n = \infty$ and $\lim_{n \rightarrow \infty} x^n = -\infty$ when n is odd.
3. $\lim_{n \rightarrow \infty} \frac{1}{x^n} = \lim_{n \rightarrow \infty} x^{-n} = 0$
4. $\lim_{n \rightarrow \infty} p(x) = \infty$ or $-\infty$, depending on the degree of the polynomial or the leading coefficient of a_n .

End Behavior and Asymptotes of Rational Functions

Suppose $f(x) = \frac{p(x)}{q(x)}$ is a rational function, where

$$p(x) = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_2 x^2 + a_1 x + a_0 \quad (19)$$

and

$$q(x) = b_n x^n + b_{n-1} x^{n-1} + \cdots + b_2 x^2 + b_1 x + b_0 \quad (20)$$

with $a_m \neq 0$ and $b_n \neq 0$.

1. If $m < n$ then $\lim_{n \rightarrow \infty} f(x) = 0$, and $y = 0$ is a horizontal asymptote of f .
2. If $m = n$, then $\lim_{n \rightarrow \infty} f(x) = a_m/b_n$, and $y = a_m/b_n$ is a horizontal asymptote.
3. If $m > n$, then $\lim_{n \rightarrow \infty} f(x) = \infty$ or $-\infty$, and f has no horizontal asymptote.
4. If $m = n + 1$, then $\lim_{n \rightarrow \infty} f(x) = \infty$ or $-\infty$, f has no horizontal asymptote, but f has a slant asymptote.
5. Assuming that $f(x)$ is in reduced form (p and q share no common factors), vertical asymptotes occur at the zeros of q .

End Behavior of e^x , e^{-x} , and $\ln x$

The end behavior for e^x and e^{-x} on $(-\infty, \infty)$ and $\ln x$ on $(0, \infty)$ is given by the following limits:

$$\lim_{x \rightarrow \infty} e^x = \infty \quad \text{and} \quad \lim_{x \rightarrow -\infty} e^x = 0 \quad (21)$$

$$\lim_{x \rightarrow \infty} e^{-x} = 0 \quad \text{and} \quad \lim_{x \rightarrow -\infty} e^{-x} = \infty \quad (22)$$

$$\lim_{x \rightarrow 0^+} \ln x = -\infty \quad \text{and} \quad \lim_{x \rightarrow \infty} \ln x = \infty \quad (23)$$

MATH1215

Calculus II



Chapter 6: Applications of Integration

Illya Starikov

August 6, 2025

Contents

6.7 Physical Applications	2

6.7 Physical Applications

Mass of a One-Dimensional Object

Suppose a thin bar or wire is represented by a line segment on the interval $a \leq x \leq b$ with a density function ρ (with units of mass per length). The mass of the object is

$$m = \int_a^b \rho(x) dx \quad (1)$$

Work

The work done by a variable force F in moving an object along a line from $x = a$ to $x = b$ in the direction of the force is

$$W = \int_a^b F(x) dx \quad (2)$$

Solving Lifting Problems

1. Draw a y -axis in the vertical direction (parallel to gravity) and choose a convenient origin. Assume the interval $[a, b]$ corresponds to the vertical extent of the fluid.
2. For $a \leq y \leq b$, find the cross-sectional area $A(y)$ of the horizontal slices and the distance $D(y)$ the slices must be lifted.
3. The work required to lift the water is

$$W = \int_a^b \rho g A(y) D(y) dy \quad (3)$$

Solving Force/Pressure Problems

1. Draw a y -axis on the face of the dam in the vertical direction and choose a convenient origin (often taken to be the base of the dam).
2. Find the width function $w(y)$ for each value of y on the face of the dam.

3. If the base of the dam is at $y = 0$ and the top of the dam is at $y = a$, then the total force on the dam is

$$F = \int \rho g(a - y)w(y) dy \quad (4)$$

Chapter 7: Logarithmic and Exponential Functions

Illya Starikov

August 6, 2025

Contents

7.1	Inverse Function	2
7.2	The Natural Logarithmic and Exponential Functions	3
7.3	Logarithmic and Exponential Functions with Other Bases	5
7.5	Inverse Trigonometric Functions	7
7.6	L'Hôpital's Rule and Growth Rates of Functions	8
7.7	Hyperbolic Functions	9

7.1 Inverse Function

Derivative of the Inverse Function

Let f be differentiable and have an inverse on an interval I . If x_0 is a point of I at which $f'(x_0) \neq 0$, then f^{-1} is differentiable at $y_0 = f(x_0)$ and

$$(f^{-1})'(y_0) = \frac{1}{f'(x_0)}, \text{ where } y_0 = f(x_0) \quad (1)$$

7.2 The Natural Logarithmic and Exponential Functions

The Natural Logarithm

The **natural logarithm** of a number $x > 0$, denoted $\ln x$, is defined

$$\ln x = \int_1^x \frac{dt}{t} \quad (2)$$

Properties of the Natural Logarithm

1. The domain and range of $\ln x$ are $(0, \infty)$ and $(-\infty, \infty)$, respectively.
2. $\ln(xy) = \ln x + \ln y, \forall x, y \in \mathbf{R}^+$
3. $\ln(x/y) = \ln x - \ln y, \forall x, y \in \mathbf{R}^+$
4. $\ln x^p = p \ln x, \forall x \in \mathbf{Q}^+$
5. $\frac{d}{dx}(\ln |x|) = \frac{1}{x}, \forall x \in \mathbf{R} \wedge x \neq 0$
6. $\frac{d}{dx}(\ln |u(x)|) = \frac{u'(x)}{u(x)}$
7. $\int \frac{dx}{x} = \ln |x| + C$

The Number e

The number e is the real number that satisfies

$$\ln e = \int_1^e \frac{dt}{t} = 1 \quad (3)$$

The Exponential Function

$\forall x, y \in \mathbf{R}$

$$y = e^x \Leftrightarrow x = \ln y \quad (4)$$

Properties of e

- $e^{x+y} = e^x e^y$
- $e^{x-y} = e^x / e^y$
- $(e^x)^y = e^{xy}, \forall y \in \mathbf{Q}$
- $\ln(e^x) = x, \forall x \in \mathbf{R}$
- $e^{\ln x} = x, \forall x \in \mathbf{R}^+$

Exponential Functions with General Bases

Let $b \in \mathbf{R}^+ \wedge b \neq 1. \forall x \in \mathbf{R}^+$,

$$b^x = e^{x \ln b} \quad (5)$$

Derivative and Integral of the Exponential Function

$\forall x \in \mathbf{R}$,

$$\frac{d}{dx}(e^{u(x)}) = e^{u(x)} u'(x) \quad (6)$$

$$\int e^x dx = e^x + C \quad (7)$$

7.3 Logarithmic and Exponential Functions with Other Bases

Logarithmic Function Base b

For any base $b > 0$, with $b \neq 1$, the **logarithmic function base b** , denoted $\log_b x$, is the inverse of the exponential function b^x .

Inverse Relations for Exponential and Logarithmic Functions

For any base $b > 0$, with $b \neq 1$, the following inverse relation holds.

- $b^{\log_b x} = x, \forall x \in \mathbf{R}^+$
- $\log_b b^x = x, \forall x$

Derivative of b^x

If $b > 0 \wedge b \neq 1, \forall x$,

$$\frac{d}{dx}(b^x) = b^x \ln b \quad (8)$$

Indefinite Integral of b^x

For $b > 0 \wedge b \neq 1$,

$$\int b^x dx = \frac{1}{\ln b} b^x + C \quad (9)$$

General Power Rule

$\forall p, x \in \mathbf{R}^+$,

$$\frac{d}{dx}(x^p) = px^{p-1} \quad (10)$$

Furthermore, if u is a positive differentiable function on its domain, then

$$\frac{d}{dx}(u(x)^p) = p(u(x))^{p-1} \times u'(x) \quad (11)$$

Derivative of $\log_b x$

If $b > 1$,

$$\frac{d}{dx}(\log_b x) = \frac{1}{x \ln b} \quad \forall x \in \mathbf{R}^+ \wedge x \neq 0 \quad (12)$$

$$\frac{d}{dx}(\log_b |x|) = \frac{1}{x \ln b} \quad \forall x \in \mathbf{R}^+ \wedge x \neq 0 \quad (13)$$

7.5 Inverse Trigonometric Functions

Derivative of Inverse Trigonometric Functions

$$\begin{aligned}\frac{d}{dx}(\sin^{-1} x) &= \frac{1}{\sqrt{1-x^2}} & \frac{d}{dx}(\cos^{-1} x) &= -\frac{1}{\sqrt{1-x^2}}, \{x \in \mathbf{R} \mid -1 < x < 1\} \\ \frac{d}{dx}(\tan^{-1} x) &= \frac{1}{1+x^2} & \frac{d}{dx}(\cot^{-1} x) &= -\frac{1}{1+x^2}, \{x \in \mathbf{R} \mid -\infty < x < \infty\} \\ \frac{d}{dx}(\sec^{-1} x) &= \frac{1}{|x|\sqrt{x^2-1}} & \frac{d}{dx}(\csc^{-1} x) &= -\frac{1}{|x|\sqrt{x^2-1}}, \{x \in \mathbf{R} \mid |x| > 1\}\end{aligned}$$

Integrals Involving Inverse Trigonometric Functions

$$\int \frac{dx}{\sqrt{a^2 - x^2}} = \sin^{-1} \frac{x}{a} + C \quad (14)$$

$$\int \frac{dx}{a^2 + x^2} = \frac{1}{a} \tan^{-1} \frac{x}{a} + C \quad (15)$$

$$\int \frac{dx}{x\sqrt{a^2 + x^2}} = \frac{1}{a} \sec^{-1} \left| \frac{x}{a} \right| + C \quad (16)$$

7.6 L'Hôpital's Rule and Growth Rates of Functions

Indeterminate forms $1^\infty, 0^0, \infty^0$

Assume $\lim_{x \rightarrow \infty} f(x)^{g(x)}$ has the indeterminate form $1^\infty, 0^0$, or ∞^0 .

1. Evaluate $L = \lim_{x \rightarrow \infty} g(x) \ln f(x)$. This limit can be put in the form $0/0$ or ∞/∞ , both of which are handled by l'Hôpital's rule.
2. Then $\lim_{x \rightarrow \infty} f(x)^{g(x)} = e^L$

Growth Rates of Functions (as $x \rightarrow \infty$)

Suppose f and g are functions with $\lim_{x \rightarrow \infty} f(x) = \lim_{x \rightarrow \infty} g(x) = \infty$. Then f **grows faster than g** as $x \rightarrow \infty$ if

$$\lim_{x \rightarrow \infty} \frac{g(x)}{f(x)} = 0 \quad \text{or, quantitatively, if} \quad \lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = \infty \quad (17)$$

The functions f and g have **comparable growth rates** if

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = M$$

where $M \in \mathbb{R}^+$.

Ranking Growth Rates as $x \rightarrow \infty$

Let $f \ll g$ mean that g grows faster than f as $f \rightarrow \infty$. With positive real numbers p, q, r, s and $b > 1$,

$$\ln^q x \ll x^p \ll x^p \ln^r x \ll x^{p+s} \ll b^x \ll x^x \quad (18)$$

7.7 Hyperbolic Functions

Hyperbolic Functions

Hyperbolic Cosine

$$\cosh x = \frac{e^x + e^{-x}}{2} \quad (19)$$

Hyperbolic Sine

$$\sinh x = \frac{e^x - e^{-x}}{2} \quad (20)$$

Hyperbolic Tangent

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (21)$$

Hyperbolic Cotangent

$$\coth x = \frac{\cosh x}{\sinh x} = \frac{e^x + e^{-x}}{e^x - e^{-x}} \quad (22)$$

Hyperbolic Secant

$$\operatorname{sech} x = \frac{1}{\cosh x} = \frac{2}{e^x + e^{-x}} \quad (23)$$

Hyperbolic Cosecant

$$\operatorname{csch} x = \frac{1}{\sinh x} = \frac{2}{e^x - e^{-x}} \quad (24)$$

Hyperbolic Identities

$$\cosh^2 x - \sinh^2 x = 1$$

$$1 - \tanh^2 x = \operatorname{sech}^2 x$$

$$\coth^2 x - 1 = \operatorname{csch}^2 x$$

$$\coth(x + y) = \cosh x \cosh y + \sinh x \sinh y$$

$$\sinh(x + y) = \sinh x \cosh y + \cosh x \sinh y$$

$$\cosh 2x = \cosh^2 x + \sinh^2 x$$

$$\cosh^2 x = \frac{\cosh 2x + 1}{2}$$

$$\cosh(-x) = \cosh x$$

$$\sinh(-x) = -\sinh x$$

$$\tanh(-x) = -\tanh x$$

$$\sinh 2x = 2 \sinh x \cosh x$$

$$\sinh^2 x = \frac{\cosh 2x - 1}{2}$$

Derivatives and Integral Formulas

1. $\frac{d}{dx}(\cosh x) = \sinh x \Rightarrow \int \sinh x \, dx = \cosh x + C$
2. $\frac{d}{dx}(\sinh x) = \cosh x \Rightarrow \int \cosh x \, dx = \sinh x + C$
3. $\frac{d}{dx}(\tanh x) = \operatorname{sech}^2 x \Rightarrow \int \operatorname{sech}^2 x \, dx = \tanh x + C$
4. $\frac{d}{dx}(\coth x) = -\operatorname{csch}^2 x \Rightarrow \int \operatorname{csch}^2 x \, dx = -\coth x + C$
5. $\frac{d}{dx}(\operatorname{sech} x) = -\operatorname{sech} x \tanh x \Rightarrow \int \operatorname{sech} x \tanh x \, dx = -\operatorname{sech} x + C$
6. $\frac{d}{dx}(\operatorname{csch} x) = -\operatorname{csch} x \coth x \Rightarrow \int \operatorname{csch} x \coth x \, dx = -\operatorname{csch} x + C$

Integrals of Hyperbolic Functions

1. $\int \tanh x \, dx = \ln \cosh x + C$
2. $\int \coth x \, dx = \ln |\sinh x| + C$
3. $\int \operatorname{sech} x \, dx = \tan^{-1} |\sinh x| + C$
4. $\int \operatorname{csch} x \, dx = \ln |\tanh(x/2)| + C$

Inverses of the Hyperbolic Functions Expressed as Logarithms

$$\begin{aligned}\cosh^{-1} x &= \ln(x + \sqrt{x^2 - 1}) \quad (x \geq 1) & \operatorname{sech}^{-1} x &= \cosh^{-1} \frac{1}{x} \quad (0 < x \leq 1) \\ \sinh^{-1} x &= \ln(x + \sqrt{x^2 + 1}) & \operatorname{csch}^{-1} x &= \sinh^{-1} \frac{1}{x} \quad (x \neq 0) \\ \tanh^{-1} x &= \frac{1}{2} \ln\left(\frac{1+x}{1-x}\right) \quad (|x| < 1) & \coth^{-1} x &= \tanh^{-1} \frac{1}{x} \quad (|x| > 1)\end{aligned}$$

Derivatives of the Inverse Hyperbolic Functions

$$\begin{aligned}\frac{d}{dx}(\cosh^{-1} x) &= \frac{1}{\sqrt{x^2 - 1}} \quad (x > 1) & \frac{d}{dx}(\sinh^{-1} x) &= \frac{1}{\sqrt{x^2 + 1}} \\ \frac{d}{dx}(\tanh^{-1} x) &= \frac{1}{1 - x^2} \quad (|x| < 1) & \frac{d}{dx}(\coth^{-1} x) &= \frac{1}{1 - x^2} \quad (|x| > 1) \\ \frac{d}{dx}(\operatorname{sech}^{-1} x) &= -\frac{1}{x\sqrt{1 - x^2}} \quad (0 < x < 1) & \frac{d}{dx}(\operatorname{csch}^{-1} x) &= -\frac{1}{|x|\sqrt{1 + x^2}} \quad (x \neq 0)\end{aligned}$$

Integral Formulas

1. $\int \frac{dx}{\sqrt{x^2 - a^2}} = \cosh^{-1} \frac{x}{a} + C, \text{ for } x > a$
2. $\int \frac{dx}{x^2 + a^2} = \sinh^{-1} \frac{x}{a} + C, \text{ for all } x$

3. $\int \frac{dx}{a^2-x^2} = \frac{1}{a} \tanh^{-1} \frac{x}{a} + C$, for $|x| < a = \frac{1}{a} \coth^{-1} \frac{x}{a} + C$, for $|x| > a$
4. $\int \frac{dx}{x\sqrt{a^2-x^2}} = -\frac{1}{a} \operatorname{sech}^{-1} \frac{x}{a} + C$, for $0 < x < a$
5. $\int \frac{dx}{x\sqrt{a^2+x^2}} = -\frac{1}{a} \operatorname{csch}^{-1} \frac{|x|}{a} + C$, for $x \neq 0$

Chapter 8: Integration Techniques

Illya Starikov

August 6, 2025

Contents

8.1	Basic Approaches	2
8.2	Integration By Parts	3
8.3	Trigonometric Integrals	4
8.4	Trigonometric Substitutions	6
8.5	Partial Fractions	7
8.8	Improper Integrals	9

8.1 Basic Approaches

$$\int k \, dx = kx + C \quad (1)$$

$$\int k^p \, dx = \frac{k^{p+1}}{p+1} + C, p \in \mathbf{R} \wedge \neq -1 \quad (2)$$

$$\int \cos ax \, dx = \frac{1}{a} \sin ax + C \quad (3)$$

$$\int \sin ax \, dx = -\frac{1}{a} \cos ax + C \quad (4)$$

$$\int \sec^2 ax \, dx = \frac{1}{a} \tan ax + C \quad (5)$$

$$\int \csc^2 ax \, dx = -\frac{1}{a} \cot ax + C \quad (6)$$

$$\int \sec ax \tan ax \, dx = \frac{1}{a} \sec ax + C \quad (7)$$

$$\int \csc ax \cot ax \, dx = -\frac{1}{a} \csc ax \cot ax + C \quad (8)$$

$$\int e^{ax} \, dx = \frac{1}{a} e^{ax} + C \quad (9)$$

$$\int \frac{dx}{x} = \ln |x| + C \quad (10)$$

$$\int \frac{dx}{\sqrt{a^2 - x^2}} = \sin^{-1} \frac{x}{a} + C \quad (11)$$

$$\int \frac{dx}{a^2 + x^2} = \frac{1}{a} \tan^{-1} \frac{x}{a} + C \quad (12)$$

$$\int \frac{dx}{x\sqrt{x^2 - a^2}} = \frac{1}{a} \sec^{-1} \left| \frac{x}{a} \right| + C \quad (13)$$

8.2 Integration By Parts

Suppose that u and v are differentiable functions. Then

$$\int u \, dv = uv - \int v \, du \quad (14)$$

Integration by Parts for Definite Integrals

Let u and v be differentiable. Then

$$\int_a^b u(x)v'(x) \, dx = u(x)v(x) \Big|_a^b - \int_a^b v(x)u'(x) \, dx \quad (15)$$

Integral of $\ln x$

$$\int \ln x \, dx = x \ln x - x + C \quad (16)$$

8.3 Trigonometric Integrals

$\int \sin^m x \cos^n x \, dx$ **Strategy**

m is odd, n real Split off $\sin x$, rewrite the resulting even power of $\sin x$ in terms of $\cos x$, and then use $u = \cos x$

n odd, m real Split off $\cos x$, rewrite the resulting even power of $\cos x$ in terms of $\sin x$, and then use $u = \sin x$.

m and n both even, nonnegative Use half-angle identities to transform the integrand into a polynomial in $\cos 2x$, and apply the preceding strategies once again to powers of $\cos 2x$ greater than 1.

Reduction Formulas

Assume n is a positive integer.

$$\int \sin^n x \, dx = -\frac{\sin^{n-1} x \cos x}{n} + \frac{n-1}{n} \int \sin^{n-2} x \, dx \quad (17)$$

$$\int \cos^n x \, dx = \frac{\cos^{n-1} x \sin x}{n} + \frac{n-1}{n} \int \cos^{n-2} x \, dx \quad (18)$$

$$\int \tan^n x \, dx = \frac{\tan^{n-1} x}{n-1} - \int \tan^{n-2} x \, dx, n \neq 1 \quad (19)$$

$$\int \sec^n x \, dx = \frac{\sec^{n-2} x \tan x}{n-1} + \frac{n-2}{n-1} \int \sec^{n-2} x \, dx, n \neq 1 \quad (20)$$

Integrals of $\tan x$, $\cot x$, $\sec x$, and $\csc x$

$$\int \tan x \, dx = -\ln |\cos x| + C = \ln |\sec x| + C \quad (21)$$

$$\int \cot x \, dx = \ln |\sin x| + C \quad (22)$$

$$\int \sec x \, dx = \ln |\sec x + \tan x| + C \quad (23)$$

$$\int \csc x \, dx = -\ln |\csc x + \cot x| + C \quad (24)$$

$\int \tan^m x \sec^n x \, dx$ **Strategy**

n **even** Split off $\sec^2 x$, rewrite the remaining even power of $\sec x$ in terms of $\tan x$, and use $u = \tan x$.

m **odd** Split off $\sec x \tan x$, rewrite the remaining even power of $\tan x$ in terms of $\sec x$, and use $u = \sec x$.

m **even and** n **odd** Rewrite the even power of $\tan x$ in terms of $\sec x$ to produce a polynomial in $\sec x$; apply reduction formula 4 to each term.

8.4 Trigonometric Substitutions

The Integral Contains...	Corresponding Substitution	Useful Identity
$a^2 - x^2$	$x = a \sin \theta, -\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}, \forall x \leq a$	$a^2 - a^2 \sin^2 \theta = a^2 \cos^2 \theta$
$a^2 + x^2$	$x = a \tan \theta, -\frac{\pi}{2} < \theta < \frac{\pi}{2}$	$a^2 + a^2 \tan^2 \theta = a^2 \sec^2 \theta$
$x^2 - a^2$	$x = a \sec \theta,$	$a^2 \sec^2 \theta - a^2 = a^2 \tan^2 \theta$

$$\begin{cases} 0 \leq \theta < \frac{\pi}{2}, \forall x \geq a \\ \frac{\pi}{2} < \theta \leq \pi, \forall x \leq -a \end{cases}$$

8.5 Partial Fractions

Partial Fractions with Simple Linear Factors

Suppose $f(x) = p(x) > q(x)$, where p and q are polynomials with no common factors and with the degree of p less than the degree of q . Assume that q is the product of simple linear factors. The partial fraction decomposition is obtained as follows.

Step 1. Factor the denominator q in the form $(x - r_1)(x - r_2) \cdots (x - r_n)$, where $r_1, \dots, r_n$ are real numbers.

Step 2. Partial fraction decomposition Form the partial fraction decomposition by writing

$$\frac{p(x)}{q(x)} = \frac{A_1}{x - r_1} + \frac{A_2}{x - r_2} + \cdots + \frac{A_n}{x - r_n} \quad (25)$$

Step 3. Clear denominators Multiply both sides of the equation in Step 2 by $q(x) = (x - r_1)(x - r_2) \cdots (x - r_n)$, which produces conditions for $A_1, \dots, A_n$.

Step 4. Solve for coefficients Equate like powers of x in Step 3 to solve for the undetermined coefficients $A_1, \dots, A_n$.

Partial Fractions For Repeated Linear Factors

Suppose the repeated linear factor $(x - r)^m$ appears in the denominator of a proper rational function in reduced form. The partial fraction decomposition has a partial fraction for each power of $(x - r)$ up to and including the m th power; that is, the partial fraction decomposition contains the sum

$$\frac{A_1}{(x - r)} + \frac{A_2}{(x - r)^2} + \frac{A_3}{(x - r)^3} + \cdots + \frac{A_m}{(x - r)^m} \quad (26)$$

where $A_1, \dots, A_m$ are constants to be determined.

Partial Fractions with Simple Irreducible Quadratic Factors

Suppose a simple irreducible factor $ax^2 + bx + c$ appears in the denominator of a proper rational function in reduced form. The partial fraction decomposition contains a term of the form

$$\frac{Ax + B}{ax^2 + bx + c} \quad (27)$$

where A and B are unknown coefficients to be determined.

Partial Fraction Decomposition

Let $f(x) = p(x)/q(x)$ be a proper rational function in reduced form. Assume the denominator q has been factored completely over the real numbers and m is a positive integer.

Simple linear factor A factor $x - r$ in the denominator requires the partial fraction $\frac{A}{x - r}$.

Repeated linear factor A factor $(x - r)^m$ with $m > 1$ in the denominator requires the partial fractions.

$$\frac{A_1}{(x - r)} + \frac{A_2}{(x - r)^2} + \frac{A_3}{(x - r)^3} + \cdots + \frac{A_m}{(x - r)^m} \quad (28)$$

Simple irreducible quadratic factor An irreducible factor $ax^2 + bx + c$ in the denominator requires the partial fraction

$$\frac{Ax + B}{ax^2 + bx + c} \quad (29)$$

Repeated irreducible quadratic factor An irreducible factor $(ax^2 + bx + c)^m$ with $m > 1$ in the denominator requires the partial fractions

$$\frac{A_1x + B_1}{ax^2 + bx + c} + \frac{A_2x + B_2}{(ax^2 + bx + c)^2} + \cdots + \frac{A_mx + B_m}{(ax^2 + bx + c)^m} \quad (30)$$

8.8 Improper Integrals

Improper Integrals over Infinite Intervals

1. If f is continuous on $[a, \infty)$, then

$$\int_a^\infty f(x) dx = \lim_{b \rightarrow \infty} \int_a^b f(x) dx \quad (31)$$

provided the limit exists.

2. If f is continuous on $(-\infty, b]$, then

$$\int_{-\infty}^b f(x) dx = \lim_{a \rightarrow -\infty} \int_a^b f(x) dx \quad (32)$$

provided the limit exists.

3. If f is continuous on $(-\infty, \infty)$, then

$$\int_{-\infty}^\infty f(x) dx = \lim_{a \rightarrow -\infty} \int_a^c f(x) dx + \lim_{b \rightarrow \infty} \int_c^b f(x) dx \quad (33)$$

provided both limits exist, where c is any real number.

In each case, if the limit exists, the improper integral is said to **converge**, if it does not exist, the improper integral is said to **diverge**.

Improper Integrals with an Unbounded Integrand

1. Suppose f is continuous on $(a, b]$ with $\lim_{x \rightarrow a^+} f(x) = \pm \infty$. Then

$$\int_a^b f(x) dx = \lim_{c \rightarrow a^+} \int_c^b f(x) dx \quad (34)$$

provided the limit exists.

2. Suppose f is continuous on $[a, b)$ with $\lim_{x \rightarrow b^-} f(x) = \pm \infty$. Then

$$\int_a^b f(x) dx = \lim_{c \rightarrow b^-} \int_a^c f(x) dx \quad (35)$$

provided the limit exists.

3. Suppose f is continuous on $[a, b]$ except at the interior point p where f is unbounded. Then

$$\int_a^b f(x) dx = \int_a^p f(x) dx + \int_p^b f(x) dx, \quad (36)$$

provided the improper integrals on the right side exist.

In each case, if the limit exists, the improper integrals is said to **converge**, if it does not exists, the improper integral is said to **diverge**.

Chapter 9: Sequences and Infinite Series

Illya Starikov

August 6, 2025

Contents

9.1	An Overview	2
9.2	Sequences	4
9.3	Infinite Series	6
9.4	The Divergence and Integral Tests	7
9.5	The Ratio, Root, and Comparison Tests	9
9.6	Alternating Series	11

9.1 An Overview

Sequence

A **sequence** $\{a_n\}$ is an ordered list of numbers of the form

$$\{a_1, a_2, a_3, \dots, a_n, \dots\} \quad (1)$$

A sequence may be generated by a **recurrence relations** of the form $a_{n+1} = f(a_n)$, for $n = 1, 2, 3, \dots$, where a_1 is given. A sequence may also be defined with an **explicit form** of the form $a_n = f(n)$, for $n = 1, 2, 3, \dots$

Limit of a Sequence

If the terms of a sequence $\{a_n\}$ approach a unique number L as n increases, then we say $\lim_{n \rightarrow \infty} a_n = L$ exists, and the sequence **converges** to L . If the terms of the sequence do not approach a single number as n increases, the sequence has no limits, and the sequence **diverges**.

Infinite Series

Given a set of numbers $\{a_1, a_2, a_3, \dots\}$, the sum

$$a_1 + a_2 + a_3 + \dots = \sum_{k=1}^{\infty} a_k \quad (2)$$

is called an **infinity series**. Its **sequence of partial sums** $\{S_n\}$ has the terms

$$S_1 = a_1 \quad (3)$$

$$S_2 = a_1 + a_2 \quad (4)$$

$$S_3 = a_1 + a_2 + a_3 \quad (5)$$

$$\vdots \quad (6)$$

$$S_n = a_1 + a_2 + a_3 + \dots + a_n = \sum_{k=1}^n a_k, \quad n = 1, 2, 3, \dots \quad (7)$$

If the sequence of partial sums $\{S_n\}$ has a limit L , the infinite series **converges** to that limits, and we write

$$\sum_{k=1}^{\infty} a_k = \lim_{n \rightarrow \infty} \sum_{k=1}^n a_k = \lim_{n \rightarrow \infty} S_n = L \quad (8)$$

If the sequence of partial sums diverges, the infinite series also **diverges**.

9.2 Sequences

Limits of Sequences from Limits of Functions

Suppose f is a function such that $f(n) = a_n$ for all positive integers n . If $\lim_{n \rightarrow \infty} f(n) = L$, then the limits of the sequences $\{a_n\}$ is also L .

Properties of Limits of Sequences

Assume that the sequence $\{a_n\}$ and $\{b_n\}$ have limits A and B , respectively. Then,

1. $\lim_{n \rightarrow \infty} (a_n \pm b_n) = A \pm B$
2. $\lim_{n \rightarrow \infty} ca_n = cA$, where c is a real number
3. $\lim_{n \rightarrow \infty} a_nb_n = AB$
4. $\lim_{n \rightarrow \infty} \frac{a_n}{b_n} = \frac{A}{B}$, provided $B \neq 0$.

Geometric Sequences

Let r be a real number. Then,

$$\lim_{n \rightarrow \infty} r^n = \begin{cases} 0 & \Leftrightarrow |r| < 1 \\ 1 & \Leftrightarrow r = 1 \\ \text{does not exist} & \Leftrightarrow r \leq -1 \vee r > 1 \end{cases} \quad (9)$$

Squeeze Theorem for Sequences

Let $\{a_n\}$, $\{b_n\}$, and $\{c_n\}$ be sequences with $a_n \leq b_n \leq c_n$ for all n greater than some index N . If $\lim_{n \rightarrow \infty} a_n = \lim_{n \rightarrow \infty} c_n = L$, then $\lim_{n \rightarrow \infty} b_n = L$.

Bounded Monotonic Sequences

A bounded monotonic sequence converges.

Growth Rates of Sequences

The following sequences are ordered according to increasing growth rates as $n \rightarrow \infty$; that is, if $\{a_n\}$ appears before $\{b_n\}$ in the list, then $\lim_{n \rightarrow \infty} \frac{a_n}{b_n} = 0$ and $\lim_{n \rightarrow \infty} \frac{b_n}{a_n} = \infty$

$$\{\ln^q n\} \ll \{n^p\} \ll \{n^p \ln^r n\} \ll \{n^{p+s}\} \ll \{b^n\} \ll \{n!\} \ll \{n^n\} \quad (10)$$

The ordering applies for $p, q, r, s, b \in \mathbb{R}^+ \wedge b > 1$.

Limit of a Sequence

The sequence $\{a_n\}$ converges to L provided the terms of a_n can be made arbitrarily close to L by taking n sufficiently large. More precisely, $\{a_n\}$ has the unique limit L if given any tolerance $\epsilon > 0$, it is possible to find a positive integer N (depending only on ϵ) such that

$$|a_n - L| < \epsilon \quad \text{whenever } n > N \quad (11)$$

if the **limit of a sequence** is L , we say the sequence **converges** to L , written

$$\lim_{n \rightarrow \infty} a_n = L \quad (12)$$

A sequence that does not converge is said to **diverge**.

9.3 Infinite Series

Geometric Series

$$\sum_{k=0}^{n-1} ar^k = S_n = a \frac{1 - r^n}{1 - r} \quad (13)$$

Geometric Series

Let $a \neq 0$ and r be real numbers. If $|r| < 1$, then $\sum_{k=0}^{\infty} ar^k = \frac{a}{1-r}$. If $|r| \geq 1$, then the series diverges.

9.4 The Divergence and Integral Tests

Divergence Test

If $\sum a_k$ converges, then $\lim_{n \rightarrow \infty} a_k = 0$. Equivalently, if $\lim_{n \rightarrow \infty} a_k \neq 0$, then the series diverges. However, this cannot be used to prove convergence. If $\lim_{n \rightarrow \infty} a_k = 0$, the test is inconclusive.

Harmonic Series

The harmonic series $\sum_{k=1}^{\infty} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \cdots$ diverges, even though the terms of the series approach zero.

Integral Test

Suppose f is a continuous, positive, decreasing function, for $x \geq 1$, and let $a_k = f(k)$, for $k = 1, 2, 3, \dots$. Then

$$\sum_{k=1}^{\infty} a_k \quad \text{and} \quad \int_1^{\infty} f(x) dx \quad (14)$$

either both converge or both diverge. In the case of convergence, the value of the integral is *not*, in general, equal to the value of the series.

Convergence of the p -Series

The p -Series $\sum_{k=1}^{\infty} \frac{1}{k^p}$ converges, for $p > 1$, and diverges for $p \leq 1$.

Estimating Series with Positive Terms

Let f be continuous, positive, decreasing function, for $x \geq 1$, and let $a_k = f(k)$, for $k = 1, 2, 3, \dots$. Let $S = \sum_{k=1}^{\infty} a_k$ be a convergence series and let $S_n = \sum_{k=1}^n a_k$ be the sum of the first n terms of the series. The remainder $R_n = S - S_n$ satisfies

$$R_n \leq \int_n^{\infty} f(x) dx \quad (15)$$

Furthermore, the exact value of the series is bounded as follows:

$$S_n + \int_{n+1}^{\infty} f(x) dx \leq \sum_{k=1}^{\infty} a_k \leq S_n + \int_n^{\infty} f(x) dx. \quad (16)$$

Properties of Convergent Series

1. Suppose $\sum a_k$ converges to A and let c be a real number. The series $\sum ca_k$ converges and $\sum ca_k = c \sum a_k = cA$
2. Suppose $\sum a_k$ converges to A and $\sum b_k$ converges to B . The series $\sum(a_k \pm b_k)$ converges and $\sum(a_k \pm b_k) = \sum a_k \pm \sum b_k = A \pm B$
3. *Whether* a series converges does not depend on a finite number of terms added to or removed from the series. Specifically, if M is a positive integer, then $\sum_{k=1}^{\infty} a_k$ and $\sum_{k=M}^{\infty} a_k$ both converge or both diverge. However, the *value* of a convergent series does change if nonzero terms are added or deleted.

9.5 The Ratio, Root, and Comparison Tests

Useful Identities

$$\lim_{k \rightarrow \infty} \left(1 + \frac{1}{k}\right)^k = \lim_{k \rightarrow \infty} \left(\frac{k+1}{k}\right)^k = e \quad (17)$$

$$\lim_{k \rightarrow \infty} k^{\frac{1}{k}} = 1 \quad (18)$$

The Ratio Test

Let $\sum a_k$ be an infinite series with positive terms and let $r = \lim_{n \rightarrow \infty} \frac{a_{k+1}}{a_k}$.

1. If $0 \leq r < 1$, the series converges.
2. If $r > 1$ (including $r = \infty$), the series diverges.
3. If $r = 1$, the test is inconclusive.

The Root Test

Let $\sum a_k$ be an infinite series with nonnegative terms and let $\rho = \lim_{n \rightarrow \infty} \sqrt[k]{a_k}$.

1. If $0 \leq \rho < 1$, the series converges.
2. If $\rho > 1$ (including $\rho = \infty$), the series diverges.
3. If $\rho = 1$, test is inconclusive.

The Comparison Test

Let $\sum a_k$ and $\sum b_k$ be a series with positive terms.

1. If $0 < a_k \leq b_k$ and $\sum b_k$ converges, then $\sum a_k$ converges.
2. If $0 < b_k \leq a_k$ and $\sum b_k$ diverges, then $\sum a_k$ diverges.

The Limit Comparison Test

Let $\sum a_k$ and $\sum b_k$ be series with positive terms and let

$$\lim_{k \rightarrow \infty} \frac{a_k}{b_k} = L \quad (19)$$

- If $0 < L < \infty$ (that is, L is a finite, positive number), then $\sum a_k$ and $\sum b_k$ either both converge or both diverge.
- If $L = 0$ and $\sum b_k$ converges, then $\sum a_k$ converges.
- If $L = \infty$ and $\sum b_k$ diverges, then $\sum a_k$ diverges.

Guidelines

- Begin with the Divergence Test. If you show that $\lim_{n \rightarrow \infty} a_n \neq 0$, then the series diverges and your work is finished.
- Is the series a special series? Recall the convergence properties for the following series:
 - Geometric series: $\sum ar^k$ converges for $|r| < 1$ and diverges for $|r| \geq 1$ ($a \neq 0$).
 - p -series: $\sum \frac{1}{k^p}$ converges for $p > 1$, and diverges for $p \leq 1$.
 - Check also for telescoping series.
- If the general k th term of the series looks like a function you can integrate, then try the Integral Test.
- If the general k th term of the series involves $k!$, k^k , or a^k , where a is a constant, the Ratio Test is advisable. Series with k in an exponent may yield to the Root Test.
- If the general k th term of the series is a rational function of k (or a root of a rational function), use the Comparison or the Limit Comparison Test. Use the families of series given in Step 2 as comparison series.

9.6 Alternating Series

The Alternating Series Test

The alternating series $\sum (-1)^{k+1} a_k$ converges provided

1. the terms of the series are non-increasing in magnitude ($0 < a_{k+1} \leq a_k$, for k greater than some index N) and
2. $\lim_{n \rightarrow \infty} a_n = 0$

Alternating Harmonic Series

The alternating harmonic series $\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \dots$ converges (even though the harmonic series $\sum_{k=1}^{\infty} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \dots$ diverges).

Remainder in Alternating Series

Let $R_n = |S - S_n|$ be the remainder in approximating the value of a convergent alternating series $\sum_{k=1}^{\infty} (-1)^{k+1} a_k$ by the sum of its first n terms. Then $R_n \leq a_{n+1}$. In other words, the remainder is less than or equal to the magnitude of the first neglected term.

Absolute and Conditional Convergence

Assume the infinite series $\sum a_k$ converges. The series $\sum a_k$ **converges absolutely** if the series $\sum |a_k|$ converges. Otherwise, the series $\sum a_k$ **converges conditionally**.

Absolute Convergence Implies Convergence

If $\sum |a_k|$ converges, then $\sum a_k$ converges (absolute convergence implies convergence). If $\sum a_k$ diverges, then $\sum |a_k|$ diverges.

Series or Test	Form of Series	Condition for Convergence	Condition for Divergence	Comments
Geometric Series	$\sum_{k=0}^{\infty} ar^k, a \neq 0$	$ r < 1$	$ r \geq 1$	If $ r < 1$, then $\sum_{k=1}^{\infty} ar^k = \frac{a}{1-r}$
Divergence Test	$\sum_{k=1}^{\infty} a_k$	Does Not Apply	$\lim_{k \rightarrow \infty} a_k \neq 0$	Cannot be used to prove convergence.
Integral Test	$\sum_{k=1}^{\infty} a_k$, where $a_k = f(k)$ and f is continuous, positive, and decreasing.	$\int_1^{\infty} f(x)dx < \infty$	$\int_1^{\infty} f(x)dx$ does not exist	The value of the integral is not the value of the series.
p -Series	$\sum_{k=1}^{\infty} \frac{1}{k^p}$	$p > 1$	$p \leq 1$	Useful for comparison tests.
Ratio Test	$\sum_{k=1}^{\infty} a_k^{\infty}$, where $a_k > 0$	$\lim_{k \rightarrow \infty} \frac{a_{k+1}}{a_k} < 1$	$\lim_{k \rightarrow \infty} \frac{a_{k+1}}{a_k} > 1$	Inconclusive if $\lim_{k \rightarrow \infty} \frac{a_{k+1}}{a_k} = 1$
Root Test	$\sum_{k=1}^{\infty} a_k^{\infty}$, where $a_k \geq 0$	$\lim_{k \rightarrow \infty} \sqrt[k]{a_k} < 1$	$\lim_{k \rightarrow \infty} \sqrt[k]{a_k} > 1$	Inconclusive if $\lim_{k \rightarrow \infty} \sqrt[k]{a_k} = 1$
Comparison Test	$\sum_{k=1}^{\infty} a_k^{\infty}$, where $a_k > 0$	$0 < a_k \leq b_k$ and $\sum_{k=1}^{\infty} b_k$ converges	$0 < b_k \leq a_k$ and $\sum_{k=1}^{\infty} b_k$ diverges	$\sum_{k=1}^{\infty} a_k^{\infty}$ is given, you supply $\sum_{k=1}^{\infty} b_k$
Limit Comparison Test	$\sum_{k=1}^{\infty} a_k^{\infty}$, where $a_k > 0, b_k > 0$	$0 \leq \lim_{k \rightarrow \infty} \frac{a_k}{b_k} < \infty$ and $\sum_{k=1}^{\infty} b_k$ converges	$\lim_{k \rightarrow \infty} \frac{a_k}{b_k} > 0$ and $\sum_{k=1}^{\infty} b_k$ diverges.	$\sum_{k=1}^{\infty} a_k^{\infty}$ is given, you supply $\sum_{k=1}^{\infty} b_k$
Alternating Series Test	$\sum_{k=1}^{\infty} (-1)^k a_k$, where $a_k > 0, 0 < a_{k+1} \leq a_k$	$\lim_{k \rightarrow \infty} a_k = 0$	$\lim_{k \rightarrow \infty} a_k \neq 0$	Remainder R_n satisfies $R_n \leq a_{n+1}$
Absolute Convergence	$\sum_{k=1}^{\infty} a_k^{\infty}$, a_k arbitrary	$\sum_{k=1}^{\infty} a_k $ converges.	Applies to arbitrary series	

Chapter 10: Power Series

Illya Starikov

August 6, 2025

Contents

10.1 Approximating Functions with Polynomials	2
10.2 Properties of Power Series	4
10.3 Taylor Series	6

10.1 Approximating Functions with Polynomials

Taylor Polynomials

Let f be a function with $f', f'', \dots, f^{(n)}$ defined at a . The **n th-order Taylor polynomial** for f with its **center** at a , denoted p_n , has the property that it matches f in value, slope, and all derivatives up to the n th derivative at a ; that is,

$$p_n(a) = f(a), p'_n(a) = f'(a), \dots, p_n^{(n)}(a) = f^{(n)}(a). \quad (1)$$

The n th-order Taylor polynomial centered at a is

$$p_n(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \dots + \frac{f^{(n)}(a)}{n!}(x-a)^n. \quad (2)$$

More compactly, $p_n(x) = \sum_{k=0}^n c_k(x-a)^k$, where the **coefficients** are

$$c_k = \frac{f^{(k)}(a)}{k!}, \quad \text{for } k = 0, 1, 2, \dots, n \quad (3)$$

Remainder in a Taylor Polynomial

Let p_n be the Taylor polynomial of order n for f . The **remainder** in using p_n to approximate f at the point x is

$$R_n(x) = f(x) - p_n(x) \quad (4)$$

Taylor's Theorem

Let f have continuous derivatives up to $f^{(n+1)}$ on an open interval I containing a . For all x in I ,

$$f(x) = p_n(x) + R_n(x), \quad (5)$$

where p_n is the n th-order Taylor polynomial for f centered at a , and the remainder is

$$R_n(x) = \frac{f^{(n+1)}(c)}{(n+1)!}(x-a)^{(n+1)}, \quad (6)$$

for some point c between x and a .

Estimate of the Remainder

Let n be a fixed positive integer. Suppose there exists a number M such that $|f^{(n+1)}(c)| \leq M$, for all c between a and x inclusive. The remainder in the n th-order Taylor polynomial for f centered at a satisfies

$$|R_n(x)| = |f(x) - p_n(x)| \leq M \frac{|x - a|^{n+1}}{(n+1)!} \quad (7)$$

10.2 Properties of Power Series

Power Series

A **power series** has the general form

$$\sum_{k=0}^{\infty} c_k(x-a)^k \quad (8)$$

where a and c_k are real numbers, and x is a variable. The c_k 's are the **coefficients** of the power series and a is the **center** of the power series. The set of values of x for which the series converges is its **interval of convergence**. The **radius of convergence** of the power series, denoted R , is the distance from the center of the series to the boundary of the interval of convergence.

Convergence of Power Series

A power series $\sum_{k=0}^{\infty} c_k(x-a)^k$ centered at a converges in one of three ways:

1. The series converges absolutely for all x , in which case the interval of convergence is $(-\infty, \infty)$ and the radius of convergence is $R = \infty$.
2. There is a real number $R > 0$ such that the series converges absolutely for $|x-a| < R$ and diverges for $|x-a| > R$, in which case the radius of convergence is R .
3. The series converges only at a , in which case the radius of convergence is $R = 0$.

Combining Power Series

Suppose the power series $\sum c_k x^k$ and $\sum d_k x^k$ converge absolutely to $f(x)$ and $g(x)$, respectively, on an interval I .

1. **Sum and difference:** The power series $\sum (c_k \pm d_k)x^k$ converges absolutely to $f(x) \pm g(x)$ on I .
2. **Multiplication by a power:** The power series $x^m \sum c_k x^k = \sum c_k x^{k+m}$ converges absolutely to $x^m f(x)$ on I , provided m is an integer such that $k+m \geq 0$ for all terms of the series.

3. **Composition:** If $h(x) = bx^m$, where m is a positive integer and b is a real number, the power series $\sum c_k(h(x))^k$ converges absolutely to the composite function $f(h(x))$, for all x such that $h(x)$ is in I .

Differentiating and Integrating Power Series

Let the function f be defined by the power series $\sum c_k(x - a)^k$ on its interval of convergence I .

- f is a continuous function on I .
- The power series may be differentiated or integrated term by term, and the resulting power series converges to $f'(x)$ or $\int f(x) dx + C$, respectively, at all points in the interior of I , where C is an arbitrary constant.

10.3 Taylor Series

Taylor/Maclaurin Series for a Function

Suppose the function f has derivatives of all orders on an interval containing the point a . The **Taylor series for f centered at a** is

$$f(a) + f'(a)(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \frac{f^{(3)}(a)}{3!}(x - a)^3 + \cdots \quad (9)$$

$$= \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x - a)^k. \quad (10)$$

A Taylor series centered at 0 is called a **Maclaurin series**.

Binomial Coefficients

$$\forall p, k \in \mathbb{R} \wedge k \geq 1$$

$$\binom{p}{k} = \frac{p(p-1)(p-2)\cdots(p-k+1)}{k!}. \quad (11)$$

With the special case of $\binom{p}{0} = 1$.

Binomial Series

$\forall p \in \mathbb{R} \wedge p \neq 0$, the Taylor series for $f(x) = (1+x)^p$ centered at 0 is the **binomial series**

$$\sum_{k=0}^{\infty} x^k = \sum_{k=0}^{\infty} \frac{p(p-1)(p-2)\cdots(p-k+1)}{k!} x^k \quad (12)$$

$$= 1 + px + \frac{p(p-1)(p-2)}{3!} x^3 + \cdots \quad (13)$$

The series converges for $|x| < 1$ (and possibly at the endpoints, depending on p). If p is a nonnegative integer, the series terminates and results in a polynomial of degree p .

Convergence of Taylor Series

Let f have derivatives of all orders on an open interval I containing a . The Taylor series for f centered at a converges to f , for all x in I , if and only if $\lim_{n \rightarrow \infty} R_n(x) = 0$, for all x in I , where

$$R_n(x) = \frac{f^{(n+1)}(c)}{(n+1)!} (x-a)^{n+1} \quad (14)$$

is the remainder at x (with c between x and a).

Taylor Series Functions

Chapter 11: Parametric and Polar Curves

Illya Starikov

August 7, 2025

Contents

11.1 Parametric Equations	2
11.2 Polar Coordinates	3
11.3 Calculus in Polar Coordinates	11

11.1 Parametric Equations

Forward or Positive Orientation

The direction in which a parametric curve is generated as the parameter increases is called the **forward**, or **positive, orientation** of the curve.

Derivative for Parametric Curves

Let $x = g(t)$ and $y = h(t)$, where g and h are differentiable on an interval $[a, b]$. Then

$$\frac{dy}{dx} = \frac{dy/dt}{dx/dt} = \frac{y'(t)}{x'(t)} \quad (1)$$

provided $\frac{dx}{dt} \neq 0$.

11.2 Polar Coordinates

Introduction

- Up to now we have only studied in a Cartesian coordinate system.
 - A Cartesian coordinate system is just a plane described by Cartesian (or, algebraic) equations and points in a finite dimensions.
 - * *One Dimension*: Lines.
 - * *Two Dimensions*: x^2 .
 - * *Three, Four*: Upper-level Calculus and Physics.
- Let's define an alternative coordinate system — **polar coordinate**.
 - coordinates are constants on circles and rays.
 - Useful for navigation, position, and gravitation fields.

Defining Polar Coordinates

Pole The origin of the coordinate system.

Polar Axis Synonymous for the positive x -axis.

Polar Coordinates A polar coordinates P has the form (r, θ) .

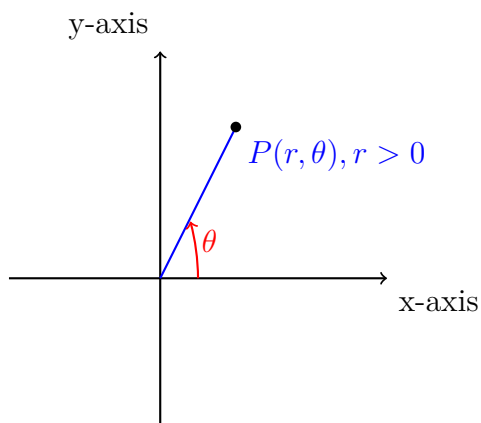
Radial Coordinate The radial coordinate r describes the *signed*, or *directed*, distance from the origin to P .

Angular Coordinate The angular coordinate θ describes an angle whose initial side is the positive x -axis and whose terminal side lies on the ray passing through the origin and P .

Notes

- **Positive angle measurements are measured *counterclockwise* from the origin.**
- Every point has multiple representations.
 - Angles are periodic, so multiples of 2π gives the same angle.

- Coordinates may be negative. So (r, θ) can be represented as $(-r, \theta + \pi)$ and $(-r, \theta - \pi)$



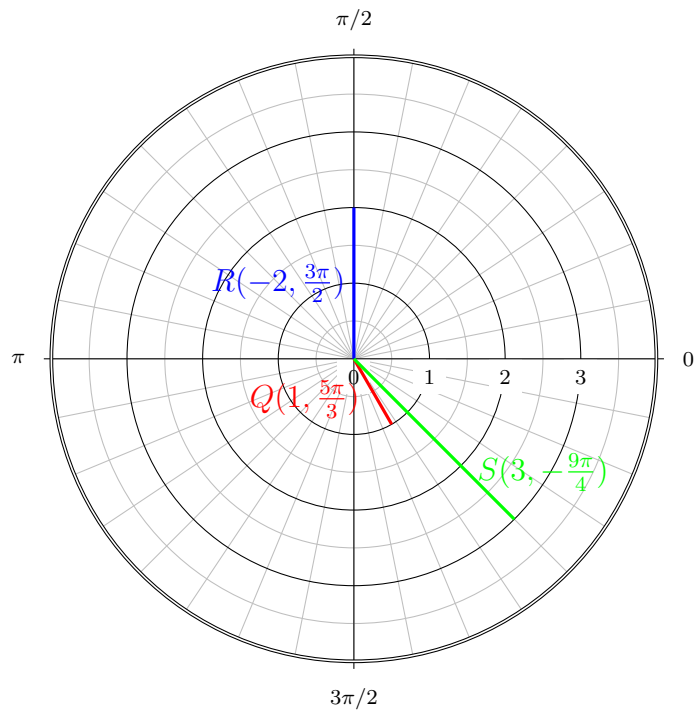
In summary,

- $(r, \theta + 2\pi)$ represents the same point as (r, θ)
- $P(r, \theta)$ and $P'(-r, \theta)$ are reflections through the origin.

Examples

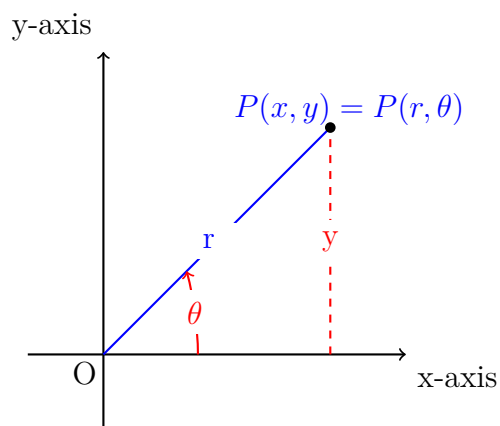
Graph the following points in polar coordinates:

- $Q(1, \frac{5\pi}{3})$
- $R(-2, \frac{3\pi}{2})$
- $S(3, -\frac{9\pi}{4})$
 - Now give two alternative representations.
 - $S'(3, \frac{1\pi}{4})$
 - $S''(-3, -\frac{5\pi}{4})$



Converting Between Cartesian and Polar Coordinates

- We sometimes need to convert between Cartesian and polar coordinates.
- Let's turn this problem into a right triangle.



A point with polar coordinates (r, θ) has Cartesian coordinates (x, y) , where

$$x = r \cos \theta \quad \text{and} \quad y = r \sin \theta \quad (2)$$

A point with Cartesian coordinates (x, y) has polar coordinates (r, θ) , where

$$r^2 = x^2 + y^2 \quad \text{and} \quad \tan \theta = \frac{y}{x} \quad (3)$$

Examples

BE SURE TO GRAPH POINTS IN CARTESIAN FIRST.

Polar to Cartesian

Express the point with the following polar coordinates in Cartesian coordinates: $P(3, \frac{2\pi}{3})$

$$\begin{aligned} x &= r \cos \theta & y &= r \sin \theta \\ &= 3 \cos(2\pi/3) & &= 3 \sin(2\pi/3) \\ &= -3(1/2) & &= 3(\sqrt{3}/2) \\ &= -3/2 & &= 3\sqrt{3}/2 \end{aligned}$$

Polar to Cartesian

Express the point with the following polar coordinates in Cartesian coordinates: $Q(e, -\frac{\pi}{4})$

$$\begin{aligned} x &= r \cos \theta & y &= r \sin \theta \\ &= e \cos(-\pi/4) & &= e \sin(-\pi/4) \\ &= e(\sqrt{2}/2) & &= -e(\sqrt{2}/2) \\ &= e\sqrt{2}/2 & &= -e\sqrt{2}/2 \end{aligned}$$

Cartesian To Polar

Express the point with the following Polar coordinates to Cartesian coordinates: $R(1, -1)$

$$\begin{aligned} r &= \sqrt{x^2 + y^2} & \tan \theta &= y/x \\ &= \sqrt{1^2 + (-1)^2} & &= -1/1 \\ &= \sqrt{2} & &= -1 \\ & & \theta &= -\pi/4 \text{ or } 7\pi/4. \end{aligned}$$

Therefore, two possible solutions are: $(\sqrt{2}, -\pi/4)$ or $(\sqrt{2}, 7\pi/4)$

Cartesian To Polar

Express the point with the following Polar coordinates to Cartesian coordinates: $S(1, \sqrt{3})$

$$\begin{aligned} r &= \sqrt{x^2 + y^2} & \tan \theta &= y/x \\ &= \sqrt{(\sqrt{3})^2 + (1)^2} & &= \sqrt{3}/1 \\ &= 2 & \theta &= \pi/3 \text{ or } 4\pi/3. \end{aligned}$$

Therefore, two possible solutions are: $(2, \pi/3)$ or $(2, 4\pi/3)$.

Basic Curves in Polar Coordinates

- A curve in polar coordinates is the set of **points** that satisfy an equation in r and θ .
- This makes graphing some things easier than others.
- Look at $r = 3$ is the set of all points that satisfy being away from the origin of 3 units.
 - This is because θ is not specified, it's arbitrary. Basically, θ is the function.

- In general, $r = a, \forall a \in \mathbb{R}^+$ describes a circle.
- Taking the converse, let r be arbitrary.
 - If the r is arbitrary, and we specify the angle, what do you think we get?
 - A line!
 - Take $\sqrt{2}/2$.

Polar to Cartesian Graph Example

Convert the polar equation $r = 6 \sin \theta$ to Cartesian coordinates and describe the corresponding graph.

$$r^2 = 6r \sin \theta \quad (4)$$

$$x^2 + y^2 = 6y \quad (5)$$

$$0 = x^2 + y^2 - 6y \quad (6)$$

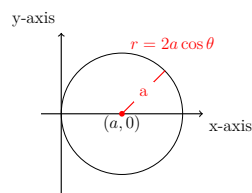
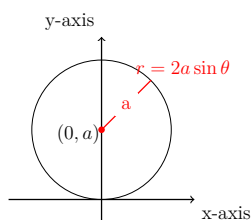
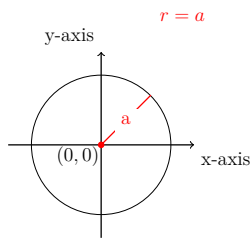
$$= x^2 + (y^2 - 6y + 9) - 9 \quad (7)$$

$$= x^2 + (y - 3)^2 - 9 \quad (8)$$

We recognize this to be the equation of a circle, centered at $(0, 3)$ at 3. We can also generalize this.

Circle in Polar Coordinates

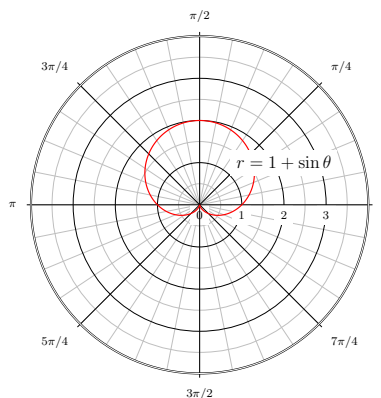
- The equation $r = a$ describes a circle of radius $|a|$ centered at $(0, 0)$.
- The equation $r = 2a \sin \theta$ describes a circle of radius $|a|$ centered at $(0, a)$.
- The equation $r = 2a \cos \theta$ describes a circle of radius $|a|$ centered at $(a, 0)$.



Graphing In Polar Coordinates

Graph the polar equation $r = f(\theta) = 1 + \sin \theta$

θ	$r = 1 + \sin \theta$
0	1
$\pi/6$	$3/2$
$\pi/2$	2
$5\pi/6$	$3/2$
π	1
$7\pi/6$	$1/2$
$3\pi/2$	0
$11\pi/6$	$1/2$
2π	1



The resulting curve is known as a **cardioid**.

Cartesian-to-Polar Method for Graphing $r = f(\theta)$

1. Graph $r = f(\theta)$ as if r and θ were Cartesian coordinates with θ on the horizontal axis and r on the vertical axis. Be sure to choose an interval in θ on which the entire polar curve is produced.
2. Use the Cartesian graph in Step 1 as a guide to sketch the points (r, θ) on the final *polar* curve.

Example

With the alternate graphing method, graph $r = 1 + \sin \theta$.

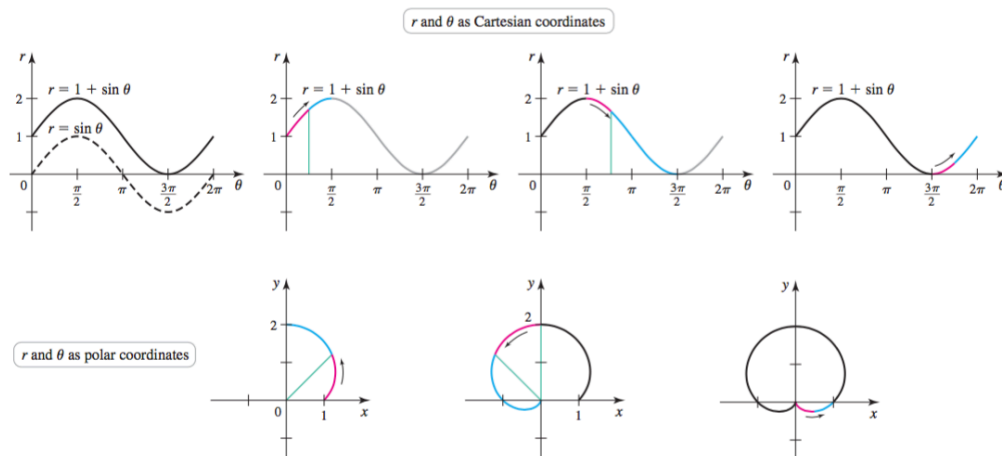


FIGURE 11.26

Symmetry In Polar Equations

Symmetry about the x-axis occurs if the point (r, θ) is on the graph whenever $(r, -\theta)$ is on the graph.

Symmetry about the y-axis occurs if the point (r, θ) is on the graph whenever $(r, \pi - \theta) = (-r, -\theta)$ is on the graph.

Symmetry about the origin occurs if the point (r, θ) is on the graph whenever $(-r, \theta) = (r, \theta + \pi)$ is on the graph.

11.3 Calculus in Polar Coordinates

Slope of a Tangent Line

Let f be a differentiable function at θ_0 . The slope of the line tangent to the curve $r = f(\theta)$ at the point $(f(\theta_0), \theta_0)$ is

$$\frac{dy}{dx} = \frac{f'(\theta_0) \sin \theta_0 + f(\theta_0) \cos \theta_0}{f'(\theta_0) \cos \theta_0 - f(\theta_0)} \quad (9)$$

Area of Regions in Polar Coordinates

Let R be the region bounded by the graphs of $r = f(\theta)$ and $r = g(\theta)$, between $\theta = \alpha$ and $\theta = \beta$, where f and g are continuous and $f(\theta) \geq g(\theta) \geq 0$ on $[\alpha, \beta]$. The area of R is

$$\int_{\alpha}^{\beta} \frac{1}{2} (f(\theta)^2 - g(\theta)^2) d\theta \quad (10)$$

11.2 Polar Coordinates

Illya Starikov

August 7, 2025

11.2 Polar Coordinates

Introduction

- Up to now we have only studied in a Cartesian coordinate system.
 - A Cartesian coordinate system is just a plane described by Cartesian (or, algebraic) equations and points in a finite dimensions.
 - * *One Dimension*: Lines.
 - * *Two Dimensions*: x^2 .
 - * *Three, Four*: Upper-level Calculus and Physics.
- Let's define an alternative coordinate system — **polar coordinate**.
 - coordinates are constants on circles and rays.
 - Useful for navigation, position, and gravitation fields.

Defining Polar Coordinates

Pole The origin of the coordinate system.

Polar Axis Synonymous for the positive x -axis.

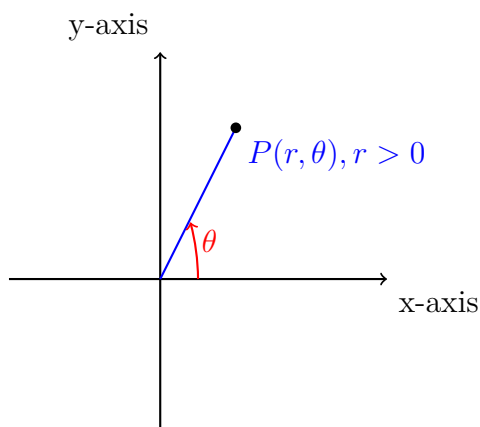
Polar Coordinates A polar coordinates P has the form (r, θ) .

Radial Coordinate The radial coordinate r describes the *signed*, or *directed*, distance from the origin to P .

Angular Coordinate The angular coordinate θ describes an angle whose initial side is the positive x -axis and whose terminal side lies on the ray passing through the origin and P .

Notes

- **Positive angle measurements are measured *counterclockwise* from the origin.**
- Every point has multiple representations.
 - Angles are periodic, so multiples of 2π gives the same angle.
 - Coordinates may be negative. So (r, θ) can be represented as $(-r, \theta + \pi)$ and $(-r, \theta - \pi)$



In summary,

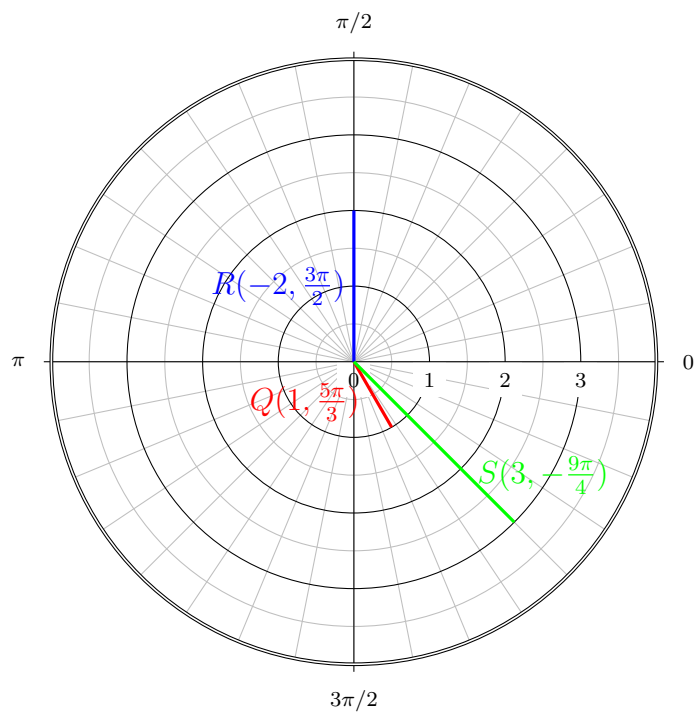
- $(r, \theta + 2\pi)$ represents the same point as (r, θ)
- $P(r, \theta)$ and $P'(-r, \theta)$ are reflections through the origin.

Examples

Graph the following points in polar coordinates:

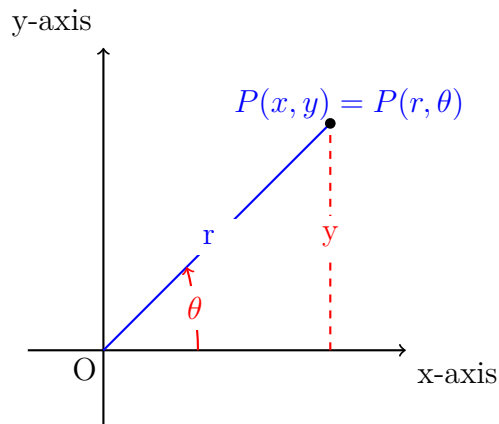
- $Q(1, \frac{5\pi}{3})$
- $R(-2, \frac{3\pi}{2})$
- $S(3, -\frac{9\pi}{4})$
 - Now give two alternative representations.

- $S'(3, \frac{1\pi}{4})$
- $S''(-3, -\frac{5\pi}{4})$



Converting Between Cartesian and Polar Coordinates

- We sometimes need to convert between Cartesian and polar coordinates.
- Let's turn this problem into a right triangle.



A point with polar coordinates (r, θ) has Cartesian coordinates (x, y) , where

$$x = r \cos \theta \quad \text{and} \quad y = r \sin \theta \quad (1)$$

A point with Cartesian coordinates (x, y) has polar coordinates (r, θ) , where

$$r^2 = x^2 + y^2 \quad \text{and} \quad \tan \theta = \frac{y}{x} \quad (2)$$

Examples

BE SURE TO GRAPH POINTS IN CARTESIAN FIRST.

Polar to Cartesian

Express the point with the following polar coordinates in Cartesian coordinates: $P(3, \frac{2\pi}{3})$

$$\begin{aligned} x &= r \cos \theta & y &= r \sin \theta \\ &= 3 \cos(2\pi/3) & &= 3 \sin(2\pi/3) \\ &= -3(1/2) & &= 3(\sqrt{3}/2) \\ &= -3/2 & &= 3\sqrt{3}/2 \end{aligned}$$

Polar to Cartesian

Express the point with the following polar coordinates in Cartesian coordinates: $Q(e, -\frac{\pi}{4})$

$$\begin{aligned}x &= r \cos \theta & y &= r \sin \theta \\&= e \cos(-\pi/4) & &= e \sin(-\pi/4) \\&= e(\sqrt{2}/2) & &= -e(\sqrt{2}/2) \\&= e\sqrt{2}/2 & &= -e\sqrt{2}/2\end{aligned}$$

Cartesian To Polar

Express the point with the following Polar coordinates to Cartesian coordinates: $R(1, -1)$

$$\begin{aligned}r &= \sqrt{x^2 + y^2} & \tan \theta &= y/x \\&= \sqrt{1^2 + (-1)^2} & &= -1/1 \\&= \sqrt{2} & &= -1 \\& & \theta &= -\pi/4 \text{ or } 7\pi/4.\end{aligned}$$

Therefore, two possible solutions are: $(\sqrt{2}, -\pi/4)$ or $(\sqrt{2}, 7\pi/4)$

Cartesian To Polar

Express the point with the following Polar coordinates to Cartesian coordinates: $S(1, \sqrt{3})$

$$\begin{aligned}r &= \sqrt{x^2 + y^2} & \tan \theta &= y/x \\&= \sqrt{(\sqrt{3})^2 + (1)^2} & &= \sqrt{3}/1 \\&= 2 & \theta &= \pi/3 \text{ or } 4\pi/3.\end{aligned}$$

Therefore, two possible solutions are: $(2, \pi/3)$ or $(2, 4\pi/3)$.

Basic Curves in Polar Coordinates

- A curve in polar coordinates is the set of **points** that satisfy an equation in r and θ .
- This makes graphing some things easier than others.
- Look at $r = 3$ is the set of all points that satisfy being away from the origin of 3 units.
 - This is because θ is not specified, it's arbitrary. Basically, θ is the function.
 - In general, $r = a, \forall a \in \mathbb{R}^+$ describes a circle.
- Taking the converse, let r be arbitrary.
 - If the r is arbitrary, and we specify the angle, what do you think we get?
 - A line!
 - Take $\sqrt{2}/2$.

Polar to Cartesian Graph Example

Convert the polar equation $r = 6 \sin \theta$ to Cartesian coordinates and describe the corresponding graph.

$$r^2 = 6r \sin \theta \quad (3)$$

$$x^2 + y^2 = 6y \quad (4)$$

$$0 = x^2 + y^2 - 6y \quad (5)$$

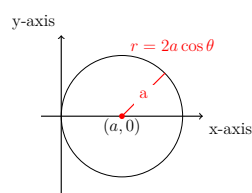
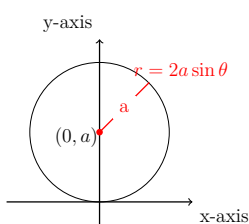
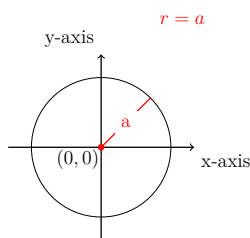
$$= x^2 + (y^2 - 6y + 9) - 9 \quad (6)$$

$$= x^2 + (y - 3)^2 - 9 \quad (7)$$

We recognize this to be the equation of a circle, centered at $(0, 3)$ at 3. We can also generalize this.

Circle in Polar Coordinates

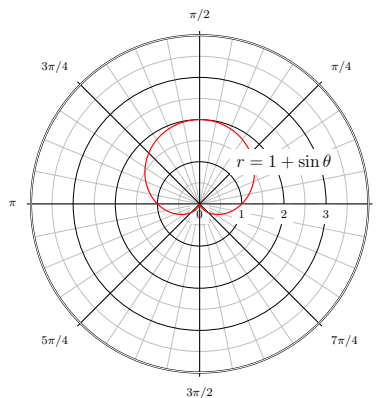
- The equation $r = a$ describes a circle of radius $|a|$ centered at $(0, 0)$.
- The equation $r = 2a \sin \theta$ describes a circle of radius $|a|$ centered at $(0, a)$.
- The equation $r = 2a \cos \theta$ describes a circle of radius $|a|$ centered at $(a, 0)$.



Graphing In Polar Coordinates

Graph the polar equation $r = f(\theta) = 1 + \sin \theta$

θ	$r = 1 + \sin \theta$
0	1
$\pi/6$	$3/2$
$\pi/2$	2
$5\pi/6$	$3/2$
π	1
$7\pi/6$	$1/2$
$3\pi/2$	0
$11\pi/6$	$1/2$
2π	1



The resulting curve is known as a **cardioid**.

Cartesian-to-Polar Method for Graphing $r = f(\theta)$

1. Graph $r = f(\theta)$ as if r and θ were Cartesian coordinates with θ on the horizontal axis and r on the vertical axis. Be sure to choose an interval in θ on which the entire polar curve is produced.

- Use the Cartesian graph in Step 1 as a guide to sketch the points (r, θ) on the final *polar* curve.

Example

With the alternate graphing method, graph $r = 1 + \sin \theta$.

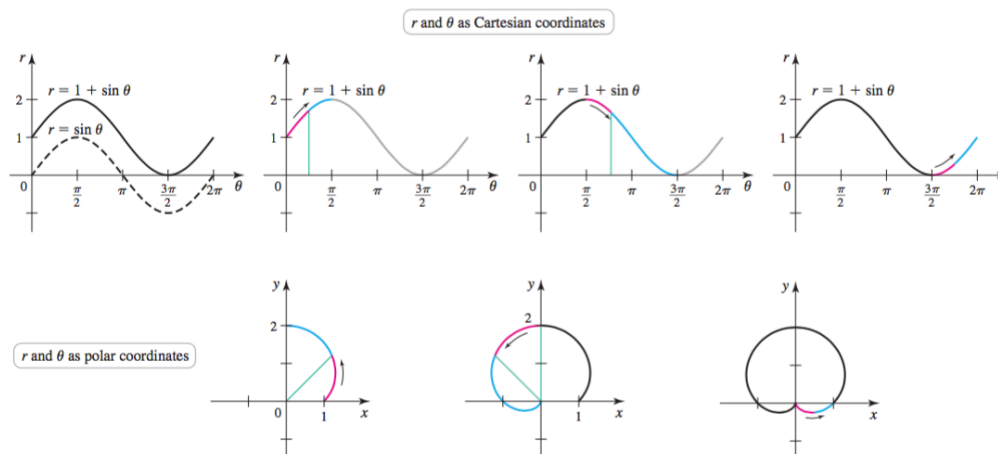


FIGURE 11.26

Symmetry In Polar Equations

Symmetry about the x-axis occurs if the point (r, θ) is on the graph whenever $(r, -\theta)$ is on the graph.

Symmetry about the y-axis occurs if the point (r, θ) is on the graph whenever $(r, \pi - \theta) = (-r, -\theta)$ is on the graph.

Symmetry about the origin occurs if the point (r, θ) is on the graph whenever $(-r, \theta) = (r, \theta + \pi)$ is on the graph.

MATH2100

Foundations Of Mathematics



Chapter `_num`₁

- We talked about using rational numbers as a model for our axiom systems.
 - Rationals work, upto Axiom 6.
 - Suppose we take π as the dividing step. The hypothesis is now satisfied, but either S_1 would not have a top level or S_2 would not have a bottom level.

MATH2222

Calculus III



Chapter 12: Vectors and Vector Valued Functions

Illya Starikov

August 6, 2025

Contents

12.1 Vectors in the Plane	2
12.2 Vectors in Three Dimensions	4
12.3 Dot Product	5
12.4 Cross Product	7
12.5 Lines and Curves in Space	9
12.6 Calculus of Vector-Valued Functions	10
12.7 Motion In Space	12
12.8 Length of Curves	14
12.9 Curvature and Normal Vectors	15

12.1 Vectors in the Plane

Vectors, Equal Vectors, Scalars, Zero Vector

Vectors are quantities that have both **length** (or **magnitude**) and **direction**. Two vectors are **equal** if they have the same magnitude and direction. Quantities having magnitude but no direction are called **scalars**. One exception is the **zero** vector, denoted **0**: It has length 0 and no direction.

Scalar Multiples and Parallel Vectors

Given a scalar c and a vector $\mathbf{u}$, the scalar multiple $c\mathbf{v}$ is a vector whose magnitude is $|c|$ multiplied by the magnitude of $\mathbf{v}$. If $c > 0$, then $c\mathbf{v}$ has the same direction as $\mathbf{v}$. If $c < 0$, then $c\mathbf{v}$ and $\mathbf{v}$ point in opposite directions. Two vectors are **parallel** if they are scalar multiples of each other.

Position Vectors and Vector Components

A vector $\mathbf{v}$ with its tail at the origin and head at the point (v_1, v_2) is called a **position vector** (or is said to be in **standard position**) and is written $\langle v_1, v_2 \rangle$. The real numbers v_1 and v_2 are the **x-** and **y-components** of $\mathbf{v}$, respectively. The position vectors $\mathbf{u} = \langle u_1, u_2 \rangle$ and $\mathbf{v} = \langle v_1, v_2 \rangle$ are equal if and only if $u_1 = v_1$ and $u_2 = v_2$.

Magnitude of a Vector

Given the points $P(x_1, y_1)$ and $Q(x_2, y_2)$, the **magnitude**, or **length**, of $\vec{PQ} = \langle x_2 - x_1, y_2 - y_1 \rangle$, denoted $|\vec{PQ}|$, is the distance between P and Q :

$$|\vec{PQ}| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (1)$$

The magnitude of the position vector $\mathbf{v} = \langle v_1, v_2 \rangle$ is $|\mathbf{v}| = \sqrt{v_1^2 + v_2^2}$

Vector Operations

Suppose c is a scalar, $\mathbf{u} = \langle u_1, u_2 \rangle$ and $\mathbf{v} = \langle v_1, v_2 \rangle$.

$$\mathbf{u} + \mathbf{v} = \langle u_1 + v_1, u_2 + v_2 \rangle \quad \text{Vector addition} \quad (2)$$

$$\mathbf{u} - \mathbf{v} = \langle u_1 - v_1, u_2 - v_2 \rangle \quad \text{Vector subtraction} \quad (3)$$

$$c\mathbf{u} = \langle cu_1, cu_2 \rangle \quad \text{Scalar multiplication} \quad (4)$$

Unit Vectors and Vectors of a Specified Length

A **unit vector** is any vector with length 1. Given a nonzero vector $\mathbf{v}$, $\pm \frac{\mathbf{v}}{|\mathbf{v}|}$ are unit vectors parallel to $\mathbf{v}$. For a scalar $c > 0$, the vectors $\pm \frac{c\mathbf{v}}{|v|}$ are vectors of length c parallel to $\mathbf{v}$.

Properties of Vector Operations

Suppose $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ are vectors and a and c are scalars. Then the following properties hold (for vectors in any number of dimensions).

$$\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u} \quad \text{Commutative property of addition} \quad (5)$$

$$(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w}) \quad \text{Associative property of addition} \quad (6)$$

$$\mathbf{v} + \mathbf{0} = \mathbf{v} \quad \text{Additive identity} \quad (7)$$

$$\mathbf{v} + (-\mathbf{v}) = \mathbf{0} \quad \text{Additive identity} \quad (8)$$

$$c(\mathbf{u} + \mathbf{v}) = c\mathbf{u} + c\mathbf{v} \quad \text{Distributive property 1} \quad (9)$$

$$(a + c)\mathbf{v} = a\mathbf{v} + c\mathbf{v} \quad \text{Distributive property 2} \quad (10)$$

$$0\mathbf{v} = \mathbf{0} \quad \text{Multiplication by zero scalar} \quad (11)$$

$$c\mathbf{0} = \mathbf{0} \quad \text{Multiplication by zero vector} \quad (12)$$

$$1\mathbf{v} = \mathbf{v} \quad \text{Multiplicative identity} \quad (13)$$

$$a(c\mathbf{v}) = (ac)\mathbf{v} \quad \text{Associative property of scalar multiplication} \quad (14)$$

12.2 Vectors in Three Dimensions

Distance Formula in xyz -Space

The distance between the points $P(x_1, y_1, z_1)$ and $Q(x_2, y_2, z_2)$ is

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad (15)$$

Spheres and Balls

A **sphere** centered at (a, b, c) with radius r is the set of points satisfying the equation

$$(x - a)^2 + (y - b)^2 + (z - c)^2 = r^2 \quad (16)$$

A ball centered at (a, b, c) with radius r is the set of points satisfying the inequality

$$(x - a)^2 + (y - b)^2 + (z - c)^2 \leq r^2 \quad (17)$$

Vector Operations in $\mathbb{R}$

Let c be a scalar, $\mathbf{u} = \langle u_1, u_2, u_3 \rangle$ and $\mathbf{v} = \langle v_1, v_2, v_3 \rangle$.

$$\mathbf{u} + \mathbf{v} = \langle u_1 + v_1, u_2 + v_2, u_3 + v_3 \rangle \quad \text{Vector addition} \quad (18)$$

$$\mathbf{u} - \mathbf{v} = \langle u_1 - v_1, u_2 - v_2, u_3 - v_3 \rangle \quad \text{Vector subtraction} \quad (19)$$

$$c\mathbf{u} = \langle cu_1, cu_2, cu_3 \rangle \quad (20)$$

Magnitude of a Vector

The **magnitude** (or **length**) of the vector $\vec{PQ} = \langle x_2 - x_1, y_2 - y_1, z_2 - z_1 \rangle$ is the distance from $P(x_1, y_1, z_1)$ to $Q(x_2, y_2, z_2)$

$$|\vec{PQ}| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad (21)$$

12.3 Dot Product

Dot Product

Given two nonzero vectors $\mathbf{u}$ and $\mathbf{v}$ in two or three dimensions, their **dot product** is

$$\mathbf{u} \cdot \mathbf{v} = |\mathbf{u}| |\mathbf{v}| \cos \theta \quad (22)$$

where θ is the angle between $\mathbf{u}$ and $\mathbf{v}$ with $0 \leq \theta \leq \pi$. If $\mathbf{u} = \mathbf{0}$ or $\mathbf{v} = \mathbf{0}$, then $\mathbf{u} \cdot \mathbf{v} = 0$, and θ is undefined.

Orthogonal Vectors

Two vectors $\mathbf{u}$ and $\mathbf{v}$ are **orthogonal** if and only if $\mathbf{u} \cdot \mathbf{v} = 0$. The zero vector is orthogonal to all vectors. In two or three dimensions, two nonzero orthogonal vectors are perpendicular to each other.

Dot Product

Given two vectors $\mathbf{u} = \langle u_1, u_2, u_3 \rangle$ and $\mathbf{v} = \langle v_1, v_2, v_3 \rangle$,

$$\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + u_2 v_2 + u_3 v_3 \quad (23)$$

Properties of the Dot Product

Suppose $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ are vectors and let c be a scalar.

$$\mathbf{u} \cdot \mathbf{v} = \mathbf{v} \cdot \mathbf{u} \quad \text{Commutative property} \quad (24)$$

$$c(\mathbf{u} \cdot \mathbf{v}) = (c\mathbf{u}) \cdot \mathbf{v} = \mathbf{u} \cdot (c\mathbf{v}) \quad \text{Associative property} \quad (25)$$

$$\mathbf{u} \cdot (\mathbf{v} + \mathbf{w}) = \mathbf{u} \cdot \mathbf{v} + \mathbf{u} \cdot \mathbf{w} \quad (26)$$

(Orthogonal) Projection of $\mathbf{u}$ onto $\mathbf{v}$

The **orthogonal projection of $\mathbf{u}$ onto $\mathbf{v}$** , denoted $\text{proj}_{\mathbf{v}} \mathbf{u}$, where $\mathbf{v} \neq \mathbf{0}$, is

$$\text{proj}_{\mathbf{v}} \mathbf{u} = |\mathbf{u}| \cos \theta \left(\frac{\mathbf{v}}{|\mathbf{v}|} \right) \quad (27)$$

The orthogonal projection may also be computed with the formulas

$$\text{proj}_{\mathbf{v}} \mathbf{u} = \text{scal}_{\mathbf{v}} \mathbf{u} \left(\frac{\mathbf{v}}{|\mathbf{v}|} \right) = \left(\frac{\mathbf{u} \cdot \mathbf{v}}{\mathbf{v} \cdot \mathbf{v}} \right) \mathbf{v} \quad (28)$$

where the **scalar component of $\mathbf{u}$ in the direction of $\mathbf{v}$** is

$$\text{scal}_{\mathbf{v}} \mathbf{u} = |\mathbf{u}| \cos \theta = \frac{\mathbf{u} \cdot \mathbf{v}}{|\mathbf{v}|} \quad (29)$$

Work

Let a constant force $\mathbf{F}$ be applied to an object, producing a displacement $\mathbf{d}$. If the angle between $\mathbf{F}$ and $\mathbf{d}$ is θ , then the **work** done by the force is

$$W = |\mathbf{F}| |\mathbf{d}| \cos \theta = \mathbf{F} \cdot \mathbf{d} \quad (30)$$

12.4 Cross Product

Given two nonzero vectors $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{R}^3$, the **cross product** $\mathbf{u} \times \mathbf{v}$ is a vector with magnitude

$$|\mathbf{u} \times \mathbf{v}| = |\mathbf{u}||\mathbf{v}| \sin \theta \quad (31)$$

where $0 \leq \theta \leq \pi$ is the angle between $\mathbf{u}$ and $\mathbf{v}$. The direction of $\mathbf{u} \times \mathbf{v}$ is given by the **right-hand rule**: When you put the vectors tail to tail and let the fingers of your right hand curl from $\mathbf{u}$ to $\mathbf{v}$ the direction of $\mathbf{u} \times \mathbf{v}$ is the direction of your thumb, orthogonal to both $\mathbf{u}$ and $\mathbf{v}$. When $\mathbf{u} \times \mathbf{v} = 0$, the direction of $\mathbf{u} \times \mathbf{v}$ is undefined.

Geometry of the Cross Product

Let $\mathbf{u}$ and $\mathbf{v}$ be two nonzero vectors in $\mathbb{R}^3$.

1. The vectors $\mathbf{u}$ and $\mathbf{v}$ are parallel ($\theta = 0$ or $\theta = \pi$) if and only if $\mathbf{u} \times \mathbf{v} = 0$.
2. If $\mathbf{u}$ and $\mathbf{v}$ are two sides of a parallelogram, then the area of the parallelogram is

$$|\mathbf{u} \times \mathbf{v}| = |\mathbf{u}||\mathbf{v}| \sin \theta \quad (32)$$

Properties of the Cross Product

Let $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ be nonzero vectors in $\mathbb{R}^3$, and let a and b be scalars.

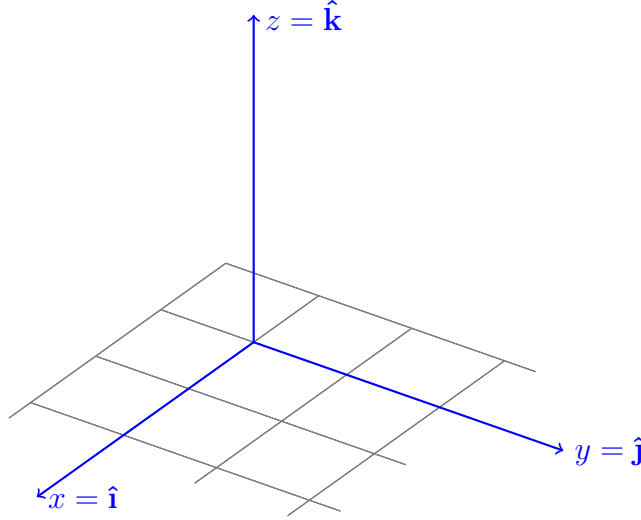
$$\mathbf{u} \times \mathbf{v} = -(\mathbf{v} \times \mathbf{u}) \quad \text{Anticommutative property} \quad (33)$$

$$(a\mathbf{u}) \times (b\mathbf{v}) = ab(\mathbf{u} \times \mathbf{v}) \quad \text{Associative property} \quad (34)$$

$$\mathbf{u} \times (\mathbf{v} + \mathbf{w}) = (\mathbf{u} \times \mathbf{v}) + (\mathbf{u} \times \mathbf{w}) \quad \text{Distributive property} \quad (35)$$

$$(\mathbf{u} + \mathbf{v}) \times \mathbf{w} = (\mathbf{u} \times \mathbf{w}) + (\mathbf{v} \times \mathbf{w}) \quad \text{Distributive property} \quad (36)$$

Cross Products of Coordinate Unit Vectors



$$\hat{\mathbf{i}} \times \hat{\mathbf{j}} = -(\hat{\mathbf{j}} \times \hat{\mathbf{i}}) = \hat{\mathbf{k}} \quad (37)$$

$$\hat{\mathbf{j}} \times \hat{\mathbf{k}} = -(\hat{\mathbf{k}} \times \hat{\mathbf{j}}) = \hat{\mathbf{i}} \quad (38)$$

$$\hat{\mathbf{k}} \times \hat{\mathbf{i}} = -(\hat{\mathbf{i}} \times \hat{\mathbf{k}}) = \hat{\mathbf{j}} \quad (39)$$

$$\hat{\mathbf{i}} \times \hat{\mathbf{i}} = \hat{\mathbf{j}} \times \hat{\mathbf{j}} = \hat{\mathbf{k}} \times \hat{\mathbf{k}} = 0 \quad (40)$$

Evaluating the Cross Product

Let $\mathbf{u} = u_1\hat{\mathbf{i}} + u_2\hat{\mathbf{j}} + u_3\hat{\mathbf{k}}$ and $\mathbf{v} = v_1\hat{\mathbf{i}} + v_2\hat{\mathbf{j}} + v_3\hat{\mathbf{k}}$. Then

$$\mathbf{u} \times \mathbf{v} = \begin{vmatrix} \hat{\mathbf{i}} & \hat{\mathbf{j}} & \hat{\mathbf{k}} \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{vmatrix} = \begin{vmatrix} u_2 & u_3 \\ v_2 & v_3 \end{vmatrix} \hat{\mathbf{i}} - \begin{vmatrix} u_1 & u_3 \\ v_1 & v_3 \end{vmatrix} \hat{\mathbf{j}} + \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix} \hat{\mathbf{k}} \quad (41)$$

12.5 Lines and Curves in Space

Equation of a Line

An equation of the line passing through the point $P_0(x_0, y_0, z_0)$ in the direction of the vector $\mathbf{v} = \langle a, b, c \rangle$ is $\mathbf{r} = \mathbf{r}_0 + t\mathbf{v}$, or

$$\langle x, y, z \rangle = \langle x_0, y_0, z_0 \rangle + t\langle a, b, c \rangle, \quad \text{for } -\infty < t < \infty \quad (42)$$

Equivalently, the parametric equations of the line are

$$x = x_0 + at, \quad y = y_0 + bt, \quad z = z_0 + ct, \quad \text{for } -\infty < t < \infty \quad (43)$$

Limit of a Vector-Valued Function

A vector-valued function $\mathbf{r}$ approaches the limit $\mathbf{L}$ as t approaches a , written $\lim_{t \rightarrow a} \mathbf{r}(t) = \mathbf{L}$, provided $\lim_{t \rightarrow a} |\mathbf{r}(t) - \mathbf{L}| = 0$

12.6 Calculus of Vector-Valued Functions

Derivative and Tangent Vector

Let $\mathbf{r}(t) = f(t)\hat{\mathbf{i}} + g(t)\hat{\mathbf{j}} + h(t)\hat{\mathbf{k}}$, where f , g , and h are differentiable functions on (a, b) . Then $\mathbf{r}$ has a **derivative** (or is **differentiable**) on (a, b) and

$$\mathbf{r}'(t) = f'(t)\hat{\mathbf{i}} + g'(t)\hat{\mathbf{j}} + h'(t)\hat{\mathbf{k}} \quad (44)$$

Provided $\mathbf{r}'(t) \neq \mathbf{0}$, $\mathbf{r}'(t)$ is a **tangent vector** (or velocity vector) at the point corresponding to $\mathbf{r}$.

Unit Tangent Vector

Let $\mathbf{r} = f(t)\hat{\mathbf{i}} + g(t)\hat{\mathbf{j}} + h(t)\hat{\mathbf{k}}$ be a smooth parameterized curve, for $a \leq t \leq b$. The **unit tangent vector** for a particular value of t is

$$\mathbf{T}(t) = \frac{\mathbf{r}'(t)}{|\mathbf{r}'(t)|} \quad (45)$$

Derivative Rules

Let $\mathbf{u}$ and $\mathbf{v}$ be differentiable vector-valued functions and let f be a differentiable scalar-valued function, all at a point t . Let $\mathbf{c}$ be a constant vector. The following rules apply.

$$\frac{d}{dt}(\mathbf{c}) = \mathbf{0} \quad \text{Constant Rule} \quad (46)$$

$$\frac{d}{dt}(\mathbf{u}(t) + \mathbf{v}(t)) = \mathbf{u}'(t) + \mathbf{v}'(t) \quad \text{Sum Rule} \quad (47)$$

$$\frac{d}{dt}(f(t)\mathbf{u}(t)) = f'(t)\mathbf{u}(t) + f(t)\mathbf{u}'(t) \quad \text{Product Rule} \quad (48)$$

$$\frac{d}{dt}(\mathbf{u}(f(t))) = \mathbf{u}'(f(t))f'(t) \quad \text{Chain Rule} \quad (49)$$

$$\frac{d}{dt}(\mathbf{u}(t) \cdot \mathbf{v}(t)) = \mathbf{u}'(t) \cdot \mathbf{v}(t) + \mathbf{u}(t) \cdot \mathbf{v}'(t) \quad \text{Dot Product Rule} \quad (50)$$

$$\frac{d}{dt}(\mathbf{u}(t) \times \mathbf{v}(t)) = \mathbf{u}'(t) \times \mathbf{v}(t) + \mathbf{u}(t) \times \mathbf{v}'(t) \quad \text{Cross Product Rule} \quad (51)$$

Indefinite Integral of a Vector-Valued Function

Let $\mathbf{r} = f\hat{\mathbf{i}} + g\hat{\mathbf{j}} + h\hat{\mathbf{k}}$ be a vector function and let $\mathbf{R} = F\hat{\mathbf{i}} + G\hat{\mathbf{j}} + H\hat{\mathbf{k}}$, where F , G , and H are antiderivatives of f , g , and h , respectively. The **indefinite integral** of $\mathbf{r}$ is

$$\int \mathbf{r}(t) dt = \mathbf{R}(t) + \mathbf{C} \quad (52)$$

where $\mathbf{C}$ is an arbitrary constant vector.

Definite Integral of a Vector-Valued Function

Let $\mathbf{r}(t) = f(t)\hat{\mathbf{i}} + g(t)\hat{\mathbf{j}} + h(t)\hat{\mathbf{k}}$, where f , g , and h are integrable on the interval $[a, b]$.

$$\int \mathbf{r}(t) dt = \left[\int_a^b f(t) dt \right] \hat{\mathbf{i}} + \left[\int_a^b g(t) dt \right] \hat{\mathbf{j}} + \left[\int_a^b h(t) dt \right] \hat{\mathbf{k}} \quad (53)$$

12.7 Motion In Space

Position, Velocity, Speed, Acceleration

Let the position of an object moving in three-dimensional space be given by $\mathbf{r}(t) = \langle x(t), y(t), z(t) \rangle$, for $t \geq 0$. The **velocity** of the object is

$$\mathbf{v}(t) = \mathbf{r}'(t) = \langle x'(t), y'(t), z'(t) \rangle \quad (54)$$

The **speed** of the object is the scalar function

$$|\mathbf{v}(t)| = \sqrt{x'(t)^2 + y'(t)^2 + z'(t)^2} \quad (55)$$

The **acceleration** of the object is $\mathbf{a}(t) = \mathbf{v}'(t) = \mathbf{r}''(t)$.

Motion with Constant $|\mathbf{r}|$

Let $\mathbf{r}$ describe a path on which $|\mathbf{r}|$ is constant (motion on a circle or sphere centered at the origin). Then, $\mathbf{r} \cdot \mathbf{v} = 0$, which means the position vector and the velocity vector are orthogonal at all times for which the functions are defined.

Two-Dimensional Motion in a Gravitational Field

Consider an object moving in a plane with horizontal x -axis and a vertical y -axis, subject only to the force of gravity. Given the initial velocity $\mathbf{v}(0) = \langle u_0, v_0 \rangle$ and the initial position $\mathbf{r}(0) = \langle x_0, y_0 \rangle$, the velocity of the object, for $t \geq 0$, is

$$\mathbf{v}(t) = \langle x'(t), y'(t) \rangle = \langle u_0, -gt + v_0 \rangle \quad (56)$$

and the position is

$$\mathbf{r}(t) = \langle x(t), y(t) \rangle = \left\langle u_0 t + x_0, -\frac{1}{2}gt^2 + v_0 t + y_0 \right\rangle \quad (57)$$

Two-Dimensional Motion

Assume an object traveling over horizontal ground, acted on only by the gravitational force, has an initial position $\langle x_0, y_0 \rangle = \langle 0, 0 \rangle$ and initial velocity

$\langle u_0, v_0 \rangle = \langle |\mathbf{v}_0| \cos \alpha, |\mathbf{v}_0| \sin \alpha \rangle$. The trajectory, which is a segment or a parabola, has the following properties.

$$\text{time of flight} = T = \frac{2|\mathbf{v}_0| \sin \alpha}{g} \quad (58)$$

$$\text{range} = \frac{|\mathbf{v}_0| \sin 2\alpha}{g} \quad (59)$$

$$\text{maximum height} = y\left(\frac{T}{2}\right) = \frac{(|\mathbf{v}_0| \sin \alpha)^2}{2g} \quad (60)$$

12.8 Length of Curves

Consider the parameterized curve $\mathbf{r}(t) = \langle f(t), g(t), h(t) \rangle$, where f' , g' , and h' are continuous, and the curve is traversed once for $a \leq t \leq b$. The **arc length** of the curve between $(f(a), g(a), h(a))$ and $(f(b), g(b), h(b))$ is

$$L = \int_a^b \sqrt{f'(t)^2 + g'(t)^2 + h'(t)^2} dt = \int_a^b |\mathbf{r}'(t)| dt \quad (61)$$

Arc Length of a Polar Curve

Let f have a continuous derivative on the interval $[\alpha, \beta]$. The **arc length** of the polar curve $r = f(\theta)$ on $[\alpha, \beta]$ is

$$L = \int_{\alpha}^{\beta} \sqrt{f(\theta)^2 + f'(\theta)^2} d\theta. \quad (62)$$

Arc Length as a Function of a Parameter

Let $\mathbf{r}(t)$ describe a smooth curve, for $t \geq a$. The arc length is given by

$$s(t) = \int_a^t |\mathbf{v}(u)| du, \quad (63)$$

where $|\mathbf{v}| = |\mathbf{r}'|$. Equivalently, $\frac{ds}{dt} = |\mathbf{v}t| > 0$. If $|\mathbf{v}(t)| = 1$, for all $t \geq a$, then the parameter t corresponds to arc length.

Arc Length as a Function of a Parameter

Let $\mathbf{r}(t)$ describe a smooth curve, for $t \geq a$. The arc length is given by

$$s(t) = \int_a^t |\mathbf{v}(u)| du, \quad (64)$$

where $|\mathbf{v}| = |\mathbf{r}'|$. Equivalently, $\frac{ds}{dt} = |\mathbf{v}(t)| > 0$. If $|\mathbf{v}(t)| = 1$, for all $t \geq a$, then the parameter t corresponds to arc length.

12.9 Curvature and Normal Vectors

Curvature

Let $\mathbf{r}$ describe a smooth parameterized curve. If s denotes arc length and $\mathbf{T} = \frac{\mathbf{r}'}{|\mathbf{r}'|}$ is the unit tangent vector, the **curvature** is $\kappa(s) = \left| \frac{d\mathbf{T}}{ds} \right|$

Curvature Formula

Let $\mathbf{r}(t)$ describes a smooth parameterized curve, where t is any parameter. If $\mathbf{v} = \mathbf{r}'$ is the velocity and $\mathbf{T}$ is the unit tangent vector, then the curvature is

$$\kappa(t) = \frac{1}{|\mathbf{v}|} \left| \frac{d\mathbf{T}}{dt} \right| = \left| \frac{\mathbf{T}'(t)}{\mathbf{r}'(t)} \right| \quad (65)$$

Alternative Curvature Formula

Let $\mathbf{r}$ be the position of an object moving on a smooth curve. The **curvature** at a point on the curve is

$$\kappa = \frac{|\mathbf{v} \times \mathbf{a}|}{|\mathbf{v}|^3}, \quad (66)$$

where $\mathbf{v} = \mathbf{r}'$ is the velocity and $\mathbf{a} = \mathbf{v}'$ is the acceleration.

Principal Unit Normal Vector

Let $\mathbf{r}$ describe a smooth parameterized curve. The **principal unit normal vector** at a point P on the curve at which $\kappa \neq 0$ is

$$\mathbf{N}(s) = \frac{d\mathbf{T}/ds}{|d\mathbf{T}/ds|} = \frac{1}{\kappa} \frac{d\mathbf{T}}{ds} \quad (67)$$

In practice, we use the equivalent formula

$$\mathbf{N}(t) = \frac{d\mathbf{T}/dt}{|d\mathbf{T}/dt|} \quad (68)$$

evaluated at the value of t corresponding to P .

Properties of the Principal Unit Normal Vector

Let $\mathbf{r}$ describe a smooth parameterized curve with unit tangent vector $\mathbf{T}$ and principal unit normal vector $\mathbf{N}$.

1. $\mathbf{T}$ and $\mathbf{N}$ are orthogonal at all points of the curve; that is, $\mathbf{T}(t) \cdot \mathbf{N}(t) = 0$, at all points where $\mathbf{N}$ is defined.
2. The principal unit normal vector points to the inside of the curve—in the direction that the curve is turning.

Tangential and Normal Components of the Acceleration

The acceleration vector of an object moving in space along a smooth curve has the following representation in terms of its **tangential component** a_T (in the direction of $\mathbf{T}$) and its normal component a_N (in the direction of $\mathbf{N}$):

$$\mathbf{a} = a_N \mathbf{N} + a_T \mathbf{T}, \quad (69)$$

where $a_N = \kappa |\mathbf{v}|^2 = \frac{|\mathbf{v} \times \mathbf{a}|}{|\mathbf{v}|}$ and $a_T = \frac{d^2 s}{dt^2}$.

Unit Binormal Vector and Torsion

Let C be a smooth parameterized curve with unit tangent and principal unit normal vectors $\mathbf{T}$ and $\mathbf{N}$, respectively. Then, at each point of the curve at which the curvature is nonzero, the **unit binormal vector** is

$$\mathbf{B} = \mathbf{T} \times \mathbf{N} \quad (70)$$

and the **torsion** is

$$\tau = -\frac{d\mathbf{B}}{ds} \cdot \mathbf{N} \quad (71)$$

Formulas for Curves in Space

1. Position function: $\mathbf{r}(t) = \langle x(t), y(t), z(t) \rangle$
2. Velocity: $\mathbf{v} = \mathbf{r}'$
3. Acceleration: $\mathbf{a} = \mathbf{v}'$

4. Unit tangent vector: $\mathbf{T} = \frac{\mathbf{v}}{|\mathbf{v}|}$
5. Principal unit normal vector: $\mathbf{N} = \frac{d\mathbf{T}/dt}{|d\mathbf{T}/dt|}$ (provided $d\mathbf{T}/dt \neq \mathbf{0}$)
6. Curvature: $\kappa = \frac{d\mathbf{T}}{ds} = \frac{1}{|\mathbf{v}|} \left| \frac{d\mathbf{T}}{dt} \right| = \left| \frac{\mathbf{T}'(t)}{\mathbf{r}'(t)} \right| = \frac{|\mathbf{v} \times \mathbf{a}|}{|\mathbf{v}|^3}$
7. Components of acceleration: $\mathbf{a} = a_N \mathbf{N} + a_T \mathbf{T}$, where $a_N = \kappa |\mathbf{v}|^2 = \frac{|\mathbf{v} \times \mathbf{a}|}{|\mathbf{v}|}$ and $a_T = \frac{d^2 s}{dt^2} = \frac{\mathbf{v} \cdot \mathbf{a}}{|\mathbf{v}|}$
8. Unit binormal vector: $\mathbf{B} = \mathbf{T} \times \mathbf{N} = \frac{\mathbf{v} \times \mathbf{a}}{|\mathbf{v} \times \mathbf{a}|}$
9. Torsion $\tau = -\frac{d\mathbf{B}}{ds} \cdot \mathbf{N} = \frac{(\mathbf{v} \times \mathbf{a}) \cdot \mathbf{a}'}{|\mathbf{v} \times \mathbf{a}|^2} = \frac{(\mathbf{r}' \times \mathbf{r}'') \cdot \mathbf{r}'''}{(\mathbf{r}' \times \mathbf{r}'')^2}$

Chapter 13: Functions of Several Variables

Illya Starikov

August 6, 2025

Contents

13.1 Planes and Surfaces	2
13.2 Graphs and Level Curves	5
13.3 Limits and Continuity	6
13.4 Partial Derivatives	8
13.5 The Chain Rule	9
13.6 Directional Derivatives and the Gradient	10
13.7 Tangent Planes and Linear Approximation	12
13.8 Maximum/Minimum Problems	13
13.9 Lagrange Multipliers	15

13.1 Planes and Surfaces

Plane in $\mathbb{R}^3$

Given a fixed point P_0 and a nonzero **normal vector** $\mathbf{n}$, the set of points P in $\mathbb{R}^3$ for which $\vec{P_0P}$ is orthogonal to $\mathbf{n}$ is called a **plane**.

General Equation of a Plane in $\mathbb{R}^3$

The plane passing through the point $P_0(x_0, y_0, z_0)$ with a nonzero normal vector $\mathbf{n} = \langle a, b, c \rangle$ is described by the equation

$$a(x - x_0) + b(y - y_0) + c(z - z_0) = 0 \quad \text{or} \quad ax + by + cz = d \quad (1)$$

where $d = ax_0 + by_0 + cz_0$.

Parallel and Orthogonal Planes

Two distinct planes are **parallel** if their respective normal vectors are parallel (that is, the normal vectors are scalar multiples of each other). Two planes are **orthogonal** if their respective normal vectors are orthogonal (that is, the dot product of the normal vectors is zero).

Cylinder

Given a curve C in a plane P and a line ℓ not in P , a **cylinder** is the surface consisting of all lines parallel to ℓ that pass through C .

Trace

A **trace** of a surface is the set of points at which the surface intersects a plane that is parallel to one of the coordinate planes. The trace in the coordinate planes are called **xy-trace**, the **xz-trace**, and the **yz-trace**.

Quadratic Surfaces

Name	Standard Equation	Features
Ellipsoid	$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$	All traces are ellipses.
Elliptic paraboloid	$z = \frac{x^2}{a^2} + \frac{y^2}{b^2}$	Traces with $z = z_0 > 0$ are ellipses. Traces with $x = x_0$ or $y = y_0$ are parabolas.
Hyperboloid of one sheet	$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1$	Traces with $z = z_0$ are ellipses for all z_0 . Traces with $x = x_0$ or $y = y_0$ are hyperbolas.
Hyperboloid of two sheets	$-\frac{x^2}{a^2} - \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$	Traces with $z = z_0$ with $ z_0 > c $ are ellipses. Traces with $x = x_0$ and $y = y_0$ are hyperbolas.
Elliptic cone	$-\frac{x^2}{a^2} + \frac{y^2}{b^2} = \frac{z^2}{c^2}$	Traces with $z = z_0 \neq 0$ are ellipses. Traces with $x = x_0$ or $y = y_0$ are hyperbolas or intersecting lines.
Hyperbolic paraboloid	$z = \frac{x^2}{a^2} - \frac{y^2}{b^2}$	Traces with $z = z_0 \neq 0$ are hyperbolas. Traces with $x = x_0$ or $y = y_0$ are parabolas.

13.2 Graphs and Level Curves

Function, Domain, and Range with Two Independent Variables

A **function** $z = f(x, y)$ assigns to each point (x, y) in a set D in $\mathbb{R}^2$ a unique real number z in a subset of $\mathbb{R}$. The set D is the **domain** of f . The **range** of f is the set of real numbers z that are assumed as the points (x, y) vary over the domain.

Function, Domain, and Range with n Independent Variables

The **function** $y = f(x_1, x_2, \dots, x_n)$ assigns a unique real number y to each point $(x_1, x_2, \dots, x_n)$ in a set D in $\mathbb{R}^n$. The set D is the **domain** of f . The **range** is the set of real numbers y that are assumed as the points $(x_1, x_2, \dots, x_n)$ vary over the domain.

13.3 Limits and Continuity

Limit of a Function of Two Variables

The function f has the **limit** L as $P(x, y)$ approaches $P_0(a, b)$ written

$$\lim_{(x, y) \rightarrow (a, b)} f(x, y) = L \quad (2)$$

if, given any $\varepsilon > 0$, there exists a $\delta > 0$ such that

$$|f(x, y) - L| < \varepsilon \quad (3)$$

whenever (x, y) is in the domain of f and

$$0 < |PP_0| = \sqrt{(x - a)^2 + (y - b)^2} < \delta \quad (4)$$

Limits of Constant and Linear Functions

Let a, b , and c be real numbers.

1. Constant functions $f(x, y) = c : \lim_{(x, y) \rightarrow (a, b)} c = c$
2. Linear function $f(x, y) = x : \lim_{(x, y) \rightarrow (a, b)} x = a$
3. Linear function $f(x, y) = y : \lim_{(x, y) \rightarrow (a, b)} y = b$

Limit Laws and Functions of Two Variables

Let L and M be real numbers and suppose that $\lim_{(x, y) \rightarrow (a, b)} f(x, y) = L$ and $\lim_{(x, y) \rightarrow (a, b)} g(x, y) = M$. Assume c is a constant, and $\forall m, n \in \mathbb{Z}$.

1 Sum $\lim_{(x, y) \rightarrow (a, b)} (f(x, y) + g(x, y)) = L + M$

2 Difference $\lim_{(x, y) \rightarrow (a, b)} (f(x, y) - g(x, y)) = L - M$

3 Constant multiple $\lim_{(x, y) \rightarrow (a, b)} cf(x, y) = cL$

4 Product $\lim_{(x, y) \rightarrow (a, b)} f(x, y) \cdot g(x, y) = L \cdot M$

5 Quotient $\lim_{(x, y) \rightarrow (a, b)} \left[\frac{f(x, y)}{g(x, y)} \right] = \frac{L}{M}$

6 Power $\lim_{(x,y) \rightarrow (a,b)} (f(x, y))^n = L^n$

7 m/n Power If m and n have no common factors and $n \neq 0$, then $\lim_{(x,y) \rightarrow (a,b)} [f(x, y)]^{m/n} = L^{m/n}$, where we assume $L > 0$ if n is even.

Interior and Boundary Points

Let R be a region in $\mathbb{R}^2$. An **interior point** P of R lies entirely within R , which means it is possible to find a disk centered at P that contains only points of R .

A **boundary point** Q of R lies on the edge of R in the sense that *every* disk centered at Q contains at least one point in R and at least one point not in R .

Open and Closed Sets

A region is **open** if it consists entirely of interior points. A region is **closed** if it contains all its boundary points.

Two-Path Test for Nonexistence of Limits

If $f(x, y)$ approaches two different values as (x, y) approaches (a, b) along two different paths in the domain of f , then $\lim_{(x,y) \rightarrow (a,b)} f(x, y)$ does not exist.

Continuity

A function f is continuous at the point (a, b) provided

1. f is defined at (a, b) .
2. $\lim_{(x,y) \rightarrow (a,b)} f(x, y)$ exists.
3. $\lim_{(x,y) \rightarrow (a,b)} f(x, y) = f(a, b)$.

Continuity of Composite Functions

If $u = g(x, y)$ is continuous at (a, b) and $z = f(u)$ is continuous at $g(a, b)$, then the composite function $z = f(g(x, y))$ is continuous at (a, b) .

13.4 Partial Derivatives

The **partial derivative of f with respect to x at the point (a, b)** is

$$f_x(a, b) = \lim_{h \rightarrow 0} \frac{f(a + h, b) - f(a, b)}{h}. \quad (5)$$

The **partial derivative of f with respect to y at the point (a, b)** is

$$f_y(a, b) = \lim_{h \rightarrow 0} \frac{f(a, b + h) - f(a, b)}{h}. \quad (6)$$

provided these limits exists.

Equality of Mixed Partial Derivatives

Assume that f is defined on an open set D of $\mathbb{R}^2$, and f_{xy} and f_{yx} are continuous throughout D . Then $f_{xy} = f_{yx}$ at all points of D .

Differentiability

The function $z = f(x, y)$ is **differentiable at (a, b)** provided $f_x(a, b)$ and $f_y(a, b)$ exist and the change $\Delta z = f(a + \Delta x, b + \Delta y) - f(a, b)$ equals

$$\Delta z = f_x(a, b)\Delta x + f_y(a, b)\Delta y + \varepsilon_1\Delta x + \varepsilon_2\Delta y, \quad (7)$$

where for fixed a and b , ε_1 and ε_2 are functions that depend only on Δx and Δy , with $(\varepsilon_1, \varepsilon_2) \rightarrow (0, 0)$ as $(\Delta x, \Delta y) \rightarrow (0, 0)$. A function is **differentiable** on an open set R if it is differentiable at every point on R .

Conditions for Differentiability

Suppose the function f has partial derivatives f_x and f_y defined on an open set containing (a, b) , with f_x and f_y continuous (a, b) . Then f is differentiable at (a, b) .

Differentiability Implies Continuity

If a function f is differentiable at (a, b) , then it is continuous at (a, b)

13.5 The Chain Rule

Chain Rule (One Independent Variable)

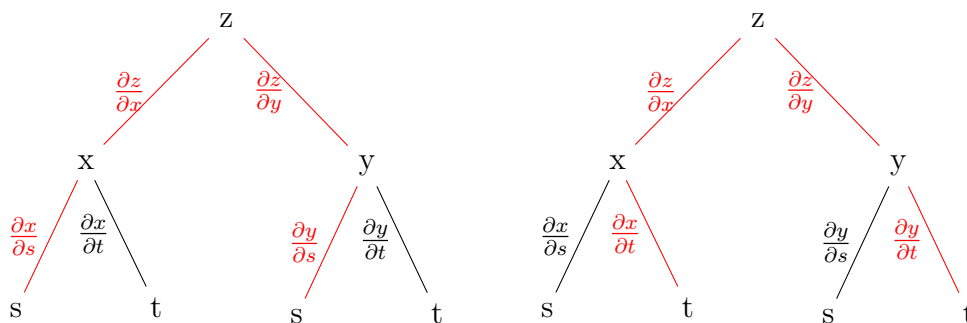
Let z be a differentiable function of x and y on its domain, where x and y are differentiable functions of t on an interval I . Then

$$\frac{dz}{dt} = \frac{\partial z}{\partial x} \frac{dx}{dt} + \frac{\partial z}{\partial y} \frac{dy}{dt}. \quad (8)$$

Chain Rule (Two Independent Variables)

Let z be a differentiable function of x and y , where x and y are differentiable functions of s and t . Then

$$\frac{dz}{ds} = \frac{\partial z}{\partial x} \frac{\partial x}{\partial s} + \frac{\partial z}{\partial y} \frac{\partial y}{\partial s} \quad \text{and} \quad \frac{dz}{dt} = \frac{\partial z}{\partial x} \frac{\partial x}{\partial t} + \frac{\partial z}{\partial y} \frac{\partial y}{\partial t} \quad (9)$$



Implicit Differentiation

Let F be differentiable on its domain and suppose that $F(x, y) = 0$ defines y as a differentiable function of x . Provided $F_y \neq 0$.

$$\frac{dy}{dx} = -\frac{F_x}{F_y} \quad (10)$$

13.6 Directional Derivatives and the Gradient

Directional Derivative

Let f be differentiable at (a, b) and let $\mathbf{u} = \langle \cos \theta, \sin \theta \rangle$ be a unit vector in the xy -plane. The **directional derivatives of f at (a, b) in the direction of $\mathbf{u}$** is

$$D_{\mathbf{u}}f(a, b) = \lim_{h \rightarrow 0} \frac{f(a + h \cos \theta, b + h \sin \theta) - f(a, b)}{h} \quad (11)$$

provided the limit exists.

Directional Derivative

Let f be differentiable on (a, b) and let $\mathbf{u} = \langle u_1, u_2 \rangle$ be a unit vector in the x, y -plane. The **directional derivative of f at (a, b) in the direction of $\mathbf{u}$** is

$$D_{\mathbf{u}}f(a, b) = \langle f_x(a, b), f_y(a, b) \rangle \cdot \langle u_1, u_2 \rangle \quad (12)$$

Gradient (Two Dimensions)

Let f be differentiable at the point (x, y) . The **gradient of f at (x, y)** is the vector-valued function

$$\nabla f(x, y) = \langle f_x(x, y), f_y(x, y) \rangle = f_x(x, y)\mathbf{i} + f_y(x, y)\mathbf{j} \quad (13)$$

Directions of Change

Let f be differentiable at (a, b) with $\nabla f(a, b) \neq \mathbf{0}$

1. f has its maximum rate of increase at (a, b) in the direction of the gradient $\nabla f(a, b)$. The rate of increase in this direction is $|\nabla f(a, b)|$
2. f has its maximum rate of decrease at (a, b) in the direction of $-\nabla f(a, b)$. The rate of decrease in this direction is $-|\nabla f(a, b)|$.
3. The directional derivative is zero in any direction orthogonal to $\nabla f(a, b)$.

The Gradient and Level Curves

Given a function f differentiable at (a, b) , the line tangent to the level curve of f at (a, b) is orthogonal to the gradient $\nabla f(a, b)$, provided $\nabla f(a, b) \neq \mathbf{0}$.

Gradient and Directional Derivative in Three Dimensions

Let f be differentiable at the point (x, y, z) . The **gradient** of f at (x, y, z) is the vector-valued function

$$\nabla f(x, y, z) = \langle f_x(x, y, z), f_y(x, y, z), f_z(x, y, z) \rangle \quad (14)$$

$$= f_x(x, y, z)\hat{\mathbf{i}} + f_y(x, y, z)\hat{\mathbf{j}} + f_z(x, y, z)\hat{\mathbf{k}} \quad (15)$$

The **directional derivative** of f in the direction of the unit vector $\mathbf{u} = \langle u_1, u_2, u_3 \rangle$ at the point (a, b, c) is $D_{\mathbf{u}}f(a, b, c) = \nabla f(a, b, c) \cdot \mathbf{u}$

13.7 Tangent Planes and Linear Approximation

Let F be differentiable at the point $P_0(a, b, c)$ with $\nabla F(a, b, c) \neq \mathbf{0}$. The plane tangent to the surface $F(x, y, z) = 0$ at P_0 , called the **tangent plane**, is the plane passing through P_0 orthogonal $\nabla F(a, b, c)$. An equation of the tangent plane is

$$F_x(a, b, c)(x - a) + F_y(a, b, c)(y - b) + F_z(a, b, c)(z - c) = 0. \quad (16)$$

Tangent Plane for $z = f(x, y)$

Let f be differentiable at the point (a, b) . An equation of the plane tangent to the surface $z = f(x, y)$ at the point $(a, b, f(a, b))$ is

$$z = f_x(a, b)(x - a) + f_y(a, b)(y - b) + f(a, b) \quad (17)$$

Linear Approximation

Let f be differentiable at (a, b) . The linear approximation to the surface $z = f(x, y)$ at the point $(a, b, f(a, b))$ is the tangent plane at that point, given by the equation

$$L(x, y) = f_x(a, b)(x - a) + f_y(a, b)(y - b) + f(a, b) \quad (18)$$

The Differential dz

Let f be differentiable at the point (a, b) . The change in $z = f(x, y)$ as the independent variables change from (a, b) to $(a + dx, b + dy)$ is denoted by the differential dz :

$$\Delta z \approx dz = f_x(a, b)dx + f_y(a, b)dy \quad (19)$$

13.8 Maximum/Minimum Problems

Local Maximum/Minimum Values

A function f has a **local maximum value** at (a, b) if $f(x, y) \leq f(a, b)$ for all (x, y) in the domain of f in some open disk centered at (a, b) . A function f has a **local minimum value** at (a, b) if $f(x, y) \geq f(a, b)$ for all (x, y) in the domain of f in some open disk centered at (a, b) . Local maximum and local minimum values are also called **local extreme values** or **local extrema**.

Derivatives and Local Maximum/Minimum Values

If f has a local maximum or minimum value at (a, b) and the partial derivatives f_x and f_y exist at (a, b) then $f_x(a, b) = f_y(a, b) = 0$.

Critical Point

An interior point (a, b) in the domain of f is a **critical point** of f is either

1. $f_x(a, b) = f_y(a, b) = 0$, or
2. one (or both) of f_x or f_y does not exist at (a, b)

Saddle Point

A function f has a **saddle point** at a critical point (a, b) if, in every open disk centered at (a, b) , there are points (x, y) for which $f(x, y) > f(a, b)$ and points for which $f(x, y) < f(a, b)$

Second Derivative Test

Suppose that the second partial derivative of f are continuous throughout an open disk centered at the point (a, b) , where $f_x(a, b) = f_y(a, b) = 0$. Let $D(x, y) = f_{xx}(x, y)f_{yy}(x, y) - (f_{xy}(x, y))^2$.

1. If $D(a, b) > 0$ and $f_{xx}(a, b) < 0$, then f has a local maximum value at (a, b) .
2. If $D(a, b) > 0$ and $f_{xx}(a, b) > 0$, then f has a local minimum value at (a, b) .

3. If $D(a, b) < 0$, then f has a saddle point at (a, b) .
4. If $D(a, b) = 0$, then the test is inconclusive.

Absolute Maximum/Minimum Values

If $f(x, y) \leq f(a, b)$ for all (x, y) in the domain of f , then f has an **absolute maximum value** at (a, b) . If $f(x, y) \geq f(a, b)$ for all (x, y) in the domain of f , then f has an **absolute minimum value** at (a, b) .

Finding Absolute Maximum/Minimum Values on Closed, Bounded Sets

Let f be continuous on a closed bounded set R in $\mathbb{R}^2$. To find the absolute maximum and minimum values of f on R :

1. Determine the values of f at all critical points in R .
2. Find the maximum and minimum values of f on the boundary of R .
3. The greatest function values found in Step 1 and 2 is the absolute maximum value of f on R , and the least function value found in Steps 1 and 2 is the absolute minimum value of f on R .

13.9 Lagrange Multipliers

Parallel Gradients (Ball Park Theorem)

Let f be a differentiable function in a region of $\mathbb{R}^2$ that contains the smooth curve C given by $g(x, y) = 0$. Assume that f has a local extreme value (relative to values of f on C) at a point $P(a, b)$ on C . Then $\nabla f(a, b)$ is orthogonal to the line tangent to C at P . Assuming $\nabla g(a, b) \neq 0$, it follows that there is a real number λ (called a **Lagrange multiplier**) such that $\nabla f(a, b) = \lambda \nabla g(a, b)$.

Method of Lagrange Multipliers in Two Variables

Let the objective function f and the constraint function g be differentiable on a region of $\mathbb{R}^2$ with $\nabla g(x, y) \neq 0$ on the curve $g(x, y) = 0$. To locate the maximum and minimum values of f subject to the constraint $g(x, y) = 0$, carry out the following steps.

1. Find the values of x, y and λ (if they exist) that satisfy the equations

$$\nabla f(x, y) = \lambda \nabla g(x, y) \quad \text{and} \quad g(x, y) = 0 \quad (20)$$

2. Among the values (x, y) found in Step 1, select the largest and smallest corresponding function values, which are the maximum and minimum values of f subject to the constraint.

Method of Lagrange Multipliers in Three Variables

Let f and g be differentiable on a region of $\mathbb{R}^3$ with $\nabla g(x, y, z) \neq 0$ on the surface $g(x, y, z) = 0$. To locate the maximum and minimum values of f subject to the constraint $g(x, y, z) = 0$, carry out the following steps.

1. Find the values of x, y, z and λ that satisfy the equations

$$\nabla f(x, y, z) = \lambda \nabla g(x, y, z) \quad \text{and} \quad g(x, y, z) = 0 \quad (21)$$

2. Among the points (x, y, z) found in Step 1, select the largest and smallest corresponding values of the objective function. These values are the maximum and minimum values of f subject to the constraint.

Chapter 14: Multiple Integration

Illya Starikov

August 6, 2025

Contents

14.1 Double Integrals over Rectangular Regions	2
14.2 Double Integrals over General Regions	3
14.3 Double Integrals in Polar Coordinates	4
14.4 Triple Integrals	5
14.5 Triple Integrals in Cylindrical and Spherical Coordinates . . .	6
14.6 Integrals for Mass Calculations	8

14.1 Double Integrals over Rectangular Regions

Volumes and Double Integrals

A function f defined on a rectangular region R in the xy -plane is **integratable** on R if $\lim_{\Delta \rightarrow 0} \sum_{k=1}^n f(x_k^*, y_k^*) \Delta A_k$ exists for all partitions of R and for all choices of (x_k^*, y_k^*) within those partitions. The limit is the **double integral of f over R** , which we write

$$\iint_R f(x, y) dA = \lim_{\Delta \rightarrow 0} \sum_{k=1}^n f(x_k^*, y_k^*) \Delta A_k \quad (1)$$

If f is nonnegative on R , then the double integral equals the volume of the solid bounded by $z = f(x, y)$ and the xy -plane over R .

Double Integrals on Rectangular Regions

Let f be continuous on the rectangular region $R = \{(x, y) : a \leq x \leq b, c \leq y \leq d\}$. The double integral of f over R may be evaluated by either of two iterated integrals:

$$\iint_R f(x, y) dA = \int_c^d \int_a^b f(x, y) dx dy = \int_a^b \int_c^d f(x, y) dy dx \quad (2)$$

Average Value of a Function over a Plane Region

The **average value** of an integrable function f over a region R is

$$\bar{f} = \frac{1}{\text{area of } R} \iint_R f(x, y) dA \quad (3)$$

14.2 Double Integrals over General Regions

Let R be a region bounded below and above by the graphs of the continuous functions $y = g(x)$ and $y = h(x)$, respectively, and by the lines $x = a$ and $x = b$. If f is continuous on R , then

$$\iint_R f(x, y) dA = \int_a^b \int_{g(x)}^{h(x)} f(x, y) dy dx \quad (4)$$

Let R be a region bounded on the left and right by the graphs of the continuous functions $x = g(y)$ and $x = h(y)$, respectively, and the lines $y = c$ and $y = d$. If f is continuous on R , then

$$\iint_R f(x, y) dA = \int_c^d \int_{g(y)}^{h(y)} f(x, y) dx dy \quad (5)$$

Areas of Regions by Double Integrals

Let R be a region in the xy -plane. Then

$$\text{area of } R = \iint_R 1 dA \quad (6)$$

14.3 Double Integrals in Polar Coordinates

Double Integrals over Polar Rectangular Region

Let f be continuous on the region in the xy -plane $R = \{(r, \theta) : 0 \leq a \leq r \leq b, \alpha \leq \theta \leq \beta\}$, where $\beta - \alpha \leq 2\pi$. Then

$$\iint_R f(r, \theta) dA = \int_{\alpha}^{\beta} \int_a^b f(r, \theta) r dr d\theta \quad (7)$$

Double Integrals over More General Polar Regions

Let f be continuous on the region in the xy -plane

$$R = \{(r, \theta) : 0 \leq g(\theta) \leq r \leq h(\theta), \alpha \leq \theta \leq \beta\} \quad (8)$$

where $0 < \beta - \alpha \leq 2\pi$. Then

$$\iint_R f(r, \theta) dA = \int_{\alpha}^{\beta} \int_{g(\theta)}^{h(\theta)} f(r, \theta) r dr d\theta. \quad (9)$$

Area of Polar Regions

The area of the region $R = \{(r, \theta) : 0 \leq g(\theta) \leq r \leq h(\theta), \alpha \leq \theta \leq \beta\}$, where $0 < \beta - \alpha \leq 2\pi$, is

$$A = \iint_R dA = \int_{\alpha}^{\beta} \int_{g(\theta)}^{h(\theta)} r dr d\theta \quad (10)$$

14.4 Triple Integrals

Let f be continuous over the region

$$D = \{(x, y, z) : a \leq x \leq b, g(x) \leq y \leq h(x), G(x, y) \leq z \leq H(x, y)\} \quad (11)$$

where g, h, G , and H are continuous functions. Then f is integrable over D and the triple integral is evaluated as the iterated integral

$$\iiint_D f(x, y, z) dV = \int_a^b \int_{g(x)}^{h(x)} \int_{G(x, y)}^{H(x, y)} f(x, y, z) dz dy dx. \quad (12)$$

Average Value of a Function of Three Variables

If f is continuous on a region D of $\mathbb{R}^3$, then the average value of f over D is

$$\bar{f} = \frac{1}{\text{volume}(D)} \iiint_D f(x, y, z) dV \quad (13)$$

14.5 Triple Integrals in Cylindrical and Spherical Coordinates

Transformations Between Cylindrical and Rectangular Coordinates

Rectangular $\rightarrow$ Cylindrical Cylindrical $\rightarrow$ Rectangular

$$r^2 = x^2 + y^2$$

$$\tan \theta = y/x$$

$$z = z$$

$$x = r \cos \theta$$

$$y = r \sin \theta$$

$$z = z$$

Triple Integrals in Cylindrical Coordinates

Let f be continuous over the region

$$D = \{(r, \theta, z) : g(\theta) \leq r \leq h(\theta), \alpha \leq \theta \leq \beta, G(x, y) \leq z \leq H(x, y)\} \quad (14)$$

Then f is integrable over D and the triple integral of f over D in cylindrical coordinates is

$$\iiint_D f(r, \theta, z) dV = \int_{\alpha}^{\beta} \int_{g(\theta)}^{h(\theta)} \int_{G(r \cos \theta, r \sin \theta)}^{H(r \cos \theta, r \sin \theta)} f(r, \theta, z) dz r dr d\theta \quad (15)$$

Transformations Between Spherical and Rectangular Coordinates

Rectangular $\rightarrow$ Spherical Spherical $\rightarrow$ Rectangular

$$\rho^2 = x^2 + y^2 + z^2$$

Use trigonometry to find φ and θ

$$x = \rho \sin \varphi \cos \theta$$

$$y = \rho \sin \varphi \sin \theta$$

$$z = \rho \cos \varphi$$

Triple Integrals in Spherical Coordinates

Let f be continuous over the region

$$D = \{(\rho, \varphi, \theta) : g(\varphi, \theta) \leq \rho \leq h(\varphi, \theta), a \leq \varphi \leq b, \alpha \leq \theta \leq \beta\} \quad (16)$$

Then f is integrable over D , and the triple integral of f over D in spherical coordinates is

$$\iiint_D f(\rho, \varphi, \theta) dV = \int_{\alpha}^{\beta} \int_a^b \int_{g(\varphi, \theta)}^{h(\varphi, \theta)} f(\rho, \varphi, \theta) \rho^2 \sin \varphi \, d\rho \, d\varphi \, d\theta \quad (17)$$

14.6 Integrals for Mass Calculations

Center of Mass in One Dimension

Let ρ be an integrable density function on the interval $[a, b]$ (which represents a thin rod or wire). The **center of mass** is location at the point $\bar{x} = \frac{M}{m}$, where the **total moment** M and mass m are

$$M = \int_a^b x\rho(x) dx \quad \text{and} \quad m = \int_a^b \rho(x) dx \quad (18)$$

Center of Mass in Two Dimensions

Let ρ be integrable density function defined over a closed bounded region R in $\mathbb{R}^2$. The coordinates of the center of mass of the object represented by R are

$$\bar{x} = \frac{M_y}{m} = \frac{1}{m} \iint_R x\rho(x, y) dA \quad \text{and} \quad \bar{y} = \frac{M_x}{m} = \frac{1}{m} \iint_R y\rho(x, y) dA \quad (19)$$

Center of Mass in Three Dimensions

Let ρ be integrable density function on a closed bounded region D in $\mathbb{R}^3$. The coordinates of the center of mass of the region are

$$\bar{x} = \frac{M_{yz}}{m} = \frac{1}{m} \iiint_R x\rho(x, y, z) dV \quad (20)$$

$$\bar{y} = \frac{M_{xz}}{m} = \frac{1}{m} \iiint_R y\rho(x, y, z) dV \quad (21)$$

$$\bar{z} = \frac{M_{xy}}{m} = \frac{1}{m} \iiint_R z\rho(x, y, z) dV \quad (22)$$

where $m = \iiint_D \rho(x, y, z) dV$ is the mass, and M_{yz} , M_{xz} and M_{xy} are the moments with respect to the coordinate planes.

Chapter 15: Vector Calculus

Illya Starikov

August 6, 2025

Contents

15.1	Vector Fields	2
15.2	Line Integrals	4
15.3	Conservative Vector Fields	7
15.4	Green’s Theorem	9
15.5	Divergence and Curl	10
15.6	Surface Integrals	12
15.7	Stokes’ Theorem	14
15.8	Divergence Theorem	15

15.1 Vector Fields

Vector Fields in Two Dimensions

Let f and g be defined on a region R of $\mathbb{R}^2$. A **vector field** in $\mathbb{R}^2$ is a function $\mathbf{F}$ that assigns to each point in R a vector $\langle f(x, y), g(x, y) \rangle$. The vector field is written as

$$\mathbf{F}(x, y) = \langle f(x, y), g(x, y) \rangle \quad \text{or} \quad \mathbf{F}(x, y) = f(x, y)\hat{\mathbf{i}} + g(x, y)\hat{\mathbf{j}} \quad (1)$$

A vector field $\mathbf{F} = \langle f, g \rangle$ is continuous or differentiable on a region R of $\mathbb{R}^2$ if f and g are continuous or differentiable on R , respectively.

Radial Vector Fields in $\mathbb{R}^2$

Let $\mathbf{r} = \langle x, y \rangle$. A vector field of the form $\mathbf{F} = f(x, y)\mathbf{r}$, where f is a scalar-valued function, is a **radial vector field**. Of specific interest are the radial vector field

$$\mathbf{F}(x, y) = \frac{\mathbf{r}}{|\mathbf{r}|^p} = \frac{\langle x, y \rangle}{|\mathbf{r}|^p} \quad (2)$$

where p is a real number. At every point (except the origin), the vectors of this field are directed outward from the origin with the magnitude of $|\mathbf{F}| = \frac{1}{|\mathbf{r}|^{p-1}}$.

Vector Fields in Three Dimensions

Let f , g , and h be defined on a region D of $\mathbb{R}^3$. A **vector field** in $\mathbb{R}^3$ is a function $\mathbf{F}$ that assigns to each point in D a vector $\langle f(x, y, z), g(x, y, z), h(x, y, z) \rangle$. The vector field is written as

$$\mathbf{F}(x, y, z) = \langle f(x, y, z), g(x, y, z), h(x, y, z) \rangle \quad \text{or} \quad (3)$$

$$\mathbf{F}(x, y, z) = f(x, y, z)\hat{\mathbf{i}} + g(x, y, z)\hat{\mathbf{j}} + h(x, y, z)\hat{\mathbf{k}} \quad (4)$$

A vector field $\mathbf{F} = \langle f, g, h \rangle$ is continuous or differentiable on a region D of $\mathbb{R}^3$ if f , g , h are continuous or differentiable on R , respectively. Of particular importance are the **radial vector fields**

$$\mathbf{F}(x, y, z) = \frac{\mathbf{r}}{|\mathbf{r}|^p} = \frac{\langle x, y, z \rangle}{|\mathbf{r}|^p} \quad (5)$$

where p is a real number.

Gradient Fields and Potential Functions

Let $z = \varphi(x, y)$ and $w = \varphi(x, y, z)$ be differentiable functions on regions of $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively. The vector field $\mathbf{F} = \nabla\varphi$ is **gradient field**, and the function φ is a **potential function** for $\mathbf{F}$.

15.2 Line Integrals

Scalar Line Integral in the Plane, Arc Length Parameter

Suppose the scalar-valued function f is defined on the smooth curve $C : \mathbf{r}(s) = \langle x(s), y(s) \rangle$, parameterized by the arc length s . The **line integral of f over C** is

$$\int_C f(x(s), y(s)) ds = \lim_{\Delta \rightarrow 0} \sum_{k=1}^n f(x(s_k^*), y(s_k^*)) \Delta s_k, \quad (6)$$

provided this limit exists over all partitions of C . When the limit exists, f is said to be **integrable** on C .

Evaluating Scalar Line Integrals in $\mathbb{R}^2$

Let f be continuous on a region containing a smooth curve $C : \mathbf{r}(t) = \langle x(t), y(t) \rangle$, for $a \leq t \leq b$. Then

$$\int_C f ds = \int_a^b f(x(t), y(t)) |\mathbf{r}'(t)| dt \quad (7)$$

$$= \int_a^b f(x(t), y(t)) \sqrt{x'(t)^2 + y'(t)^2} dt \quad (8)$$

Evaluating the Line Integral $\int_C f ds$

1. Find a parametric description of C in the form $\mathbf{r}(t) = \langle x(t), y(t) \rangle$, for $a \leq t \leq b$
2. Computer $|\mathbf{r}'(t)| = \sqrt{x'(t)^2 + y'(t)^2}$
3. Make substitutions for x and y in the integrand and evaluate an ordinary integral

$$\int_C f ds = \int_a^b f(x(t), y(t)) |\mathbf{r}'(t)| dt \quad (9)$$

Evaluating Scalar Line Integrals in $\mathbb{R}^3$

Let f be continuous on a region containing a smooth curve $C : \mathbf{r}(t) = \langle x(t), y(t), z(t) \rangle$, for $a \leq t \leq b$. Then

$$\int f \, ds = \int_a^b f(x(t), y(t), z(t)) |\mathbf{r}'(t)| \, dt \quad (10)$$

$$= \int_a^b f(x(t), y(t), z(t)) \sqrt{x'(t)^2 + y'(t)^2 + z'(t)^2} \, dt \quad (11)$$

Line Integral of a Vector Field

Let $\mathbf{F}$ be a vector field that is continuous on a region containing a smooth oriented curve C parameterized by arc length. Let $\mathbf{T}$ be the unit tangent vector at each point of C consistent with the orientation. The line integral of $\mathbf{F}$ over C is $\int_C \mathbf{F} \cdot \mathbf{T} \, ds$.

Different Forms of Line Integrals of Vector Fields

The line integral $\int_C \mathbf{F} \cdot \mathbf{T} \, ds$ may be expressed in the following forms, where $\mathbf{F} = \langle f, g, h \rangle$, for $a \leq t \leq b$:

$$\int_a^b \mathbf{F} \cdot \mathbf{r}'(t) \, dt = \int_a^b (f x'(t), g y'(t), h z'(t)) \, dt \quad (12)$$

$$= \int_C f \, dx + g \, dy + h \, dz \quad (13)$$

$$= \int_C \mathbf{F} \cdot d\mathbf{r} \quad (14)$$

For line integrals in the plane, we let $\mathbf{F} = \langle f, g \rangle$ and assume C is parameterized in the form $\mathbf{r}(t) = \langle x(t), y(t) \rangle$, for $a \leq t \leq b$. Then

$$\int_C \mathbf{F} \cdot \mathbf{T} \, ds = \int_a^b (f x'(t) + g y'(t)) \, dt = \int_C f \, dx + g \, dy = \int_C \mathbf{F} \cdot d\mathbf{r} \quad (15)$$

Work Done in a Force Field

Let $\mathbf{F}$ be a continuous force field in a region D of $\mathbb{R}^3$ and let $C : \mathbf{r}(t) = \langle x(t), y(t), z(t) \rangle$, for $a \leq t \leq b$, be a smooth curve in D with a unit tangent vector $\mathbf{T}$ consistent with the orientation. The work done in moving an object C in the positive direction is

$$W = \int_C \mathbf{F} \cdot \mathbf{T} \, ds = \int_a^b \mathbf{F} \cdot \mathbf{r}'(t) \, dt \quad (16)$$

Circulation

Let $\mathbf{F}$ be a continuous vector field on a region D of $\mathbb{R}^3$ and let C be a closed smooth oriented curve in D . The **circulation** of $\mathbf{F}$ on C is $\int_C \mathbf{F} \cdot \mathbf{T} \, ds$, where $\mathbf{T}$ is the unit vector tangent to C consistent with the orientation.

Flux

Let $F = \langle f, g \rangle$ be continuous vector field on a region R of $\mathbb{R}^2$. Let $C : \mathbf{r}(t) = \langle x(t), y(t) \rangle$, for $a \leq t \leq b$, be a smooth oriented curve in R that does not intersect itself. The **flux** of the vector field across C is

$$\int_C \mathbf{F} \cdot \mathbf{n} \, ds = \int_a^b (f y'(t) - g x'(t)) \, dt, \quad (17)$$

where $\mathbf{n} = \mathbf{T} \times \hat{\mathbf{k}}$ is the unit normal vector and $\mathbf{T}$ is the unit tangent vector consistent with the orientation. If C is a closed curve with counterclockwise orientation, $\mathbf{n}$ is the outward normal vector and the flux integral gives the **outward flux** across C .

15.3 Conservative Vector Fields

Simple and Closed Curves

Suppose a curve C (in $\mathbb{R}^2$ and $\mathbb{R}^3$) is described parametrically by $\mathbf{r}(t)$, where $a \leq t \leq b$. Then C is a **simple curve** if $\mathbf{r}(t_1) \neq \mathbf{r}(t_2)$ for all t_1 and t_2 , with $a < t_1 < t_2 < b$; that is, C never intersects itself between its endpoints. The curve C is **closed** if $\mathbf{r}(a) = \mathbf{r}(b)$; that is, the initial and terminal points of C are the same.

Connected and Simply Connected Regions

An open region R in $\mathbb{R}^2$ (or D in $\mathbb{R}^3$) is **connected** if it is possible to connect any two points of R by a continuous curve lying in R . An open region R is **simply connected** if every closed simple curve in R can be deformed and contracted to a point in R .

Conservative Vector Field

A vector field F is said to be **conservative** on a region (in $\mathbb{R}^2$ or $\mathbb{R}^3$) if there exists a scalar function φ such that $\mathbf{F} = \nabla\varphi$ on that region.

Test for Conservative Vector Fields

Let $\mathbf{F} = \langle f, g, h \rangle$ be a vector field defined on a connected and simply connected region D of $\mathbb{R}^3$, where f , g , and h have continuous first partial derivatives on D . Then $\mathbf{F}$ is a conservative vector field on D (there is a potential function φ such that $\mathbf{F} = \nabla\varphi$) if and only if

$$\frac{\partial f}{\partial y} = \frac{\partial g}{\partial x}, \quad \frac{\partial f}{\partial z} = \frac{\partial h}{\partial x}, \quad \text{and} \quad \frac{\partial g}{\partial z} = \frac{\partial h}{\partial y} \quad (18)$$

For vector fields in $\mathbb{R}^2$, we have the single condition $\frac{\partial f}{\partial y} = \frac{\partial g}{\partial x}$.

Finding Potential Functions in $\mathbb{R}^3$

Suppose $\mathbf{F} = \langle f, g, h \rangle$ is a conservative vector field. To find φ such that $\mathbf{F} = \nabla\varphi$, take the following steps:

1. Integral $\varphi_x = f$ with respect to x to obtain φ , which includes an arbitrary function $c(y, z)$.

2. Compute φ_y and equate it to g to obtain an expression for $c_y(y, z)$.
3. Integrate $c_y(y, z)$ with respect to y to obtain $c(y, z)$, including an arbitrary function $d(z)$.
4. Compute φ_z and equate it to h to get $d(z)$.

Beginning the procedure with $\varphi_y = g$ or $\varphi_z = h$ maybe be easier in some cases.

Fundamental Theorem for Line Integrals

Let $\mathbf{F}$ be a continuous vector field on an open connected region R in $\mathbb{R}^2$ (or D in $\mathbb{R}^3$). There exists a potential function φ with $\mathbf{F} = \nabla\varphi$ (which means that $\mathbf{F}$ is conservative) if and only if

$$\int_C \mathbf{F} \cdot \mathbf{T} \, ds = \int_C \mathbf{F} \cdot d\mathbf{r} = \varphi(B) - \varphi(A) \quad (19)$$

for all points A and B in R and all smooth oriented curves C from A to B .

Line Integrals on Closed Curves

Let R in $\mathbb{R}^2$ (or D in $\mathbb{R}^3$) be an open region. Then $\mathbf{F}$ is a conservative vector field on R if and only if $\oint_C \mathbf{F} \cdot d\mathbf{r} = 0$ on all simple closed smooth oriented curves C in R .

15.4 Green's Theorem

Let C be a simple closed smooth curve, oriented counterclockwise, that encloses a connected and simply connected region R in the plane. Assume $\mathbf{F} = \langle f, g \rangle$, where f and g have continuous first partial derivatives in R . Then

$$\oint_C \mathbf{F} \cdot d\mathbf{r} = \oint_C f dx + g dy = \iint_R \left(\frac{\partial g}{\partial x} - \frac{\partial f}{\partial y} \right) dA. \quad (20)$$

Two-Dimensional Curl

The **two-dimensional curl** of the vector field $\mathbf{F} = \langle f, g \rangle$ is $\frac{\partial g}{\partial x} - \frac{\partial f}{\partial y}$. If the curl is zero throughout a region, the vector field is said to be **irrotational** on that region.

Area of a Plane Region by Line Integrals

Under the conditions of Green's Theorem, the area of a region R enclosed by a curve C is

$$\oint_C x dy = - \oint_C y dx = \frac{1}{2} \oint_C (x dy - y dx) \quad (21)$$

Green's Theorem, Flux Form

Let C be a simple closed smooth curve, oriented counterclockwise, that encloses a connected and simply connected region R in the plane. Assume $\mathbf{F} = \langle f, g \rangle$, where f and g have continuous first partial derivatives in R . Then

$$\oint_C \mathbf{F} \cdot \mathbf{n} ds = \oint_C f dy - g dx = \iint_R \left(\frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} \right) dA \quad (22)$$

where $\mathbf{n}$ is the outward unit normal vector on the curve.

Two-Dimensional Divergence

The **two-dimensional divergence** of the vector field $\mathbf{F} = \langle f, g \rangle$ is $\frac{\partial f}{\partial x} + \frac{\partial g}{\partial y}$. If the divergence is zero throughout a region, the vector field is said to be **source free** on that region.

15.5 Divergence and Curl

Divergence of a Vector Field

The **divergence** of a vector field $\mathbf{F} = \langle f, g, h \rangle$ that is differentiable on a region of $\mathbb{R}^3$ is

$$\operatorname{div} \mathbf{F} = \nabla \cdot \mathbf{F} = \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} + \frac{\partial h}{\partial z} \quad (23)$$

If $\nabla \cdot \mathbf{F} = 0$, the vector field is **source free**.

Divergence of Radial Vector Fields

For a real number p , the divergence of the radial vector field

$$\mathbf{F} = \frac{\mathbf{r}}{|\mathbf{r}|^p} = \frac{\langle x, y, z \rangle}{(x^2 + y^2 + z^2)^{\frac{p}{2}}} \text{ is } \nabla \cdot \mathbf{F} = \frac{3-p}{|\mathbf{r}|^p} \quad (24)$$

Curl of a Vector Field

The **curl** of a vector field $\mathbf{F} = \langle f, g, h \rangle$ that is differentiable on a region of $\mathbb{R}^3$ is

$$\nabla \times \mathbf{F} = \operatorname{curl} \mathbf{F} = \left(\frac{\partial h}{\partial y} - \frac{\partial g}{\partial z} \right) \hat{\mathbf{i}} + \left(\frac{\partial f}{\partial z} - \frac{\partial h}{\partial x} \right) \hat{\mathbf{j}} + \left(\frac{\partial g}{\partial x} - \frac{\partial f}{\partial y} \right) \hat{\mathbf{k}} \quad (25)$$

If $\nabla \times \mathbf{F} = \mathbf{0}$, the vector field is **irrotational**.

Curl of a Conservative Vector Field

The **general rotation vector field** is $\mathbf{F} = \mathbf{a} \times \mathbf{r}$, where the nonzero constant vector $\mathbf{a} = \langle a_1, a_2, a_3 \rangle$ is the axis of rotation and $\mathbf{r} = \langle x, y, z \rangle$. For all nonzero choices of $\mathbf{a}$, $|\nabla \times \mathbf{F}| = 2|\mathbf{a}|$ and $\nabla \cdot \mathbf{F} = 0$. The constant angular speed of the vector field is

$$\omega = |\mathbf{a}| = \frac{1}{2} |\nabla \times \mathbf{F}| \quad (26)$$

Curl of a Conservative Vector Field

Suppose that $\mathbf{F}$ is a conservative vector field on an open region D of $\mathbb{R}^3$. Let $\mathbf{F} = \nabla\varphi$, where φ is a potential function with continuous second partial derivatives on D . Then $\nabla \times \mathbf{F} = \nabla \times \nabla\varphi = \mathbf{0}$; that is, the curl of the gradient is the zero vector and $\mathbf{F}$ is irrotational.

Divergence of the Curl

Suppose that $\mathbf{F} = \langle f, g, h \rangle$, where f , g , and h have continuous second partial derivatives. Then $\nabla \cdot (\nabla \times \mathbf{F}) = 0$: The divergence of the curl is zero.

Product Rule for the Divergence

Let u be a scalar-valued function that is differentiable on a region D and let $\mathbf{F}$ be a vector field that is differentiable on D . Then

$$\nabla \cdot (u\mathbf{F}) = \nabla u \cdot \mathbf{F} + u(\nabla \cdot \mathbf{F}) \quad (27)$$

Properties of a Conservative Vector Field

Let $\mathbf{F}$ be a conservative vector field whose components have continuous second partial derivatives on an open connected region D in $\mathbb{R}^3$. Then $\mathbf{F}$ has the following equivalent properties.

1. There exists a potential function φ such that $\mathbf{F} = \nabla\varphi$
2. $\int_C \mathbf{F} \cdot d\mathbf{r} = \varphi(B) - \varphi(A)$ for all points A and B in D and all smooth oriented curves C from A and B .
3. $\oint_C \mathbf{F} \cdot d\mathbf{r} = 0$ on all simple smooth closed oriented curves C in D .
4. $\nabla \times \mathbf{F} = \mathbf{0}$ at all points of D .

15.6 Surface Integrals

Surface Integrals of Scalar-Valued Functions on Parameterized Surface

Let f be a continuous function on a smooth surface S given parametrically by $\mathbf{r}(u, v) = \langle x(u, v), y(u, v), z(u, v) \rangle$, where $R = \{(u, v) : a \leq u \leq b, c \leq v \leq d\}$. Assume also that the tangent vectors $\mathbf{t}_u = \frac{\partial \mathbf{r}}{\partial u} = \langle \frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u} \rangle$, and $\mathbf{t}_v = \frac{\partial \mathbf{r}}{\partial v} = \langle \frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v} \rangle$ are continuous on R and the normal vectors $\mathbf{n} = \mathbf{t}_u \times \mathbf{t}_v$ is nonzero on R . Then the **surface integral** of the scalar-valued function f over S is

$$\iint_S f(x, y, z) dS = \iint_R f(x(u, v), y(u, v), z(u, v)) |\mathbf{t}_u \times \mathbf{t}_v| dA \quad (28)$$

If $f(x, y, z) = 1$, the integral equals the surface area of S .

Evaluation of Surface Integrals of Scalar-Valued Functions on Explicitly Defined Surfaces

Let f be a continuous function on a smooth surfaces S given by $z = g(x, y)$, for (x, y) in a region R . The surface integral of f over S is

$$\iint_S f(x, y, z) dS = \iint_R f(x, y, g(x, y)) \sqrt{z_x^2 + z_y^2 + 1} dA \quad (29)$$

If $f(x, y, z) = 1$, the surface integral equals the area of the surface.

Surface Integral of a Vector Field

Suppose $\mathbf{F} = \langle f, g, h \rangle$ is a continuous vector field on a region of $\mathbb{R}^3$ containing a smooth oriented surface S . If S is defined parametrically as $\mathbf{r}(u, v) = \langle x(u, v), y(u, v), z(u, v) \rangle$, for (u, v) is a region R , then

$$\iint_S \mathbf{F} \cdot \mathbf{n} dS = \iint_R \mathbf{F} \cdot (\mathbf{t}_u \times \mathbf{t}_v) dA \quad (30)$$

where $\mathbf{t}_u = \frac{\partial \mathbf{r}}{\partial u} = \langle \frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u} \rangle$ and $\mathbf{t}_v = \frac{\partial \mathbf{r}}{\partial v} = \langle \frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v} \rangle$ are continuous on R , the normal vector $\mathbf{n} = \mathbf{t}_u \times \mathbf{t}_v$ is nonzero on R , and the direction of $\mathbf{n}$ is

consistent with the orientation of S . If S is defined in the form $z = g(x, y)$, for (x, y) in a region R , then

$$\iint_S \mathbf{F} \cdot \mathbf{n} \, dS = \iint_R (-fz_x - gz_y + h) \, dA \quad (31)$$

15.7 Stokes' Theorem

Let S be a smooth oriented surface in $\mathbb{R}^3$ with a smooth closed boundary C whose orientation is consistent with that of S . Assume that $\mathbf{F} = \langle f, g, h \rangle$ is a vector field whose components have continuous first partial derivatives on S . Then

$$\oint_C \mathbf{F} \cdot d\mathbf{r} = \iint_S (\nabla \times \mathbf{F}) \cdot \mathbf{n} \, dS \quad (32)$$

where $\mathbf{n}$ is the unit vector normal to S determined by the orientation of S .

Curl $\mathbf{F} = \mathbf{0}$ Implies $\mathbf{F}$ is Conservative

Suppose that $\nabla \times \mathbf{F} = \mathbf{0}$ throughout an open simply connected region D of $\mathbb{R}^3$. Then $\oint_C \mathbf{F} \cdot d\mathbf{r} = 0$ on all closed simple smooth curves C in D and $\mathbf{F}$ is a conservative vector field on D .

15.8 Divergence Theorem

Let $\mathbf{F}$ be a vector field whose components have continuous first partial derivatives in a connected and simply connected region D in $\mathbb{R}^3$ enclosed by a smooth oriented surface S . Then

$$\iint_S \mathbf{F} \cdot \mathbf{n} \, dS = \iiint_D \nabla \cdot \mathbf{F} \, dV \quad (33)$$

where $\mathbf{n}$ is the unit outward normal vector on S .

Divergence Theorem for Hollow Regions

Suppose the vector field $\mathbf{F}$ satisfies the conditions of the Divergence Theorem on a region D bounded by two smooth oriented surfaces S_1 and S_2 , where S_1 lies within S_2 . Let S be the entire boundary of D ($S = S_1 \cup S_2$) and let $\mathbf{n}_1$ and $\mathbf{n}_2$ be the outward unit normal vectors for S_1 and S_2 , respectively. Then

$$\iiint_D \nabla \cdot \mathbf{F} \, dV = \iint_S \mathbf{F} \cdot \mathbf{n} \, dS = \iint_{S_2} \mathbf{F} \cdot \mathbf{n}_2 \, dS + \iint_{S_1} \mathbf{F} \cdot \mathbf{n}_1 \, dS \quad (34)$$

MATH3304

Differential Equations

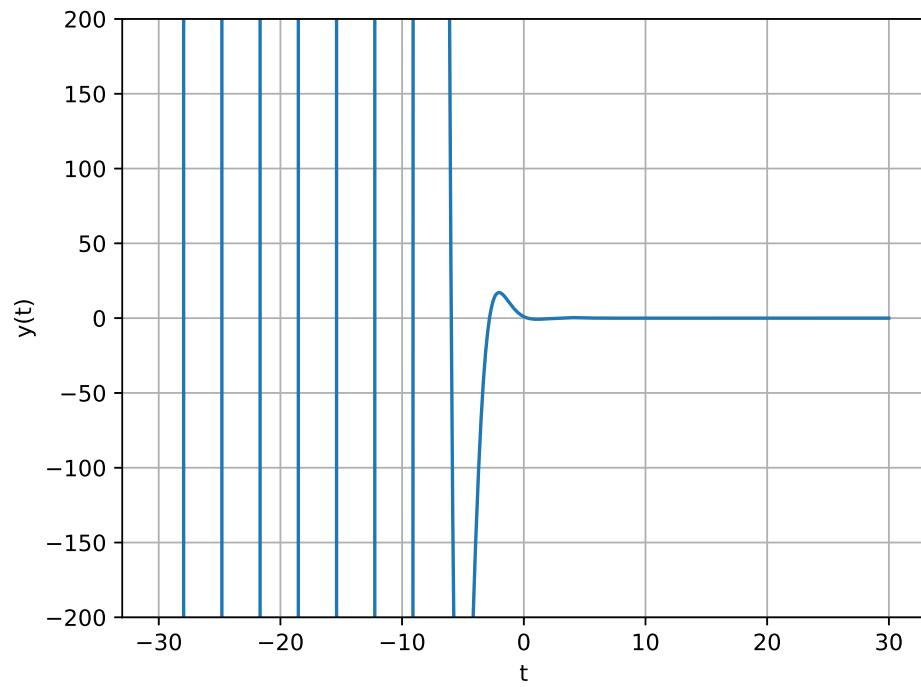


Homework Graphs

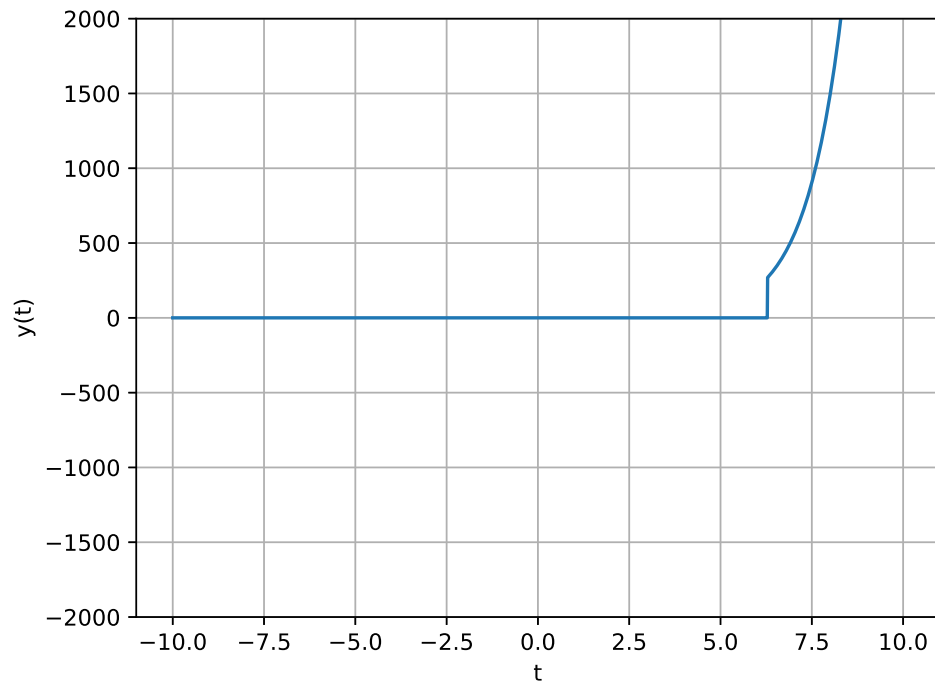
Illya Starikov

April 11, 2017

6.6 Problem #1



6.6 Problem #7



PHIL3225

Business Ethics



Business Portfolio

Illya Starikov

Due Date: May 10th, 2017

Word count: **1,339 words**.

1 Nature of Your Company

Currently, in the United States, one of the most controversial topics is self-driving cars. Google has notoriously been famous in the field for producing one of the earliest — and most famous — self-driving cars. This paved the way for Tesla to take said cars into the mainstream, with their Autopilot functionality. This made other companies take notice.

Since Tesla has shipped Autopilot in October of 2015, BMW, Mercedes Benz, Ford, and others have announced plans to create a fully-autonomous self-driving car. And why would a company not join the initiative? At the time of this writing, there are there are approximately 30,000 deaths in the United States *every year* caused by drivers. And these are not accidents where neither driver is to blame — most of these accidents **are human error**. By joining the fight to create a self-driving car, I would actually be fighting to save human lives — not just a few lives, but *saving millions of lives*.

Sidestepping the fight for humanity, my self-driving car business *will* become indispensable to humanity — as someone who has rode in a self-driving car, I can vouch for the utility. Getting over the initial hump that one is no longer in control (usually taking about 5 minutes), sitting back and *actually enjoying* driving is incomparable. Aside from the relaxation, my self-driving car company would provide much more utility, such as:

- Reduce traffic by preventing accidents and being able to know when to accelerate and decelerate in congested traffic

- Do mundane errands, like pick up groceries, people, etc.
- Virtually have no need for car insurance, saving the consumer hundreds-thousands of dollars a year

For reasons listed above, marketing the product is incredibly easily – so much so, advertising will not be necessary. Taking the Tesla approach (not spending at all on advertising) would actually be a better strategy for my company. At this point in the self-driving car life cycle, advertising would only cause a loss in focus and add additional expense to software. The same philosophy would initially apply to stakeholders. The self-driving car business would have too much politics for outside influence, so it would be ideal to be a private company with just initial funding from angel investors.

However, in a juxtaposition with that, everyone would *technically* have a stake in the company. People will come to rely on this technology; however, it *is dangerous*. A 2-ton machine made of aluminum going over 80 kph is nothing to scoff at, **it will kill you**. Because so many lives are at stake, it's imperative to make sure the algorithm works perfectly.

When considering my personal business, I would solely work on the software of the self driving car. Disconnecting from the hardware (except for the camera sensors) allows for actual focus on the brain (i.e., the algorithm). By abstracting the actual mechanics of the car, it would allow the company to focus on the major factors that matter. This would include:

- The specialized artificial intelligence. This would actually determine when to break, where to turn, how far ahead to stop, what to actually do.
- The visualization and camera sensors. This would mean taking in live video and able to analyze what is happening.
- The quality assurance. Again, there will be people at risk. It *has to be perfect*, so a dedicated team to make sure the code is flawless would be necessary.

Taking these into consideration, the team size would have to be *large* – a guess would be at least 250-500. It's better to think the size of the teams as percentages. Roughly 70% of the team would be solely dedicated to engineering the algorithm, 10% for the cameras and sensors, and the 20%

leftover would work on quality assurance. The majority 80% would simply be programmers, each with dedicated skills in

- Artificial Intelligence and Machine Learning
- Camera Input/Outputs
- Code Architecture

2 Moral Dilemmas

When considering the company, three major moral dilemmas arise:

1. The Trolley-esque problem
2. The “I am taking millions of jobs” problem
3. The luddite problem

2.1 The Trolley-esque

Consider the trolley problem

Suppose you are standing on train tracks. You notice on one side of the tracks, there is an oncoming trolley. On the other side, there is a fork in the tracks. On one side, there is a person tied to the tracks; on the other, five people are tied. The tracks are oriented towards the five people; however, there is a lever to divert the tracks. Do you save five at the expense of one?

This exactly applies to self driving cars. If one is driving, there might come a situation where there is a brake failure, and the artificial intelligence in the car might have to either kill the one passenger or the five pedestrians. This will be at the hands of the programmer to decide.

Is it the responsibility of the car to protect the driver, or take a utilitarian point and kill the driver for the five pedestrians? These are decisions I, the CEO, will have to make on behalf of the self driving cars and my company.

2.2 The “I Am Taking Millions of Jobs” Problem

It is no secret that self driving cars will take over *millions* of jobs; this involves many chauffeurs, taxi drivers, truckers, and more.

At the hand of my company, people who have to support their families will be out of a job because of my company.

2.3 The Luddite Problem

As they did with the cotton mill, luddites will definitely try to prevent self-driving car wave. Unfortunately, unlike the era of the cotton mill, many luddites will also take the form of congressmen; furthermore, not only will there be a lot of opposition from the house and senate, but other countries as well. Politically, there will be a huge barrier for me to overcome.

3 Moral Solutions

Below I will list the solutions to the above problem

3.1 The Trolly-esque Solution

I always hold the philosophy that the simplest solution is *generally* the best one. And that is a philosophy I will apply here. In reality, this situation will arise so little of the time (in a electric car, with about 10 moving parts, break failure is very uncommon), that I would not even program the car to do anything explicitly. Programming the car to protect the driver adds unnecessary code. The car will do a calculation of all methods of course correction, and whatever is the most optimal, I will force it to chose.

However, **I would never make this known**. When asked how my car handles this problem, I will never comment. If I say the car chooses the driver, then the argument could be made that I choose one life over five. If I say it chooses the five people, people would be scared to use the car. It is a lose-lose situation.

3.2 The “I Am Taking Millions of Jobs” Solution

There is no *real* good solution to this. This is an intrinsic problem of the self driving car business. Although I would creating a couple hundred jobs,

it will take millions from hard working people. I would have to hope the government could create a transition system, and possibly lobby for it. But as just what my company could do, there is next to nothing.

3.3 The Luddite Problem

This is also an intrinsic problem of the company, however more easily fixed. The choice is obvious: jobs are replaceable, human life is not. So self driving cars *should* be accepted by the government.

The best solution would be to create a coalition with other companies to fight the government's stance on self driving cars. I feel it is a moral duty, not just in a Kantian and a Utilitarian context, but in a *these save lives* context. As someone who has almost lost his life and his mother's life in a car wreck, I feel it is my duty to create market adoption.

MATH1214

Calculus I

S&TTM

Calculus I: Single-Variable Calculus

Illya Starikov

August 6, 2025

Contents

0	Functions	2
0.1	Review of Functions	2
0.2	Review of Functions	2
0.3	Representing Functions	3
0.4	Inverse, Exponential, and Logarithmic Functions	5
0.5	Trigonometric Functions and Their Inverses	7
1	Limits	8
1.2	Definitions of Limits	8
1.3	Techniques For Computing Limits	9
1.4	Infinite Limits	10
1.5	Limits at Infinity	11

num₁!

0 Functions

0.1 Review of Functions

A **function** is a rule that assigns to each value x in a set D a unique value denoted $f(x)$. The set D is the **domain** of the function. The **range** is the set of all values of $f(x)$ produced as x varies over the domain.

Vertical Line Test

A graph represents a function if and only if it passes the **vertical line test**. Every vertical line intersects the graph at most once. A graph that fails this test does not represent a function.

Composite Functions

Given two functions f and g , the composite functions $f \circ g$ is defined by $(f \circ g)(x) = f(g(x))$. It is evaluated in two steps: $y = f(u)$, where $u = g(x)$. The domain of $f \circ g$ consists of all x in the domain of g such that $u = g(x)$ is in the domain of f .

Symmetry in Functions

An **even function** has the property that $f(-x) = f(x)$, for all x in the domain. The graph of an even function is symmetric about the y-axis. Polynomials consisting of only even powers of the variable (of the form x^{2n} , where n is a nonnegative integer) are even functions.

An **odd function** f has the property that $f(-x) = -f(x)$, for all x in the domain. The graph of an odd function is symmetric about the origin. Polynomials consisting of only odd powers of the variable (of the form x^{2n+1} , where n is a nonnegative integer) are odd functions.

0.2 Review of Functions

A **function** is a rule that assigns to each value x in a set D a unique value denoted $f(x)$. The set D is the **domain** of the function. The **range** is the

set of all values of $f(x)$ produced as x varies over the domain.

Vertical Line Test

A graph represents a function if and only if it passes the **vertical line test**. Every vertical line intersects the graph at most once. A graph that fails this test does not represent a function.

Composite Functions

Given two functions f and g , the composite functions $f \circ g$ is defined by $(f \circ g)(x) = f(g(x))$. It is evaluated in two steps: $y = f(u)$, where $u = g(x)$. The domain of $f \circ g$ consists of all x in the domain of g such that $u = g(x)$ is in the domain of f .

Symmetry in Functions

An **even function** has the property that $f(-x) = f(x)$, for all x in the domain. The graph of an even function symmetric about the y-axis. Polynomials consisting of only even powers of the variable (of the form x^{2n} , where n is a nonnegative integer) are even functions.

An **odd function** f has the property that $f(-x) = -f(x)$, for all x in the domain. The graph of an odd function is symmetric about the origin. Polynomials consisting of only odd powers of the variable (of the form x^{2n+1} , where n is a nonnegative integer) are odd functions.

0.3 Representing Functions

Some brief families of functions can include

Polynomials are functions of the form

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0,$$

where the **coefficients** $a_0, a_1, \dots, a_n$ are real numbers with $a_n \neq 0$ and the nonnegative integer n is the **degree** of the polynomial. The domain of any polynomial is the set of all real numbers. An n th-degree polynomial can have as many as n real **zeros** or **roots** — values of x .

Rational Functions are ratios of the form $f(x) = p(x)/q(x)$, where p and q are polynomials. Because division by zero is prohibited, the domain of a rational function is the set of all real numbers except those for which the denominator is zero.

Algebraic Functions are constructed using the operations of algebra: addition, subtraction, multiplication, division, and roots. Examples of algebraic functions are $f(x) = \sqrt{2x^3 + 4}$ and $f(x) = x^{1/4}(x^3 + 2)$. In general, if an even root (square root, fourth root, and so forth) appears, then the domain does not contain points at which the quantity under the root is negative (and perhaps other points).

Exponential Functions have the form $f(x) = b^x$, where the base $b \neq 1$ is a positive real number. Closely associated with exponential functions are logarithmic functions of the form $f(x) = \log_b x$, where $b > 0$ and $b \neq 1$. An exponential function has a domain consisting of all real numbers. Logarithmic functions are defined for positive, real numbers. The most important function is the **natural exponential function** $f(x) = e^x$, with base $b = e$, where $e \approx 2.71828\dots$ is one of the fundamental constants of mathematics. Associated with the natural exponential function is the **natural logarithmic function** $f(x) = \ln x$, which also has the base $b = e$.

Trigonometric Functions are $\sin x$, $\cos x$, $\tan x$, $\sec x$, and $\csc x$; they are fundamental to mathematics and many areas of application. Also important are their relatives, the **inverse trigonometric functions**.

Transcendental Functions Trigonometric, exponential, and logarithmic functions are few examples of a large family called transcendental functions.

Transformations

Given the real numbers a , b , c , and d and the function f , the graph of $y = cf(a(x - b)) + d$ is obtained from the graph of $y = f(x)$ in the following steps.

$$\begin{array}{lcl}
y = f(x) & \xrightarrow{\text{horizontal scaling by a factor of } |a|} & y = f(ax) \\
& \xrightarrow{\text{horizontal shift by } b \text{ units}} & y = f(a(x - b)) \\
& \xrightarrow{\text{vertical scaling by a factor of } |c|} & y = cf(a(x - b)) \\
& \xrightarrow{\text{horizontal scaling by a factor of } |a|} & y = cf(a(x - b)) + d
\end{array}$$

0.4 Inverse, Exponential, and Logarithmic Functions

The Natural Exponential Function

The **natural exponential function** is $f(x) = e^x$, which as the base $e = 2.718281828459\dots$

Inverse Function

Given a function f , its inverse (if it exists) is a function f^{-1} such that whenever $y = f(x)$, then $f^{-1}(y) = x$.

One-to-One Functions and the Horizontal Line Test

A function f is **one-to-one** on a domain D if each value of $f(x)$ corresponds to exactly one value of x in D . More precisely, f is one-to-one on D if $y(x_1) \neq f(x_2)$ whenever $x_1 \neq x_2$ for x_1 and x_2 in D . The **horizontal line test** says that every horizontal line intersects the graph of a one-to-one function at most once.

Existence of Inverse Functions

Let f be one-to-one function on a domain D with a range R . Then f has a unique inverse f^{-1} with domain R and range D such that

$$f^{-1}(f(x)) = x \quad \text{and} \quad f(f^{-1}(y)) = y,$$

where x is in D and y is in R .

Finding an Inverse Function

Suppose f is one-to-one on an interval I . To find f^{-1} :

- Solve $y = f(x)$ for x . If necessary, choose the function that corresponds to I .
- Interchange x and y and write $y = f^{-1}(x)$.

Logarithmic Function Base b

For any base $b > 0$, with $b \neq 1$, the **logarithmic function base b** , denoted $y = \log_b x$, is the inverse of the exponential function $y = b^x$. The inverse of the natural exponential function with base $b = e$ is the **natural logarithm function**, denoted $y = \ln x$.

Inverse Relations For Exponential and Logarithmic Functions

For any base $b > 0$, with $b \neq 1$, the following inverse relations hold:

- $b^{\log_b x} = x$, for $x > 0$
- $\log_b b^x = x$, for any real values of x

Change-of-Base Rules

Let b be a positive real number with $b \neq 1$. Then

$$b^x = e^{x \ln b}, \text{ for all } x \quad \text{and} \quad \log_b x = \frac{\ln x}{\ln b}, \text{ for } x > 0$$

More generally, if c is a positive real number with $c \neq 1$, then

$$b^x = c^{x \log_c b}, \text{ for all } x \quad \text{and} \quad \log_b x = \frac{\log_c x}{\log_c b}, \text{ for } x > 0$$

0.5 Trigonometric Functions and Their Inverses

Let $P(x, y)$ be a point on a circle of radius r associated with the angle θ . Then

$$\sin \theta = \frac{y}{r} \quad \cos \theta = \frac{x}{r} \quad \tan \theta = \frac{y}{x} \quad (1)$$

$$\cot \theta = \frac{x}{y} \quad \sec \theta = \frac{r}{x} \quad \csc \theta = \frac{r}{y} \quad (2)$$

$$(3)$$

Trigonometric Identities

Reciprocal Identities

$$\tan \theta = \frac{\sin \theta}{\cos \theta} \quad \cot \theta = \frac{1}{\tan \theta} = \frac{\cos \theta}{\sin \theta} \quad (4)$$

$$\csc \theta = \frac{1}{\sin \theta} \quad \sec \theta = \frac{1}{\cos \theta} \quad (5)$$

Pythagorean

$$\sin^2 \theta + \cos^2 \theta = 1 \quad 1 + \cot^2 \theta = \csc^2 \theta \quad \tan^2 \theta + 1 = \sec^2 \theta \quad (6)$$

Double- and Half-Angle Formulas

$$\sin 2\theta = 2 \sin \theta \cos \theta \quad \cos 2\theta = \sin^2 \theta - \cos^2 \theta \quad (7)$$

$$\cos^2 \theta = \frac{1 + \cos 2\theta}{2} \quad \sin^2 \theta = \frac{1 - \cos 2\theta}{2} \quad (8)$$

Period of Trigonometric Function

The function $\sin \theta$, $\cos \theta$, $\sec \theta$, and $\csc \theta$ have a period of 2π

$$\sin(\theta + 2\pi) = \sin \theta \quad \cos(\theta + 2\pi) = \cos \theta \quad (9)$$

$$\sec(\theta + 2\pi) = \sec \theta \quad \csc(\theta + 2\pi) = \csc \theta \quad (10)$$

for all θ in the domain.

The functions $\tan \theta$ and $\cot \theta$ have a period of π :

$$\tan(\theta + \pi) = \tan \theta \quad \cot(\theta + \pi) = \cot \theta \quad (11)$$

for all θ in the domain.

Inverse Sine and Cosine

$y = \sin^{-1} x$ is the value of y such that $x = \sin y$, where $-\frac{\pi}{2} \leq y \leq \frac{\pi}{2}$.
 $y = \cos^{-1} x$ is the value of y such that $x = \cos y$, where $0 \leq y \leq \pi$. The domain of both $\sin^{-1} x$ and $\cos^{-1} x$ is $\{x : -1 \leq x \leq 1\}$.

Other Inverse Trigonometric Functions

- $\tan^{-1} x$ is the value of y such that $x = \tan y$, where $-\frac{\pi}{2} < \frac{\pi}{2}$.
- $\cot^{-1} x$ is the value of y such that $x = \tan y$, where $0 < y < \pi$.

The domain of both $\tan^{-1} x$ and $\cot^{-1} x$ is $\{x : -\infty < x < \infty\}$

- $\sec^{-1} x$ is the value of y such that $x = \sec y$, where $0 < y < \pi$ with $y \neq \frac{\pi}{2}$
- $\tan^{-1} x$ is the value of y such that $x = \tan y$, where $-\frac{\pi}{2} < \frac{\pi}{2}$.

1 Limits

1.2 Definitions of Limits

Limits of a Function (Preliminary)

Suppose the function f is defined for all x near a except possibly at a . If $f(x)$ is arbitrarily close to L (as close to L as we like) for all x sufficiently close (but not equal) to a , we write

$$\lim_{x \rightarrow a} f(x) = L$$

and say the limit of $f(x)$ as x approaches a equals L .

One-Sided Limits

1 Right-sided limits Suppose f is defined for all x near a with $x > a$. If $f(x)$ is arbitrarily close to L for all x sufficiently close to a with $x > a$, we write

$$\lim_{x \rightarrow a^+} f(x) = L \tag{12}$$

and say the limit of $f(x)$ as x approaches a from the right equals L .

2 Left-sided limits Suppose f is defined for all x near a with $x < a$. If $f(x)$ is arbitrarily close to L for all x sufficiently close to a with $x < a$, we write

$$\lim_{x \rightarrow a^-} f(x) = L \quad (13)$$

and say the limit of $f(x)$ as x approaches a from the left equals L .

Relationship Between One-Sided and Two-Sided Limits

Assume f is defined for all x near a except possibly at a . Then $\lim_{x \rightarrow a} f(x) = L$ if and only if $\lim_{x \rightarrow a^-} f(x) = \lim_{x \rightarrow a^+} f(x) = L$.

1.3 Techniques For Computing Limits

Limits of Linear Functions

Let a , b , and m be real numbers. For Linear functions $f(x) = mx + b$,

$$\lim_{x \rightarrow a} f(x) = f(a) = ma + b \quad (14)$$

Limit Laws

Assume $\lim_{x \rightarrow a} f(x)$ and $\lim_{x \rightarrow a} g(x)$ exist. The following properties hold, where c is a real number, and $m > 0$ and $n > 0$ are integers.

1 Sum $\lim_{x \rightarrow a} [f(x) + g(x)] = \lim_{x \rightarrow a} f(x) + \lim_{x \rightarrow a} g(x)$

2 Difference $\lim_{x \rightarrow a} [f(x) - g(x)] = \lim_{x \rightarrow a} f(x) - \lim_{x \rightarrow a} g(x)$

3 Constant Multiple $\lim_{x \rightarrow a} [c f(x)] = c \lim_{x \rightarrow a} f(x)$

4 Product $\lim_{x \rightarrow a} [f(x)g(x)] = \left[\lim_{x \rightarrow a} f(x) \right] \left[\lim_{x \rightarrow a} g(x) \right]$

5 Quotient $\lim_{x \rightarrow a} \left[\frac{f(x)}{g(x)} \right] = \frac{\lim_{x \rightarrow a} f(x)}{\lim_{x \rightarrow a} g(x)}$, provided $\lim_{x \rightarrow a} g(x) \neq 0$

6 Power $\lim_{x \rightarrow a} [f(x)]^n = \left[\lim_{x \rightarrow a} f(x) \right]^n$

7 Fractional Power $\lim_{n \rightarrow \infty} [f(x)]^{n/m} = \left[\lim_{n \rightarrow \infty} f(x) \right]^{n/m}$, provided $f(x) \geq 0$, for x near a , if m is even and n/m is reduced to lowest terms.

Limits of Polynomial and Rational Functions

Assume p and q are polynomials and a is a constant

- Polynomial functions: $\lim_{n \rightarrow \infty} p(x) = p(a)$
- Rational functions: $\lim_{n \rightarrow \infty} \frac{p(x)}{q(x)} = \frac{p(a)}{q(a)}$, provided $q(a) \neq 0$

Limit Laws For One-Sided Limits

Laws 1–6 hold with $\lim_{n \rightarrow \infty}$ replaced by $\lim_{n \rightarrow \infty}$ or $\lim_{n \rightarrow \infty}$. Law 7 is modified as follows, assume $m > 0$ and $n > 0$ are integers.

7 Fractional Power

- $\lim_{n \rightarrow \infty} [f(x)]^{n/m}$, provided $f(x) \geq 0$, for x near a with $x > a$, if m is even and n/m is reduced to lowest terms
- $\lim_{n \rightarrow \infty} [f(x)]^{n/m}$, provided $f(x) \geq 0$, for x near a with $x < a$, if m is even and n/m is reduced to lowest terms

The Squeeze Theorem

Assume the function f , g , and h satisfy $f(x) \leq g(x) \leq h(x)$, for all values of x near a , except possibly at a . If $\lim_{n \rightarrow \infty} f(x) = \lim_{n \rightarrow \infty} h(x) = L$, then $\lim_{n \rightarrow \infty} g(x) = L$.

1.4 Infinite Limits

Suppose f is defined for all x near a . If $f(x)$ grows arbitrarily large for all x sufficiently close (but not equal) to a , we write

$$\lim_{x \rightarrow a} f(x) = \infty \quad (15)$$

We say the limit of $f(x)$ as x approaches a is infinity.

If $f(x)$ is negative and grows arbitrarily large in magnitude for all x sufficiently close (but not equal) to a , we write

$$\lim_{x \rightarrow a} f(x) = -\infty \quad (16)$$

In this case, we say the limit of $f(x)$ as x approaches a is negative infinity. In both cases, *the limit does not exist*.

One-Sided Infinite Limits

Suppose f is defined for all x near a with $x > a$. If $f(x)$ becomes arbitrarily large for all x sufficiently close to a with $x > a$, we write $\lim_{x \rightarrow a^+} f(x) = \infty$. The one-sided infinite limit $\lim_{x \rightarrow a^-} f(x) = -\infty$, $\lim_{x \rightarrow a^+} f(x) = \infty$, and $\lim_{x \rightarrow a^-} f(x) = -\infty$ are defined analogously.

Vertical Asymptote

If $\lim_{x \rightarrow a^-} f(x) = \pm\infty$, $\lim_{x \rightarrow a^+} f(x) = \pm\infty$ or $\lim_{x \rightarrow a} f(x) = \pm\infty$, the line $x = a$ is called a **vertical asymptote** of f .

1.5 Limits at Infinity

If $f(x)$ becomes arbitrarily close to a finite number L for all sufficiently large and positive x , then we write

$$\lim_{x \rightarrow \infty} f(x) = L \quad (17)$$

We say the limit of $f(x)$ as x approaches infinity is L . In this case the line $y = L$ is a **horizontal asymptote** of f . The limit at negative infinity, $\lim_{x \rightarrow -\infty} f(x) = M$, is defined analogously. When the limit exists, the horizontal asymptote is $y = M$.

Infinite Limits at Infinity

If $f(x)$ becomes arbitrarily large as x becomes arbitrary large, then we write

$$\lim_{x \rightarrow \infty} f(x) = \infty \quad (18)$$

The limits $\lim_{x \rightarrow -\infty} f(x) = -\infty$, $\lim_{x \rightarrow -\infty} f(x) = \infty$, and $\lim_{x \rightarrow \infty} f(x) = -\infty$ are defined similarly.

Limit of Infinity at Powers and Polynomials

Let n be a positive integer and let p be the polynomial $p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$, where $a_n \neq 0$.

1. $\lim_{n \rightarrow \infty} x^n = \infty$ when n is even.
2. $\lim_{n \rightarrow \infty} x^n = \infty$ and $\lim_{n \rightarrow \infty} x^n = -\infty$ when n is odd.
3. $\lim_{n \rightarrow \infty} \frac{1}{x^n} = \lim_{n \rightarrow \infty} x^{-n} = 0$
4. $\lim_{n \rightarrow \infty} p(x) = \infty$ or $-\infty$, depending on the degree of the polynomial or the leading coefficient of a_n .

End Behavior and Asymptotes of Rational Functions

Suppose $f(x) = \frac{p(x)}{q(x)}$ is a rational function, where

$$p(x) = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_2 x^2 + a_1 x + a_0 \quad (19)$$

and

$$q(x) = b_n x^n + b_{n-1} x^{n-1} + \cdots + b_2 x^2 + b_1 x + b_0 \quad (20)$$

with $a_m \neq 0$ and $b_n \neq 0$.

1. If $m < n$ then $\lim_{n \rightarrow \infty} f(x) = 0$, and $y = 0$ is a horizontal asymptote of f .
2. If $m = n$, then $\lim_{n \rightarrow \infty} f(x) = a_m/b_n$, and $y = a_m/b_n$ is a horizontal asymptote.
3. If $m > n$, then $\lim_{n \rightarrow \infty} f(x) = \infty$ or $-\infty$, and f has no horizontal asymptote.
4. If $m = n + 1$, then $\lim_{n \rightarrow \infty} f(x) = \infty$ or $-\infty$, f has no horizontal asymptote, but f has a slant asymptote.
5. Assuming that $f(x)$ is in reduced form (p and q share no common factors), vertical asymptotes occur at the zeros of q .

End Behavior of e^x , e^{-x} , and $\ln x$

The end behavior for e^x and e^{-x} on $(-\infty, \infty)$ and $\ln x$ on $(0, \infty)$ is given by the following limits:

$$\lim_{x \rightarrow \infty} e^x = \infty \quad \text{and} \quad \lim_{x \rightarrow -\infty} e^x = 0 \quad (21)$$

$$\lim_{x \rightarrow \infty} e^{-x} = 0 \quad \text{and} \quad \lim_{x \rightarrow -\infty} e^{-x} = \infty \quad (22)$$

$$\lim_{x \rightarrow 0^+} \ln x = -\infty \quad \text{and} \quad \lim_{x \rightarrow \infty} \ln x = \infty \quad (23)$$

MATH1215

Calculus II



Calculus II: Complete Notes

Illya Starikov

Missouri S&T

Contents

6	Applications of Integration	3
6.1	Physical Applications	3
7	Logarithmic, Exponential, and Inverse Trigonometric Functions	4
7.1	Inverse Function	4
7.2	The Natural Logarithmic and Exponential Functions	5
7.3	Logarithmic and Exponential Functions with Other Bases	7
7.4	Inverse Trigonometric Functions	9
7.5	L'Hôpital's Rule and Growth Rates of Functions	10
7.6	Hyperbolic Functions	11
8	Integration Techniques	14
8.1	Basic Approaches	14
8.2	Integration By Parts	15
8.3	Trigonometric Integrals	16
8.4	Trigonometric Substitutions	18
8.5	Partial Fractions	19
8.6	Improper Integrals	21
9	Sequences and Infinite Series	23
9.1	An Overview	23
9.2	Sequences	25
9.3	Infinite Series	27
9.4	The Divergence and Integral Tests	28

9.5	The Ratio, Root, and Comparison Tests	30
9.6	Alternating Series	32
10	Power Series	35
10.1	Approximating Functions with Polynomials	35
10.2	Properties of Power Series	37
10.3	Taylor Series	39
11	Parametric and Polar Curves	41
11.1	Parametric Equations	41
11.2	Polar Coordinates	42
11.3	Calculus in Polar Coordinates	49

6 Applications of Integration

6.1 Physical Applications

Mass of a One-Dimensional Object

Suppose a thin bar or wire is represented by a line segment on the interval $a \leq x \leq b$ with a density function ρ (with units of mass per length). The mass of the object is

$$m = \int_a^b \rho(x) dx \quad (1)$$

Work

The work done by a variable force F in moving an object along a line from $x = a$ to $x = b$ in the direction of the force is

$$W = \int_a^b F(x) dx \quad (2)$$

Solving Lifting Problems

1. Draw a y -axis in the vertical direction (parallel to gravity) and choose a convenient origin. Assume the interval $[a, b]$ corresponds to the vertical extent of the fluid.
2. For $a \leq y \leq b$, find the cross-sectional area $A(y)$ of the horizontal slices and the distance $D(y)$ the slices must be lifted.
3. The work required to lift the water is

$$W = \int_a^b \rho g A(y) D(y) dy \quad (3)$$

Solving Force/Pressure Problems

1. Draw a y -axis on the face of the dam in the vertical direction and choose a convenient origin (often taken to be the base of the dam).

2. Find the width function $w(y)$ for each value of y on the face of the dam.
3. If the base of the dam is at $y = 0$ and the top of the dam is at $y = a$, then the total force on the dam is

$$F = \int \rho g(a - y)w(y) dy \quad (4)$$

7 Logarithmic, Exponential, and Inverse Trigonometric Functions

7.1 Inverse Function

Derivative of the Inverse Function

Let f be differentiable and have an inverse on an interval I . If x_0 is a point of I at which $f'(x_0) \neq 0$, then f^{-1} is differentiable at $y_0 = f(x_0)$ and

$$(f^{-1})'(y_0) = \frac{1}{f'(x_0)}, \text{ where } y_0 = f(x_0) \quad (5)$$

7.2 The Natural Logarithmic and Exponential Functions

The Natural Logarithm

The **natural logarithm** of a number $x > 0$, denoted $\ln x$, is defined

$$\ln x = \int_1^x \frac{dt}{t} \quad (6)$$

Properties of the Natural Logarithm

1. The domain and range of $\ln x$ are $(0, \infty)$ and $(-\infty, \infty)$, respectively.
2. $\ln(xy) = \ln x + \ln y, \forall x, y \in \mathbf{R}^+$
3. $\ln(x/y) = \ln x - \ln y, \forall x, y \in \mathbf{R}^+$
4. $\ln x^p = p \ln x, \forall x \in \mathbf{Q}^+$
5. $\frac{d}{dx}(\ln |x|) = \frac{1}{x}, \forall x \in \mathbf{R} \wedge x \neq 0$
6. $\frac{d}{dx}(\ln |u(x)|) = \frac{u'(x)}{u(x)}$
7. $\int \frac{dx}{x} = \ln |x| + C$

The Number e

The number e is the real number that satisfies

$$\ln e = \int_1^e \frac{dt}{t} = 1 \quad (7)$$

The Exponential Function

$\forall x, y \in \mathbf{R}$

$$y = e^x \Leftrightarrow x = \ln y \quad (8)$$

Properties of e

- $e^{x+y} = e^x e^y$
- $e^{x-y} = e^x / e^y$
- $(e^x)^y = e^{xy}, \forall y \in \mathbf{Q}$
- $\ln(e^x) = x, \forall x \in \mathbf{R}$
- $e^{\ln x} = x, \forall x \in \mathbf{R}^+$

Exponential Functions with General Bases

Let $b \in \mathbf{R}^+ \wedge b \neq 1. \forall x \in \mathbf{R}^+$,

$$b^x = e^{x \ln b} \quad (9)$$

Derivative and Integral of the Exponential Function

$\forall x \in \mathbf{R}$,

$$\frac{d}{dx}(e^{u(x)}) = e^{u(x)} u'(x) \quad (10)$$

$$\int e^x dx = e^x + C \quad (11)$$

7.3 Logarithmic and Exponential Functions with Other Bases

Logarithmic Function Base b

For any base $b > 0$, with $b \neq 1$, the **logarithmic function base b** , denoted $\log_b x$, is the inverse of the exponential function b^x .

Inverse Relations for Exponential and Logarithmic Functions

For any base $b > 0$, with $b \neq 1$, the following inverse relation holds.

- $b^{\log_b x} = x, \forall x \in \mathbf{R}^+$
- $\log_b b^x = x, \forall x$

Derivative of b^x

If $b > 0 \wedge b \neq 1, \forall x$,

$$\frac{d}{dx}(b^x) = b^x \ln b \quad (12)$$

Indefinite Integral of b^x

For $b > 0 \wedge b \neq 1$,

$$\int b^x dx = \frac{1}{\ln b} b^x + C \quad (13)$$

General Power Rule

$\forall p, x \in \mathbf{R}^+$,

$$\frac{d}{dx}(x^p) = px^{p-1} \quad (14)$$

Furthermore, if u is a positive differentiable function on its domain, then

$$\frac{d}{dx}(u(x)^p) = p(u(x))^{p-1} \times u'(x) \quad (15)$$

Derivative of $\log_b x$

If $b > 1$,

$$\frac{d}{dx}(\log_b x) = \frac{1}{x \ln b} \quad \forall x \in \mathbf{R}^+ \wedge x \neq 0 \quad (16)$$

$$\frac{d}{dx}(\log_b |x|) = \frac{1}{x \ln b} \quad \forall x \in \mathbf{R}^+ \wedge x \neq 0 \quad (17)$$

7.4 Inverse Trigonometric Functions

Derivative of Inverse Trigonometric Functions

$$\begin{aligned}\frac{d}{dx}(\sin^{-1} x) &= \frac{1}{\sqrt{1-x^2}} & \frac{d}{dx}(\cos^{-1} x) &= -\frac{1}{\sqrt{1-x^2}}, \{x \in \mathbf{R} \mid -1 < x < 1\} \\ \frac{d}{dx}(\tan^{-1} x) &= \frac{1}{1+x^2} & \frac{d}{dx}(\cot^{-1} x) &= -\frac{1}{1+x^2}, \{x \in \mathbf{R} \mid -\infty < x < \infty\} \\ \frac{d}{dx}(\sec^{-1} x) &= \frac{1}{|x|\sqrt{x^2-1}} & \frac{d}{dx}(\csc^{-1} x) &= -\frac{1}{|x|\sqrt{x^2-1}}, \{x \in \mathbf{R} \mid |x| > 1\}\end{aligned}$$

Integrals Involving Inverse Trigonometric Functions

$$\int \frac{dx}{\sqrt{a^2 - x^2}} = \sin^{-1} \frac{x}{a} + C \quad (18)$$

$$\int \frac{dx}{a^2 + x^2} = \frac{1}{a} \tan^{-1} \frac{x}{a} + C \quad (19)$$

$$\int \frac{dx}{x\sqrt{a^2 + x^2}} = \frac{1}{a} \sec^{-1} \left| \frac{x}{a} \right| + C \quad (20)$$

7.5 L'Hôpital's Rule and Growth Rates of Functions

Indeterminate forms $1^\infty, 0^0, \infty^0$

Assume $\lim_{x \rightarrow \infty} f(x)^{g(x)}$ has the indeterminate form $1^\infty, 0^0$, or ∞^0 .

1. Evaluate $L = \lim_{x \rightarrow \infty} g(x) \ln f(x)$. This limit can be put in the form $0/0$ or ∞/∞ , both of which are handled by l'Hôpital's rule.
2. Then $\lim_{x \rightarrow \infty} f(x)^{g(x)} = e^L$

Growth Rates of Functions (as $x \rightarrow \infty$)

Suppose f and g are functions with $\lim_{x \rightarrow \infty} f(x) = \lim_{x \rightarrow \infty} g(x) = \infty$. Then f **grows faster than g** as $x \rightarrow \infty$ if

$$\lim_{x \rightarrow \infty} \frac{g(x)}{f(x)} = 0 \quad \text{or, quantitatively, if} \quad \lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = \infty \quad (21)$$

The functions f and g have **comparable growth rates** if

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = M$$

where $M \in \mathbb{R}^+$.

Ranking Growth Rates as $x \rightarrow \infty$

Let $f \ll g$ mean that g grows faster than f as $f \rightarrow \infty$. With positive real numbers p, q, r, s and $b > 1$,

$$\ln^q x \ll x^p \ll x^p \ln^r x \ll x^{p+s} \ll b^x \ll x^x \quad (22)$$

7.6 Hyperbolic Functions

Hyperbolic Functions

Hyperbolic Cosine

$$\cosh x = \frac{e^x + e^{-x}}{2} \quad (23)$$

Hyperbolic Sine

$$\sinh x = \frac{e^x - e^{-x}}{2} \quad (24)$$

Hyperbolic Tangent

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (25)$$

Hyperbolic Cotangent

$$\coth x = \frac{\cosh x}{\sinh x} = \frac{e^x + e^{-x}}{e^x - e^{-x}} \quad (26)$$

Hyperbolic Secant

$$\operatorname{sech} x = \frac{1}{\cosh x} = \frac{2}{e^x + e^{-x}} \quad (27)$$

Hyperbolic Cosecant

$$\operatorname{csch} x = \frac{1}{\sinh x} = \frac{2}{e^x - e^{-x}} \quad (28)$$

Hyperbolic Identities

$$\cosh^2 x - \sinh^2 x = 1$$

$$1 - \tanh^2 x = \operatorname{sech}^2 x$$

$$\coth^2 x - 1 = \operatorname{csch}^2 x$$

$$\coth(x + y) = \cosh x \cosh y + \sinh x \sinh y$$

$$\sinh(x + y) = \sinh x \cosh y + \cosh x \sinh y$$

$$\cosh 2x = \cosh^2 x + \sinh^2 x$$

$$\cosh^2 x = \frac{\cosh 2x + 1}{2}$$

$$\cosh(-x) = \cosh x$$

$$\sinh(-x) = -\sinh x$$

$$\tanh(-x) = -\tanh x$$

$$\sinh 2x = 2 \sinh x \cosh x$$

$$\sinh^2 x = \frac{\cosh 2x - 1}{2}$$

Derivatives and Integral Formulas

1. $\frac{d}{dx}(\cosh x) = \sinh x \Rightarrow \int \sinh x \, dx = \cosh x + C$
2. $\frac{d}{dx}(\sinh x) = \cosh x \Rightarrow \int \cosh x \, dx = \sinh x + C$
3. $\frac{d}{dx}(\tanh x) = \operatorname{sech}^2 x \Rightarrow \int \operatorname{sech}^2 x \, dx = \tanh x + C$
4. $\frac{d}{dx}(\coth x) = -\operatorname{csch}^2 x \Rightarrow \int \operatorname{csch}^2 x \, dx = -\coth x + C$
5. $\frac{d}{dx}(\operatorname{sech} x) = -\operatorname{sech} x \tanh x \Rightarrow \int \operatorname{sech} x \tanh x \, dx = -\operatorname{sech} x + C$
6. $\frac{d}{dx}(\operatorname{csch} x) = -\operatorname{csch} x \coth x \Rightarrow \int \operatorname{csch} x \coth x \, dx = -\operatorname{csch} x + C$

Integrals of Hyperbolic Functions

1. $\int \tanh x \, dx = \ln \cosh x + C$
2. $\int \coth x \, dx = \ln |\sinh x| + C$
3. $\int \operatorname{sech} x \, dx = \tan^{-1} |\sinh x| + C$
4. $\int \operatorname{csch} x \, dx = \ln |\tanh(x/2)| + C$

Inverses of the Hyperbolic Functions Expressed as Logarithms

$$\begin{aligned}\cosh^{-1} x &= \ln(x + \sqrt{x^2 - 1}) \quad (x \geq 1) & \operatorname{sech}^{-1} x &= \cosh^{-1} \frac{1}{x} \quad (0 < x \leq 1) \\ \sinh^{-1} x &= \ln(x + \sqrt{x^2 + 1}) & \operatorname{csch}^{-1} x &= \sinh^{-1} \frac{1}{x} \quad (x \neq 0) \\ \tanh^{-1} x &= \frac{1}{2} \ln\left(\frac{1+x}{1-x}\right) \quad (|x| < 1) & \coth^{-1} x &= \tanh^{-1} \frac{1}{x} \quad (|x| > 1)\end{aligned}$$

Derivatives of the Inverse Hyperbolic Functions

$$\begin{aligned}\frac{d}{dx}(\cosh^{-1} x) &= \frac{1}{\sqrt{x^2 - 1}} \quad (x > 1) & \frac{d}{dx}(\sinh^{-1} x) &= \frac{1}{\sqrt{x^2 + 1}} \\ \frac{d}{dx}(\tanh^{-1} x) &= \frac{1}{1 - x^2} \quad (|x| < 1) & \frac{d}{dx}(\coth^{-1} x) &= \frac{1}{1 - x^2} \quad (|x| > 1) \\ \frac{d}{dx}(\operatorname{sech}^{-1} x) &= -\frac{1}{x\sqrt{1 - x^2}} \quad (0 < x < 1) & \frac{d}{dx}(\operatorname{csch}^{-1} x) &= -\frac{1}{|x|\sqrt{1 + x^2}} \quad (x \neq 0)\end{aligned}$$

Integral Formulas

1. $\int \frac{dx}{\sqrt{x^2 - a^2}} = \cosh^{-1} \frac{x}{a} + C, \text{ for } x > a$
2. $\int \frac{dx}{x^2 + a^2} = \sinh^{-1} \frac{x}{a} + C, \text{ for all } x$

3. $\int \frac{dx}{a^2-x^2} = \frac{1}{a} \tanh^{-1} \frac{x}{a} + C$, for $|x| < a = \frac{1}{a} \coth^{-1} \frac{x}{a} + C$, for $|x| > a$
4. $\int \frac{dx}{x\sqrt{a^2-x^2}} = -\frac{1}{a} \operatorname{sech}^{-1} \frac{x}{a} + C$, for $0 < x < a$
5. $\int \frac{dx}{x\sqrt{a^2+x^2}} = -\frac{1}{a} \operatorname{csch}^{-1} \frac{|x|}{a} + C$, for $x \neq 0$

8 Integration Techniques

8.1 Basic Approaches

$$\int k \, dx = kx + C \quad (29)$$

$$\int k^p \, dx = \frac{k^{p+1}}{p+1} + C, p \in \mathbf{R} \wedge \neq -1 \quad (30)$$

$$\int \cos ax \, dx = \frac{1}{a} \sin ax + C \quad (31)$$

$$\int \sin ax \, dx = -\frac{1}{a} \cos ax + C \quad (32)$$

$$\int \sec^2 ax \, dx = \frac{1}{a} \tan ax + C \quad (33)$$

$$\int \csc^2 ax \, dx = -\frac{1}{a} \cot ax + C \quad (34)$$

$$\int \sec ax \tan ax \, dx = \frac{1}{a} \sec ax + C \quad (35)$$

$$\int \csc ax \cot ax \, dx = -\frac{1}{a} \csc ax \cot ax + C \quad (36)$$

$$\int e^{ax} \, dx = \frac{1}{a} e^{ax} + C \quad (37)$$

$$\int \frac{dx}{x} = \ln |x| + C \quad (38)$$

$$\int \frac{dx}{\sqrt{a^2 - x^2}} = \sin^{-1} \frac{x}{a} + C \quad (39)$$

$$\int \frac{dx}{a^2 + x^2} = \frac{1}{a} \tan^{-1} \frac{x}{a} + C \quad (40)$$

$$\int \frac{dx}{x\sqrt{x^2 - a^2}} = \frac{1}{a} \sec^{-1} \left| \frac{x}{a} \right| + C \quad (41)$$

8.2 Integration By Parts

Suppose that u and v are differentiable functions. Then

$$\int u \, dv = uv - \int v \, du \quad (42)$$

Integration by Parts for Definite Integrals

Let u and v be differentiable. Then

$$\int_a^b u(x)v'(x) \, dx = u(x)v(x) \Big|_a^b - \int_a^b v(x)u'(x) \, dx \quad (43)$$

Integral of $\ln x$

$$\int \ln x \, dx = x \ln x - x + C \quad (44)$$

8.3 Trigonometric Integrals

$\int \sin^m x \cos^n x \, dx$ **Strategy**

m is odd, n real Split off $\sin x$, rewrite the resulting even power of $\sin x$ in terms of $\cos x$, and then use $u = \cos x$

n odd, m real Split off $\cos x$, rewrite the resulting even power of $\cos x$ in terms of $\sin x$, and then use $u = \sin x$.

m and n both even, nonnegative Use half-angle identities to transform the integrand into a polynomial in $\cos 2x$, and apply the preceding strategies once again to powers of $\cos 2x$ greater than 1.

Reduction Formulas

Assume n is a positive integer.

$$\int \sin^n x \, dx = -\frac{\sin^{n-1} x \cos x}{n} + \frac{n-1}{n} \int \sin^{n-2} x \, dx \quad (45)$$

$$\int \cos^n x \, dx = \frac{\cos^{n-1} x \sin x}{n} + \frac{n-1}{n} \int \cos^{n-2} x \, dx \quad (46)$$

$$\int \tan^n x \, dx = \frac{\tan^{n-1} x}{n-1} - \int \tan^{n-2} x \, dx, n \neq 1 \quad (47)$$

$$\int \sec^n x \, dx = \frac{\sec^{n-2} x \tan x}{n-1} + \frac{n-2}{n-1} \int \sec^{n-2} x \, dx, n \neq 1 \quad (48)$$

Integrals of $\tan x$, $\cot x$, $\sec x$, and $\csc x$

$$\int \tan x \, dx = -\ln |\cos x| + C = \ln |\sec x| + C \quad (49)$$

$$\int \cot x \, dx = \ln |\sin x| + C \quad (50)$$

$$\int \sec x \, dx = \ln |\sec x + \tan x| + C \quad (51)$$

$$\int \csc x \, dx = -\ln |\csc x + \cot x| + C \quad (52)$$

$\int \tan^m x \sec^n x \, dx$ **Strategy**

n **even** Split off $\sec^2 x$, rewrite the remaining even power of $\sec x$ in terms of $\tan x$, and use $u = \tan x$.

m **odd** Split off $\sec x \tan x$, rewrite the remaining even power of $\tan x$ in terms of $\sec x$, and use $u = \sec x$.

m **even and** n **odd** Rewrite the even power of $\tan x$ in terms of $\sec x$ to produce a polynomial in $\sec x$; apply reduction formula 4 to each term.

8.4 Trigonometric Substitutions

The Integral Contains...	Corresponding Substitution	Useful Identity
$a^2 - x^2$	$x = a \sin \theta, -\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}, \forall x \leq a$	$a^2 - a^2 \sin^2 \theta = a^2 \cos^2 \theta$
$a^2 + x^2$	$x = a \tan \theta, -\frac{\pi}{2} < \theta < \frac{\pi}{2}$	$a^2 + a^2 \tan^2 \theta = a^2 \sec^2 \theta$
$x^2 - a^2$	$x = a \sec \theta,$	$a^2 \sec^2 \theta - a^2 = a^2 \tan^2 \theta$

$$\begin{cases} 0 \leq \theta < \frac{\pi}{2}, \forall x \geq a \\ \frac{\pi}{2} < \theta \leq \pi, \forall x \leq -a \end{cases}$$

8.5 Partial Fractions

Partial Fractions with Simple Linear Factors

Suppose $f(x) = p(x) > q(x)$, where p and q are polynomials with no common factors and with the degree of p less than the degree of q . Assume that q is the product of simple linear factors. The partial fraction decomposition is obtained as follows.

Step 1. Factor the denominator q in the form $(x - r_1)(x - r_2) \cdots (x - r_n)$, where $r_1, \dots, r_n$ are real numbers.

Step 2. Partial fraction decomposition Form the partial fraction decomposition by writing

$$\frac{p(x)}{q(x)} = \frac{A_1}{x - r_1} + \frac{A_2}{x - r_2} + \cdots + \frac{A_n}{x - r_n} \quad (53)$$

Step 3. Clear denominators Multiply both sides of the equation in Step 2 by $q(x) = (x - r_1)(x - r_2) \cdots (x - r_n)$, which produces conditions for $A_1, \dots, A_n$.

Step 4. Solve for coefficients Equate like powers of x in Step 3 to solve for the undetermined coefficients $A_1, \dots, A_n$.

Partial Fractions For Repeated Linear Factors

Suppose the repeated linear factor $(x - r)^m$ appears in the denominator of a proper rational function in reduced form. The partial fraction decomposition has a partial fraction for each power of $(x - r)$ up to and including the m th power; that is, the partial fraction decomposition contains the sum

$$\frac{A_1}{(x - r)} + \frac{A_2}{(x - r)^2} + \frac{A_3}{(x - r)^3} + \cdots + \frac{A_m}{(x - r)^m} \quad (54)$$

where $A_1, \dots, A_m$ are constants to be determined.

Partial Fractions with Simple Irreducible Quadratic Factors

Suppose a simple irreducible factor $ax^2 + bx + c$ appears in the denominator of a proper rational function in reduced form. The partial fraction decomposition contains a term of the form

$$\frac{Ax + B}{ax^2 + bx + c} \quad (55)$$

where A and B are unknown coefficients to be determined.

Partial Fraction Decomposition

Let $f(x) = p(x)/q(x)$ be a proper rational function in reduced form. Assume the denominator q has been factored completely over the real numbers and m is a positive integer.

Simple linear factor A factor $x - r$ in the denominator requires the partial fraction $\frac{A}{x - r}$.

Repeated linear factor A factor $(x - r)^m$ with $m > 1$ in the denominator requires the partial fractions.

$$\frac{A_1}{(x - r)} + \frac{A_2}{(x - r)^2} + \frac{A_3}{(x - r)^3} + \cdots + \frac{A_m}{(x - r)^m} \quad (56)$$

Simple irreducible quadratic factor An irreducible factor $ax^2 + bx + c$ in the denominator requires the partial fraction

$$\frac{Ax + B}{ax^2 + bx + c} \quad (57)$$

Repeated irreducible quadratic factor An irreducible factor $(ax^2 + bx + c)^m$ with $m > 1$ in the denominator requires the partial fractions

$$\frac{A_1x + B_1}{ax^2 + bx + c} + \frac{A_2x + B_2}{(ax^2 + bx + c)^2} + \cdots + \frac{A_mx + B_m}{(ax^2 + bx + c)^m} \quad (58)$$

8.6 Improper Integrals

Improper Integrals over Infinite Intervals

1. If f is continuous on $[a, \infty)$, then

$$\int_a^\infty f(x) dx = \lim_{b \rightarrow \infty} \int_a^b f(x) dx \quad (59)$$

provided the limit exists.

2. If f is continuous on $(-\infty, b]$, then

$$\int_{-\infty}^b f(x) dx = \lim_{a \rightarrow -\infty} \int_a^b f(x) dx \quad (60)$$

provided the limit exists.

3. If f is continuous on $(-\infty, \infty)$, then

$$\int_{-\infty}^\infty f(x) dx = \lim_{a \rightarrow -\infty} \int_a^c f(x) dx + \lim_{b \rightarrow \infty} \int_c^b f(x) dx \quad (61)$$

provided both limits exist, where c is any real number.

In each case, if the limit exists, the improper integral is said to **converge**, if it does not exist, the improper integral is said to **diverge**.

Improper Integrals with an Unbounded Integrand

1. Suppose f is continuous on $(a, b]$ with $\lim_{x \rightarrow a^+} f(x) = \pm \infty$. Then

$$\int_a^b f(x) dx = \lim_{c \rightarrow a^+} \int_c^b f(x) dx \quad (62)$$

provided the limit exists.

2. Suppose f is continuous on $[a, b)$ with $\lim_{x \rightarrow b^-} f(x) = \pm \infty$. Then

$$\int_a^b f(x) dx = \lim_{c \rightarrow b^-} \int_a^c f(x) dx \quad (63)$$

provided the limit exists.

3. Suppose f is continuous on $[a, b]$ except at the interior point p where f is unbounded. Then

$$\int_a^b f(x) dx = \int_a^p f(x) dx + \int_p^b f(x) dx, \quad (64)$$

provided the improper integrals on the right side exist.

In each case, if the limit exists, the improper integrals is said to **converge**, if it does not exists, the improper integral is said to **diverge**.

9 Sequences and Infinite Series

9.1 An Overview

Sequence

A **sequence** $\{a_n\}$ is an ordered list of numbers of the form

$$\{a_1, a_2, a_3, \dots, a_n, \dots\} \quad (65)$$

A sequence may be generated by a **recurrence relations** of the form $a_{n+1} = f(a_n)$, for $n = 1, 2, 3, \dots$, where a_1 is given. A sequence may also be defined with an **explicit form** of the form $a_n = f(n)$, for $n = 1, 2, 3, \dots$

Limit of a Sequence

If the terms of a sequence $\{a_n\}$ approach a unique number L as n increases, then we say $\lim_{n \rightarrow \infty} a_n = L$ exists, and the sequence **converges** to L . If the terms of the sequence do not approach a single number as n increases, the sequence has no limits, and the sequence **diverges**.

Infinite Series

Given a set of numbers $\{a_1, a_2, a_3, \dots\}$, the sum

$$a_1 + a_2 + a_3 + \dots = \sum_{k=1}^{\infty} a_k \quad (66)$$

is called an **infinity series**. Its **sequence of partial sums** $\{S_n\}$ has the terms

$$S_1 = a_1 \quad (67)$$

$$S_2 = a_1 + a_2 \quad (68)$$

$$S_3 = a_1 + a_2 + a_3 \quad (69)$$

$$\vdots \quad (70)$$

$$S_n = a_1 + a_2 + a_3 + \dots + a_n = \sum_{k=1}^n a_k, \quad n = 1, 2, 3, \dots \quad (71)$$

If the sequence of partial sums $\{S_n\}$ has a limit L , the infinite series **converges** to that limits, and we write

$$\sum_{k=1}^{\infty} a_k = \lim_{n \rightarrow \infty} \sum_{k=1}^n a_k = \lim_{n \rightarrow \infty} S_n = L \quad (72)$$

If the sequence of partial sums diverges, the infinite series also **diverges**.

9.2 Sequences

Limits of Sequences from Limits of Functions

Suppose f is a function such that $f(n) = a_n$ for all positive integers n . If $\lim_{n \rightarrow \infty} f(n) = L$, then the limits of the sequences $\{a_n\}$ is also L .

Properties of Limits of Sequences

Assume that the sequence $\{a_n\}$ and $\{b_n\}$ have limits A and B , respectively. Then,

1. $\lim_{n \rightarrow \infty} (a_n \pm b_n) = A \pm B$
2. $\lim_{n \rightarrow \infty} ca_n = cA$, where c is a real number
3. $\lim_{n \rightarrow \infty} a_nb_n = AB$
4. $\lim_{n \rightarrow \infty} \frac{a_n}{b_n} = \frac{A}{B}$, provided $B \neq 0$.

Geometric Sequences

Let r be a real number. Then,

$$\lim_{n \rightarrow \infty} r^n = \begin{cases} 0 & \Leftrightarrow |r| < 1 \\ 1 & \Leftrightarrow r = 1 \\ \text{does not exist} & \Leftrightarrow r \leq -1 \vee r > 1 \end{cases} \quad (73)$$

Squeeze Theorem for Sequences

Let $\{a_n\}$, $\{b_n\}$, and $\{c_n\}$ be sequences with $a_n \leq b_n \leq c_n$ for all n greater than some index N . If $\lim_{n \rightarrow \infty} a_n = \lim_{n \rightarrow \infty} c_n = L$, then $\lim_{n \rightarrow \infty} b_n = L$.

Bounded Monotonic Sequences

A bounded monotonic sequence converges.

Growth Rates of Sequences

The following sequences are ordered according to increasing growth rates as $n \rightarrow \infty$; that is, if $\{a_n\}$ appears before $\{b_n\}$ in the list, then $\lim_{n \rightarrow \infty} \frac{a_n}{b_n} = 0$ and $\lim_{n \rightarrow \infty} \frac{b_n}{a_n} = \infty$

$$\{\ln^q n\} \ll \{n^p\} \ll \{n^p \ln^r n\} \ll \{n^{p+s}\} \ll \{b^n\} \ll \{n!\} \ll \{n^n\} \quad (74)$$

The ordering applies for $p, q, r, s, b \in \mathbb{R}^+ \wedge b > 1$.

Limit of a Sequence

The sequence $\{a_n\}$ converges to L provided the terms of a_n can be made arbitrarily close to L by taking n sufficiently large. More precisely, $\{a_n\}$ has the unique limit L if given any tolerance $\epsilon > 0$, it is possible to find a positive integer N (depending only on ϵ) such that

$$|a_n - L| < \epsilon \quad \text{whenever } n > N \quad (75)$$

if the **limit of a sequence** is L , we say the sequence **converges** to L , written

$$\lim_{n \rightarrow \infty} a_n = L \quad (76)$$

A sequence that does not converge is said to **diverge**.

9.3 Infinite Series

Geometric Series

$$\sum_{k=0}^{n-1} ar^k = S_n = a \frac{1 - r^n}{1 - r} \quad (77)$$

Geometric Series

Let $a \neq 0$ and r be real numbers. If $|r| < 1$, then $\sum_{k=0}^{\infty} ar^k = \frac{a}{1-r}$. If $|r| \geq 1$, then the series diverges.

9.4 The Divergence and Integral Tests

Divergence Test

If $\sum a_k$ converges, then $\lim_{n \rightarrow \infty} a_k = 0$. Equivalently, if $\lim_{n \rightarrow \infty} a_k \neq 0$, then the series diverges. However, this cannot be used to prove convergence. If $\lim_{n \rightarrow \infty} a_k = 0$, the test is inconclusive.

Harmonic Series

The harmonic series $\sum_{k=1}^{\infty} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \cdots$ diverges, even though the terms of the series approach zero.

Integral Test

Suppose f is a continuous, positive, decreasing function, for $x \geq 1$, and let $a_k = f(k)$, for $k = 1, 2, 3, \dots$. Then

$$\sum_{k=1}^{\infty} a_k \quad \text{and} \quad \int_1^{\infty} f(x) dx \quad (78)$$

either both converge or both diverge. In the case of convergence, the value of the integral is *not*, in general, equal to the value of the series.

Convergence of the p -Series

The p -Series $\sum_{k=1}^{\infty} \frac{1}{k^p}$ converges, for $p > 1$, and diverges for $p \leq 1$.

Estimating Series with Positive Terms

Let f be continuous, positive, decreasing function, for $x \geq 1$, and let $a_k = f(k)$, for $k = 1, 2, 3, \dots$. Let $S = \sum_{k=1}^{\infty} a_k$ be a convergence series and let $S_n = \sum_{k=1}^n a_k$ be the sum of the first n terms of the series. The remainder $R_n = S - S_n$ satisfies

$$R_n \leq \int_n^{\infty} f(x) dx \quad (79)$$

Furthermore, the exact value of the series is bounded as follows:

$$S_n + \int_{n+1}^{\infty} f(x) dx \leq \sum_{k=1}^{\infty} a_k \leq S_n + \int_n^{\infty} f(x) dx. \quad (80)$$

Properties of Convergent Series

1. Suppose $\sum a_k$ converges to A and let c be a real number. The series $\sum ca_k$ converges and $\sum ca_k = c \sum a_k = cA$
2. Suppose $\sum a_k$ converges to A and $\sum b_k$ converges to B . The series $\sum(a_k \pm b_k)$ converges and $\sum(a_k \pm b_k) = \sum a_k \pm \sum b_k = A \pm B$
3. *Whether* a series converges does not depend on a finite number of terms added to or removed from the series. Specifically, if M is a positive integer, then $\sum_{k=1}^{\infty} a_k$ and $\sum_{k=M}^{\infty} a_k$ both converge or both diverge. However, the *value* of a convergent series does change if nonzero terms are added or deleted.

9.5 The Ratio, Root, and Comparison Tests

Useful Identities

$$\lim_{k \rightarrow \infty} \left(1 + \frac{1}{k}\right)^k = \lim_{k \rightarrow \infty} \left(\frac{k+1}{k}\right)^k = e \quad (81)$$

$$\lim_{k \rightarrow \infty} k^{\frac{1}{k}} = 1 \quad (82)$$

The Ratio Test

Let $\sum a_k$ be an infinite series with positive terms and let $r = \lim_{n \rightarrow \infty} \frac{a_{k+1}}{a_k}$.

1. If $0 \leq r < 1$, the series converges.
2. If $r > 1$ (including $r = \infty$), the series diverges.
3. If $r = 1$, the test is inconclusive.

The Root Test

Let $\sum a_k$ be an infinite series with nonnegative terms and let $\rho = \lim_{n \rightarrow \infty} \sqrt[k]{a_k}$.

1. If $0 \leq \rho < 1$, the series converges.
2. If $\rho > 1$ (including $\rho = \infty$), the series diverges.
3. If $\rho = 1$, test is inconclusive.

The Comparison Test

Let $\sum a_k$ and $\sum b_k$ be a series with positive terms.

1. If $0 < a_k \leq b_k$ and $\sum b_k$ converges, then $\sum a_k$ converges.
2. If $0 < b_k \leq a_k$ and $\sum b_k$ diverges, then $\sum a_k$ diverges.

The Limit Comparison Test

Let $\sum a_k$ and $\sum b_k$ be series with positive terms and let

$$\lim_{k \rightarrow \infty} \frac{a_k}{b_k} = L \quad (83)$$

- If $0 < L < \infty$ (that is, L is a finite, positive number), then $\sum a_k$ and $\sum b_k$ either both converge or both diverge.
- If $L = 0$ and $\sum b_k$ converges, then $\sum a_k$ converges.
- If $L = \infty$ and $\sum b_k$ diverges, then $\sum a_k$ diverges.

Guidelines

- Begin with the Divergence Test. If you show that $\lim_{n \rightarrow \infty} a_n \neq 0$, then the series diverges and your work is finished.
- Is the series a special series? Recall the convergence properties for the following series:
 - Geometric series: $\sum ar^k$ converges for $|r| < 1$ and diverges for $|r| \geq 1$ ($a \neq 0$).
 - p -series: $\sum \frac{1}{k^p}$ converges for $p > 1$, and diverges for $p \leq 1$.
 - Check also for telescoping series.
- If the general k th term of the series looks like a function you can integrate, then try the Integral Test.
- If the general k th term of the series involves $k!$, k^k , or a^k , where a is a constant, the Ratio Test is advisable. Series with k in an exponent may yield to the Root Test.
- If the general k th term of the series is a rational function of k (or a root of a rational function), use the Comparison or the Limit Comparison Test. Use the families of series given in Step 2 as comparison series.

9.6 Alternating Series

The Alternating Series Test

The alternating series $\sum (-1)^{k+1} a_k$ converges provided

1. the terms of the series are non-increasing in magnitude ($0 < a_{k+1} \leq a_k$, for k greater than some index N) and
2. $\lim_{n \rightarrow \infty} a_n = 0$

Alternating Harmonic Series

The alternating harmonic series $\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \dots$ converges (even though the harmonic series $\sum_{k=1}^{\infty} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \dots$ diverges).

Remainder in Alternating Series

Let $R_n = |S - S_n|$ be the remainder in approximating the value of a convergent alternating series $\sum_{k=1}^{\infty} (-1)^{k+1} a_k$ by the sum of its first n terms. Then $R_n \leq a_{n+1}$. In other words, the remainder is less than or equal to the magnitude of the first neglected term.

Absolute and Conditional Convergence

Assume the infinite series $\sum a_k$ converges. The series $\sum a_k$ **converges absolutely** if the series $\sum |a_k|$ converges. Otherwise, the series $\sum a_k$ **converges conditionally**.

Absolute Convergence Implies Convergence

If $\sum |a_k|$ converges, then $\sum a_k$ converges (absolute convergence implies convergence). If $\sum a_k$ diverges, then $\sum |a_k|$ diverges.

Series or Test	Form of Series	Condition for Convergence	Condition for Divergence	Comments
Geometric Series	$\sum_{k=0}^{\infty} ar^k, a \neq 0$	$ r < 1$	$ r \geq 1$	If $ r < 1$, then $\sum_{k=1}^{\infty} ar^k = \frac{a}{1-r}$
Divergence Test	$\sum_{k=1}^{\infty} a_k$	Does Not Apply	$\lim_{k \rightarrow \infty} a_k \neq 0$	Cannot be used to prove convergence.
Integral Test	$\sum_{k=1}^{\infty} a_k$, where $a_k = f(k)$ and f is continuous, positive, and decreasing.	$\int_1^{\infty} f(x)dx < \infty$	$\int_1^{\infty} f(x)dx$ does not exist	The value of the integral is not the value of the series.
p -Series	$\sum_{k=1}^{\infty} \frac{1}{k^p}$	$p > 1$	$p \leq 1$	Useful for comparison tests.
Ratio Test	$\sum_{k=1}^{\infty} a_k$, where $a_k > 0$	$\lim_{k \rightarrow \infty} \frac{a_{k+1}}{a_k} < 1$	$\lim_{k \rightarrow \infty} \frac{a_{k+1}}{a_k} > 1$	Inconclusive if $\lim_{k \rightarrow \infty} \frac{a_{k+1}}{a_k} = 1$
Root Test	$\sum_{k=1}^{\infty} a_k$, where $a_k \geq 0$	$\lim_{k \rightarrow \infty} \sqrt[k]{a_k} < 1$	$\lim_{k \rightarrow \infty} \sqrt[k]{a_k} > 1$	Inconclusive if $\lim_{k \rightarrow \infty} \sqrt[k]{a_k} = 1$
Comparison Test	$\sum_{k=1}^{\infty} a_k$, where $a_k > 0$	$0 < a_k \leq b_k$ and $\sum_{k=1}^{\infty} b_k$ converges	$0 < b_k \leq a_k$ and $\sum_{k=1}^{\infty} b_k$ diverges	$\sum_{k=1}^{\infty} a_k$ is given, you supply $\sum_{k=1}^{\infty} b_k$
Limit Comparison Test	$\sum_{k=1}^{\infty} a_k$, where $a_k > 0, b_k > 0$	$0 < \lim_{k \rightarrow \infty} \frac{a_k}{b_k} < \infty$ and $\sum_{k=1}^{\infty} b_k$ converges	$\lim_{k \rightarrow \infty} \frac{a_k}{b_k} = 0$ and $\sum_{k=1}^{\infty} b_k$ diverges	$\sum_{k=1}^{\infty} a_k$ is given, you supply $\sum_{k=1}^{\infty} b_k$
Alternating Series Test	$\sum_{k=1}^{\infty} (-1)^k a_k$, where $a_k > 0, 0 < a_{k+1} \leq a_k$	$\lim_{k \rightarrow \infty} a_k = 0$	$\lim_{k \rightarrow \infty} a_k \neq 0$	Remainder R_n satisfies $R_n \leq a_{n+1}$
Absolute Convergence	$\sum_{k=1}^{\infty} a_k, a_k$ arbitrary	$\sum_{k=1}^{\infty} a_k $ converges.	Applies to arbitrary series	

10 Power Series

10.1 Approximating Functions with Polynomials

Taylor Polynomials

Let f be a function with $f', f'', \dots, f^{(n)}$ defined at a . The **n th-order Taylor polynomial** for f with its **center** at a , denoted p_n , has the property that it matches f in value, slope, and all derivatives up to the n th derivative at a ; that is,

$$p_n(a) = f(a), p'_n(a) = f'(a), \dots, p_n^{(n)}(a) = f^{(n)}(a). \quad (84)$$

The n th-order Taylor polynomial centered at a is

$$p_n(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \dots + \frac{f^{(n)}(a)}{n!}(x-a)^n. \quad (85)$$

More compactly, $p_n(x) = \sum_{k=0}^n c_k(x-a)^k$, where the **coefficients** are

$$c_k = \frac{f^{(k)}(a)}{k!}, \quad \text{for } k = 0, 1, 2, \dots, n \quad (86)$$

Remainder in a Taylor Polynomial

Let p_n be the Taylor polynomial of order n for f . The **remainder** in using p_n to approximate f at the point x is

$$R_n(x) = f(x) - p_n(x) \quad (87)$$

Taylor's Theorem

Let f have continuous derivatives up to $f^{(n+1)}$ on an open interval I containing a . For all x in I ,

$$f(x) = p_n(x) + R_n(x), \quad (88)$$

where p_n is the n th-order Taylor polynomial for f centered at a , and the remainder is

$$R_n(x) = \frac{f^{(n+1)}(c)}{(n+1)!}(x-a)^{(n+1)}, \quad (89)$$

for some point c between x and a .

Estimate of the Remainder

Let n be a fixed positive integer. Suppose there exists a number M such that $|f^{(n+1)}(c)| \leq M$, for all c between a and x inclusive. The remainder in the n th-order Taylor polynomial for f centered at a satisfies

$$|R_n(x)| = |f(x) - p_n(x)| \leq M \frac{|x-a|^{n+1}}{(n+1)!} \quad (90)$$

10.2 Properties of Power Series

Power Series

A **power series** has the general form

$$\sum_{k=0}^{\infty} c_k(x-a)^k \quad (91)$$

where a and c_k are real numbers, and x is a variable. The c_k 's are the **coefficients** of the power series and a is the **center** of the power series. The set of values of x for which the series converges is its **interval of convergence**. The **radius of convergence** of the power series, denoted R , is the distance from the center of the series to the boundary of the interval of convergence.

Convergence of Power Series

A power series $\sum_{k=0}^{\infty} c_k(x-a)^k$ centered at a converges in one of three ways:

1. The series converges absolutely for all x , in which case the interval of convergence is $(-\infty, \infty)$ and the radius of convergence is $R = \infty$.
2. There is a real number $R > 0$ such that the series converges absolutely for $|x-a| < R$ and diverges for $|x-a| > R$, in which case the radius of convergence is R .
3. The series converges only at a , in which case the radius of convergence is $R = 0$.

Combining Power Series

Suppose the power series $\sum c_k x^k$ and $\sum d_k x^k$ converge absolutely to $f(x)$ and $g(x)$, respectively, on an interval I .

1. **Sum and difference:** The power series $\sum (c_k \pm d_k)x^k$ converges absolutely to $f(x) \pm g(x)$ on I .
2. **Multiplication by a power:** The power series $x^m \sum c_k x^k = \sum c_k x^{k+m}$ converges absolutely to $x^m f(x)$ on I , provided m is an integer such that $k+m \geq 0$ for all terms of the series.

3. **Composition:** If $h(x) = bx^m$, where m is a positive integer and b is a real number, the power series $\sum c_k(h(x))^k$ converges absolutely to the composite function $f(h(x))$, for all x such that $h(x)$ is in I .

Differentiating and Integrating Power Series

Let the function f be defined by the power series $\sum c_k(x - a)^k$ on its interval of convergence I .

- f is a continuous function on I .
- The power series may be differentiated or integrated term by term, and the resulting power series converges to $f'(x)$ or $\int f(x) dx + C$, respectively, at all points in the interior of I , where C is an arbitrary constant.

10.3 Taylor Series

Taylor/Maclaurin Series for a Function

Suppose the function f has derivatives of all orders on an interval containing the point a . The **Taylor series for f centered at a** is

$$f(a) + f'(a)(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \frac{f^{(3)}(a)}{3!}(x - a)^3 + \cdots \quad (92)$$

$$= \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x - a)^k. \quad (93)$$

A Taylor series centered at 0 is called a **Maclaurin series**.

Binomial Coefficients

$$\forall p, k \in \mathbb{R} \wedge k \geq 1$$

$$\binom{p}{k} = \frac{p(p-1)(p-2) \cdots (p-k+1)}{k!}. \quad (94)$$

With the special case of $\binom{p}{0} = 1$.

Binomial Series

$\forall p \in \mathbb{R} \wedge p \neq 0$, the Taylor series for $f(x) = (1+x)^p$ centered at 0 is the **binomial series**

$$\sum_{k=0}^{\infty} x^k = \sum_{k=0}^{\infty} \frac{p(p-1)(p-2) \cdots (p-k+1)}{k!} x^k \quad (95)$$

$$= 1 + px + \frac{p(p-1)(p-2)}{3!} x^3 + \cdots \quad (96)$$

The series converges for $|x| < 1$ (and possibly at the endpoints, depending on p). If p is a nonnegative integer, the series terminates and results in a polynomial of degree p .

Convergence of Taylor Series

Let f have derivatives of all orders on an open interval I containing a . The Taylor series for f centered at a converges to f , for all x in I , if and only if $\lim_{n \rightarrow \infty} R_n(x) = 0$, for all x in I , where

$$R_n(x) = \frac{f^{(n+1)}(c)}{(n+1)!} (x-a)^{n+1} \quad (97)$$

is the remainder at x (with c between x and a).

Taylor Series Functions

11 Parametric and Polar Curves

11.1 Parametric Equations

Forward or Positive Orientation

The direction in which a parametric curve is generated as the parameter increases is called the **forward**, or **positive, orientation** of the curve.

Derivative for Parametric Curves

Let $x = g(t)$ and $y = h(t)$, where g and h are differentiable on an interval $[a, b]$. Then

$$\frac{dy}{dx} = \frac{dy/dt}{dx/dt} = \frac{y'(t)}{x'(t)} \quad (98)$$

provided $\frac{dx}{dt} \neq 0$.

11.2 Polar Coordinates

Introduction

- Up to now we have only studied in a Cartesian coordinate system.
 - A Cartesian coordinate system is just a plane described by Cartesian (or, algebraic) equations and points in a finite dimensions.
 - * *One Dimension*: Lines.
 - * *Two Dimensions*: x^2 .
 - * *Three, Four*: Upper-level Calculus and Physics.
- Let's define an alternative coordinate system — **polar coordinate**.
 - coordinates are constants on circles and rays.
 - Useful for navigation, position, and gravitation fields.

Defining Polar Coordinates

Pole The origin of the coordinate system.

Polar Axis Synonymous for the positive x -axis.

Polar Coordinates A polar coordinates P has the form (r, θ) .

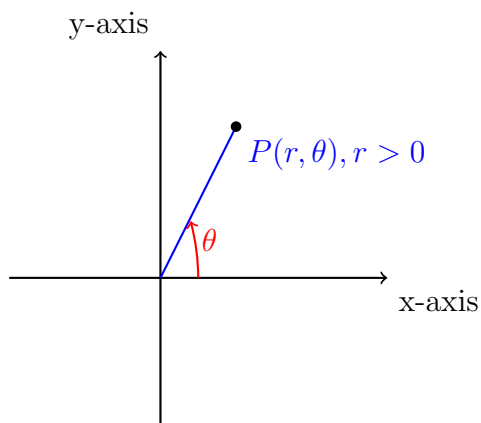
Radial Coordinate The radial coordinate r describes the *signed*, or *directed*, distance from the origin to P .

Angular Coordinate The angular coordinate θ describes an angle whose initial side is the positive x -axis and whose terminal side lies on the ray passing through the origin and P .

Notes

- **Positive angle measurements are measured *counterclockwise* from the origin.**
- Every point has multiple representations.
 - Angles are periodic, so multiples of 2π gives the same angle.

- Coordinates may be negative. So (r, θ) can be represented as $(-r, \theta + \pi)$ and $(-r, \theta - \pi)$



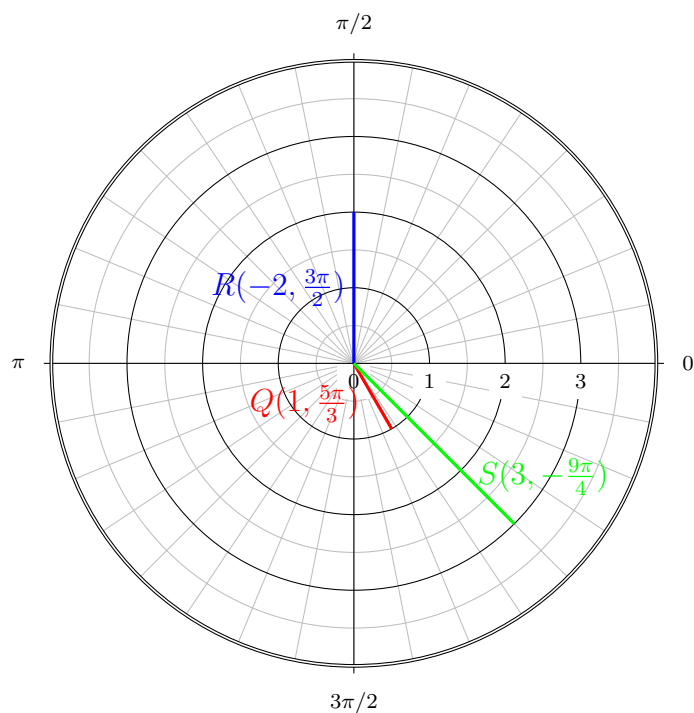
In summary,

- $(r, \theta + 2\pi)$ represents the same point as (r, θ)
- $P(r, \theta)$ and $P'(-r, \theta)$ are reflections through the origin.

Examples

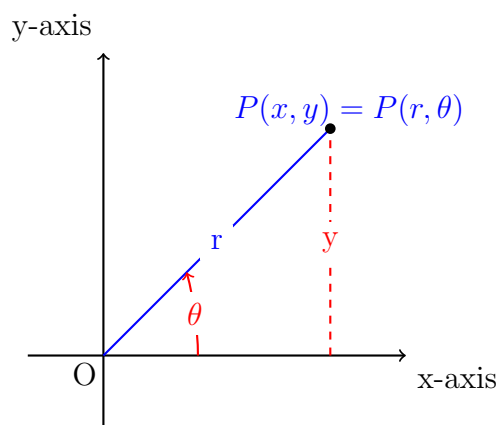
Graph the following points in polar coordinates:

- $Q(1, \frac{5\pi}{3})$
- $R(-2, \frac{3\pi}{2})$
- $S(3, -\frac{9\pi}{4})$
 - Now give two alternative representations.
 - $S'(3, \frac{1\pi}{4})$
 - $S''(-3, -\frac{5\pi}{4})$



Converting Between Cartesian and Polar Coordinates

- We sometimes need to convert between Cartesian and polar coordinates.
- Let's turn this problem into a right triangle.



A point with polar coordinates (r, θ) has Cartesian coordinates (x, y) , where

$$x = r \cos \theta \quad \text{and} \quad y = r \sin \theta \quad (99)$$

A point with Cartesian coordinates (x, y) has polar coordinates (r, θ) , where

$$r^2 = x^2 + y^2 \quad \text{and} \quad \tan \theta = \frac{y}{x} \quad (100)$$

Examples

BE SURE TO GRAPH POINTS IN CARTESIAN FIRST.

Polar to Cartesian

Express the point with the following polar coordinates in Cartesian coordinates: $P(3, \frac{2\pi}{3})$

$$\begin{aligned} x &= r \cos \theta & y &= r \sin \theta \\ &= 3 \cos(2\pi/3) & &= 3 \sin(2\pi/3) \\ &= -3(1/2) & &= 3(\sqrt{3}/2) \\ &= -3/2 & &= 3\sqrt{3}/2 \end{aligned}$$

Polar to Cartesian

Express the point with the following polar coordinates in Cartesian coordinates: $Q(e, -\frac{\pi}{4})$

$$\begin{aligned} x &= r \cos \theta & y &= r \sin \theta \\ &= e \cos(-\pi/4) & &= e \sin(-\pi/4) \\ &= e(\sqrt{2}/2) & &= -e(\sqrt{2}/2) \\ &= e\sqrt{2}/2 & &= -e\sqrt{2}/2 \end{aligned}$$

Cartesian To Polar

Express the point with the following Polar coordinates to Cartesian coordinates: $R(1, -1)$

$$\begin{array}{ll}
r &= \sqrt{x^2 + y^2} \\
&= \sqrt{1^2 + (-1)^2} \\
&= \sqrt{2} \\
\tan \theta &= y/x \\
&= -1/1 \\
&= -1 \\
\theta &= -\pi/4 \text{ or } 7\pi/4.
\end{array}$$

Therefore, two possible solutions are: $(\sqrt{2}, -\pi/4)$ or $(\sqrt{2}, 7\pi/4)$

Cartesian To Polar

Express the point with the following Polar coordinates to Cartesian coordinates: $S(1, \sqrt{3})$

$$\begin{array}{ll}
r &= \sqrt{x^2 + y^2} \\
&= \sqrt{(\sqrt{3})^2 + (1)^2} \\
&= 2 \\
\tan \theta &= y/x \\
&= \sqrt{3}/1 \\
\theta &= \pi/3 \text{ or } 4\pi/3.
\end{array}$$

Therefore, two possible solutions are: $(2, \pi/3)$ or $(2, 4\pi/3)$.

Basic Curves in Polar Coordinates

- A curve in polar coordinates is the set of **points** that satisfy an equation in r and θ .
- This makes graphing some things easier than others.
- Look at $r = 3$ is the set of all points that satisfy being away from the origin of 3 units.
 - This is because θ is not specified, it's arbitrary. Basically, θ is the function.
 - In general, $r = a, \forall a \in \mathbb{R}^+$ describes a circle.
- Taking the converse, let r be arbitrary.
 - If the r is arbitrary, and we specify the angle, what do you think we get?

- A line!
- Take $\sqrt{2}/2$.

Polar to Cartesian Graph Example

Convert the polar equation $r = 6 \sin \theta$ to Cartesian coordinates and describe the corresponding graph.

$$r^2 = 6r \sin \theta \quad (101)$$

$$x^2 + y^2 = 6y \quad (102)$$

$$0 = x^2 + y^2 - 6y \quad (103)$$

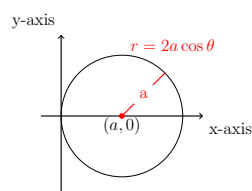
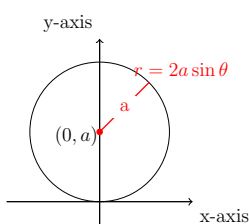
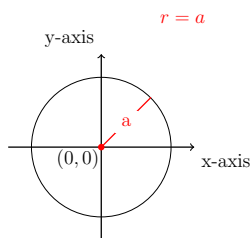
$$= x^2 + (y^2 - 6y + 9) - 9 \quad (104)$$

$$= x^2 + (y - 3)^2 - 9 \quad (105)$$

We recognize this to be the equation of a circle, centered at $(0, 3)$ at 3. We can also generalize this.

Circle in Polar Coordinates

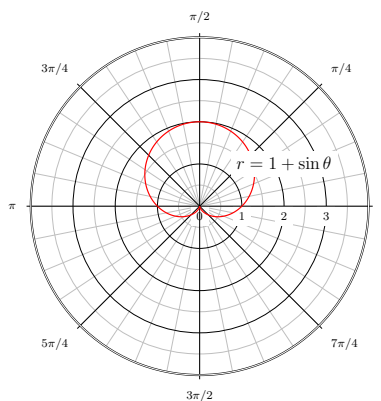
- The equation $r = a$ describes a circle of radius $|a|$ centered at $(0, 0)$.
- The equation $r = 2a \sin \theta$ describes a circle of radius $|a|$ centered at $(0, a)$.
- The equation $r = 2a \cos \theta$ describes a circle of radius $|a|$ centered at $(a, 0)$.



Graphing In Polar Coordinates

Graph the polar equation $r = f(\theta) = 1 + \sin \theta$

θ	$r = 1 + \sin\theta$
0	1
$\pi/6$	$3/2$
$\pi/2$	2
$5\pi/6$	$3/2$
π	1
$7\pi/6$	$1/2$
$3\pi/2$	0
$11\pi/6$	$1/2$
2π	1



The resulting curve is known as a **cardioid**.

Cartesian-to-Polar Method for Graphing $r = f(\theta)$

1. Graph $r = f(\theta)$ as if r and θ were Cartesian coordinates with θ on the horizontal axis and r on the vertical axis. Be sure to choose an interval in θ on which the entire polar curve is produced.
2. Use the Cartesian graph in Step 1 as a guide to sketch the points (r, θ) on the final *polar* curve.

Example

With the alternate graphing method, graph $r = 1 + \sin \theta$.

Symmetry In Polar Equations

Symmetry about the x-axis occurs if the point (r, θ) is on the graph whenever $(r, -\theta)$ is on the graph.

Symmetry about the y-axis occurs if the point (r, θ) is on the graph whenever $(r, \pi - \theta) = (-r, -\theta)$ is on the graph.

Symmetry about the origin occurs if the point (r, θ) is on the graph whenever $(-r, \theta) = (r, \theta + \pi)$ is on the graph.

11.3 Calculus in Polar Coordinates

Slope of a Tangent Line

Let f be a differentiable function at θ_0 . The slope of the line tangent to the curve $r = f(\theta)$ at the point $(f(\theta_0), \theta_0)$ is

$$\frac{dy}{dx} = \frac{f'(\theta_0) \sin \theta_0 + f(\theta_0) \cos \theta_0}{f'(\theta_0) \cos \theta_0 - f(\theta_0)} \quad (106)$$

Area of Regions in Polar Coordinates

Let R be the region bounded by the graphs of $r = f(\theta)$ and $r = g(\theta)$, between $\theta = \alpha$ and $\theta = \beta$, where f and g are continuous and $f(\theta) \geq g(\theta) \geq 0$ on $[\alpha, \beta]$. The area of R is

$$\int_{\alpha}^{\beta} \frac{1}{2} (f(\theta)^2 - g(\theta)^2) d\theta \quad (107)$$

MATH2222

Calculus III



Calculus III: Multivariable Calculus

Complete Notes

Illya Starikov

Missouri S&T

Contents

12 Vectors and Vector-Valued Functions	3
12.1 Vectors in the Plane	3
12.2 Vectors in Three Dimensions	5
12.3 Dot Product	6
12.4 Cross Product	8
12.5 Lines and Curves in Space	10
12.6 Calculus of Vector-Valued Functions	11
12.7 Motion In Space	13
12.8 Length of Curves	15
12.9 Curvature and Normal Vectors	16
13 Functions of Several Variables	19
13.1 Planes and Surfaces	19
13.2 Graphs and Level Curves	22
13.3 Limits and Continuity	23
13.4 Partial Derivatives	25
13.5 The Chain Rule	26
13.6 Directional Derivatives and the Gradient	27
13.7 Tangent Planes and Linear Approximation	29
13.8 Maximum/Minimum Problems	30
13.9 Lagrange Multipliers	32

14 Multiple Integration	33
14.1 Double Integrals over Rectangular Regions	33
14.2 Double Integrals over General Regions	34
14.3 Double Integrals in Polar Coordinates	35
14.4 Triple Integrals	36
14.5 Triple Integrals in Cylindrical and Spherical Coordinates . . .	37
14.6 Integrals for Mass Calculations	39
15 Vector Calculus	40
15.1 Vector Fields	40
15.2 Line Integrals	42
15.3 Conservative Vector Fields	45
15.4 Green's Theorem	47
15.5 Divergence and Curl	48
15.6 Surface Integrals	50
15.7 Stokes' Theorem	52
15.8 Divergence Theorem	53

12 Vectors and Vector-Valued Functions

12.1 Vectors in the Plane

Vectors, Equal Vectors, Scalars, Zero Vector

Vectors are quantities that have both **length** (or **magnitude**) and **direction**. Two vectors are **equal** if they have the same magnitude and direction. Quantities having magnitude but no direction are called **scalars**. One exception is the **zero** vector, denoted **0**: It has length 0 and no direction.

Scalar Multiples and Parallel Vectors

Given a scalar c and a vector $\mathbf{u}$, the scalar multiple $c\mathbf{v}$ is a vector whose magnitude is $|c|$ multiplied by the magnitude of $\mathbf{v}$. If $c > 0$, then $c\mathbf{v}$ has the same direction as $\mathbf{v}$. If $c < 0$, then $c\mathbf{v}$ and $\mathbf{v}$ point in opposite directions. Two vectors are **parallel** if they are scalar multiples of each other.

Position Vectors and Vector Components

A vector $\mathbf{v}$ with its tail at the origin and head at the point (v_1, v_2) is called a **position vector** (or is said to be in **standard position**) and is written $\langle v_1, v_2 \rangle$. The real numbers v_1 and v_2 are the **x-** and **y-components** of $\mathbf{v}$, respectively. The position vectors $\mathbf{u} = \langle u_1, u_2 \rangle$ and $\mathbf{v} = \langle v_1, v_2 \rangle$ are equal if and only if $u_1 = v_1$ and $u_2 = v_2$.

Magnitude of a Vector

Given the points $P(x_1, y_1)$ and $Q(x_2, y_2)$, the **magnitude**, or **length**, of $\vec{PQ} = \langle x_2 - x_1, y_2 - y_1 \rangle$, denoted $|\vec{PQ}|$, is the distance between P and Q :

$$|\vec{PQ}| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (1)$$

The magnitude of the position vector $\mathbf{v} = \langle v_1, v_2 \rangle$ is $|\mathbf{v}| = \sqrt{v_1^2 + v_2^2}$

Vector Operations

Suppose c is a scalar, $\mathbf{u} = \langle u_1, u_2 \rangle$ and $\mathbf{v} = \langle v_1, v_2 \rangle$.

$$\mathbf{u} + \mathbf{v} = \langle u_1 + v_1, u_2 + v_2 \rangle \quad \text{Vector addition} \quad (2)$$

$$\mathbf{u} - \mathbf{v} = \langle u_1 - v_1, u_2 - v_2 \rangle \quad \text{Vector subtraction} \quad (3)$$

$$c\mathbf{u} = \langle cu_1, cu_2 \rangle \quad \text{Scalar multiplication} \quad (4)$$

Unit Vectors and Vectors of a Specified Length

A **unit vector** is any vector with length 1. Given a nonzero vector $\mathbf{v}$, $\pm \frac{\mathbf{v}}{|\mathbf{v}|}$ are unit vectors parallel to $\mathbf{v}$. For a scalar $c > 0$, the vectors $\pm \frac{c\mathbf{v}}{|c\mathbf{v}|}$ are vectors of length c parallel to $\mathbf{v}$.

Properties of Vector Operations

Suppose $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ are vectors and a and c are scalars. Then the following properties hold (for vectors in any number of dimensions).

$$\mathbf{u} + a = \mathbf{v} + \mathbf{u} \quad \text{Commutative property of addition} \quad (5)$$

$$(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w}) \quad \text{Associative property of addition} \quad (6)$$

$$\mathbf{v} + \mathbf{0} = \mathbf{v} \quad \text{Additive identity} \quad (7)$$

$$\mathbf{v} + (-\mathbf{v}) = \mathbf{0} \quad \text{Additive identity} \quad (8)$$

$$c(\mathbf{u} + \mathbf{v}) = c\mathbf{u} + c\mathbf{v} \quad \text{Distributive property 1} \quad (9)$$

$$(a + c)\mathbf{v} = a\mathbf{v} + c\mathbf{v} \quad \text{Distributive property 2} \quad (10)$$

$$0\mathbf{v} = \mathbf{0} \quad \text{Multiplication by zero scalar} \quad (11)$$

$$c\mathbf{0} = \mathbf{0} \quad \text{Multiplication by zero vector} \quad (12)$$

$$1\mathbf{v} = \mathbf{v} \quad \text{Multiplicative identity} \quad (13)$$

$$a(c\mathbf{v}) = (ac)\mathbf{v} \quad \text{Associative property of scalar multiplication} \quad (14)$$

12.2 Vectors in Three Dimensions

Distance Formula in xyz -Space

The distance between the points $P(x_1, y_1, z_1)$ and $Q(x_2, y_2, z_2)$ is

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad (15)$$

Spheres and Balls

A **sphere** centered at (a, b, c) with radius r is the set of points satisfying the equation

$$(x - a)^2 + (y - b)^2 + (z - c)^2 = r^2 \quad (16)$$

A ball centered at (a, b, c) with radius r is the set of points satisfying the inequality

$$(x - a)^2 + (y - b)^2 + (z - c)^2 \leq r^2 \quad (17)$$

Vector Operations in $\mathbb{R}$

Let c be a scalar, $\mathbf{u} = \langle u_1, u_2, u_3 \rangle$ and $\mathbf{v} = \langle v_1, v_2, v_3 \rangle$.

$$\mathbf{u} + \mathbf{v} = \langle u_1 + v_1, u_2 + v_2, u_3 + v_3 \rangle \quad \text{Vector addition} \quad (18)$$

$$\mathbf{u} - \mathbf{v} = \langle u_1 - v_1, u_2 - v_2, u_3 - v_3 \rangle \quad \text{Vector subtraction} \quad (19)$$

$$c\mathbf{u} = \langle cu_1, cu_2, cu_3 \rangle \quad (20)$$

Magnitude of a Vector

The **magnitude** (or **length**) of the vector $\vec{PQ} = \langle x_2 - x_1, y_2 - y_1, z_2 - z_1 \rangle$ is the distance from $P(x_1, y_1, z_1)$ to $Q(x_2, y_2, z_2)$

$$|\vec{PQ}| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad (21)$$

12.3 Dot Product

Dot Product

Given two nonzero vectors $\mathbf{u}$ and $\mathbf{v}$ in two or three dimensions, their **dot product** is

$$\mathbf{u} \cdot \mathbf{v} = |\mathbf{u}| |\mathbf{v}| \cos \theta \quad (22)$$

where θ is the angle between $\mathbf{u}$ and $\mathbf{v}$ with $0 \leq \theta \leq \pi$. If $\mathbf{u} = \mathbf{0}$ or $\mathbf{v} = \mathbf{0}$, then $\mathbf{u} \cdot \mathbf{v} = 0$, and θ is undefined.

Orthogonal Vectors

Two vectors $\mathbf{u}$ and $\mathbf{v}$ are **orthogonal** if and only if $\mathbf{u} \cdot \mathbf{v} = 0$. The zero vector is orthogonal to all vectors. In two or three dimensions, two nonzero orthogonal vectors are perpendicular to each other.

Dot Product

Given two vectors $\mathbf{u} = \langle u_1, u_2, u_3 \rangle$ and $\mathbf{v} = \langle v_1, v_2, v_3 \rangle$,

$$\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + u_2 v_2 + u_3 v_3 \quad (23)$$

Properties of the Dot Product

Suppose $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ are vectors and let c be a scalar.

$$\mathbf{u} \cdot \mathbf{v} = \mathbf{v} \cdot \mathbf{u} \quad \text{Commutative property} \quad (24)$$

$$c(\mathbf{u} \cdot \mathbf{v}) = (c\mathbf{u}) \cdot \mathbf{v} = \mathbf{u} \cdot (c\mathbf{v}) \quad \text{Associative property} \quad (25)$$

$$\mathbf{u} \cdot (\mathbf{v} + \mathbf{w}) = \mathbf{u} \cdot \mathbf{v} + \mathbf{u} \cdot \mathbf{w} \quad (26)$$

(Orthogonal) Projection of $\mathbf{u}$ onto $\mathbf{v}$

The **orthogonal projection of $\mathbf{u}$ onto $\mathbf{v}$** , denoted $\text{proj}_{\mathbf{v}} \mathbf{u}$, where $\mathbf{v} \neq \mathbf{0}$, is

$$\text{proj}_{\mathbf{v}} \mathbf{u} = |\mathbf{u}| \cos \theta \left(\frac{\mathbf{v}}{|\mathbf{v}|} \right) \quad (27)$$

The orthogonal projection may also be computed with the formulas

$$\text{proj}_{\mathbf{v}} \mathbf{u} = \text{scal}_{\mathbf{v}} \mathbf{u} \left(\frac{\mathbf{v}}{|\mathbf{v}|} \right) = \left(\frac{\mathbf{u} \cdot \mathbf{v}}{\mathbf{v} \cdot \mathbf{v}} \right) \mathbf{v} \quad (28)$$

where the **scalar component of $\mathbf{u}$ in the direction of $\mathbf{v}$** is

$$\text{scal}_{\mathbf{v}} \mathbf{u} = |\mathbf{u}| \cos \theta = \frac{\mathbf{u} \cdot \mathbf{v}}{|\mathbf{v}|} \quad (29)$$

Work

Let a constant force $\mathbf{F}$ be applied to an object, producing a displacement $\mathbf{d}$. If the angle between $\mathbf{F}$ and $\mathbf{d}$ is θ , then the **work** done by the force is

$$W = |\mathbf{F}| |\mathbf{d}| \cos \theta = \mathbf{F} \cdot \mathbf{d} \quad (30)$$

12.4 Cross Product

Given two nonzero vectors $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{R}^3$, the **cross product** $\mathbf{u} \times \mathbf{v}$ is a vector with magnitude

$$|\mathbf{u} \times \mathbf{v}| = |\mathbf{u}||\mathbf{v}| \sin \theta \quad (31)$$

where $0 \leq \theta \leq \pi$ is the angle between $\mathbf{u}$ and $\mathbf{v}$. The direction of $\mathbf{u} \times \mathbf{v}$ is given by the **right-hand rule**: When you put the vectors tail to tail and let the fingers of your right hand curl from $\mathbf{u}$ to $\mathbf{v}$ the direction of $\mathbf{u} \times \mathbf{v}$ is the direction of your thumb, orthogonal to both $\mathbf{u}$ and $\mathbf{v}$. When $\mathbf{u} \times \mathbf{v} = 0$, the direction of $\mathbf{u} \times \mathbf{v}$ is undefined.

Geometry of the Cross Product

Let $\mathbf{u}$ and $\mathbf{v}$ be two nonzero vectors in $\mathbb{R}^3$.

1. The vectors $\mathbf{u}$ and $\mathbf{v}$ are parallel ($\theta = 0$ or $\theta = \pi$) if and only if $\mathbf{u} \times \mathbf{v} = 0$.
2. If $\mathbf{u}$ and $\mathbf{v}$ are two sides of a parallelogram, then the area of the parallelogram is

$$|\mathbf{u} \times \mathbf{v}| = |\mathbf{u}||\mathbf{v}| \sin \theta \quad (32)$$

Properties of the Cross Product

Let $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ be nonzero vectors in $\mathbb{R}^3$, and let a and b be scalars.

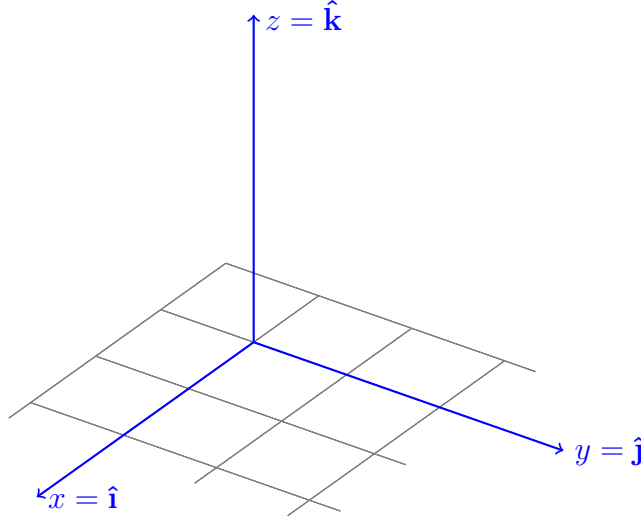
$$\mathbf{u} \times \mathbf{v} = -(\mathbf{v} \times \mathbf{u}) \quad \text{Anticommutative property} \quad (33)$$

$$(a\mathbf{u}) \times (b\mathbf{v}) = ab(\mathbf{u} \times \mathbf{v}) \quad \text{Associative property} \quad (34)$$

$$\mathbf{u} \times (\mathbf{v} + \mathbf{w}) = (\mathbf{u} \times \mathbf{v}) + (\mathbf{u} \times \mathbf{w}) \quad \text{Distributive property} \quad (35)$$

$$(\mathbf{u} + \mathbf{v}) \times \mathbf{w} = (\mathbf{u} \times \mathbf{w}) + (\mathbf{v} \times \mathbf{w}) \quad \text{Distributive property} \quad (36)$$

Cross Products of Coordinate Unit Vectors



$$\hat{\mathbf{i}} \times \hat{\mathbf{j}} = -(\hat{\mathbf{j}} \times \hat{\mathbf{i}}) = \hat{\mathbf{k}} \quad (37)$$

$$\hat{\mathbf{j}} \times \hat{\mathbf{k}} = -(\hat{\mathbf{k}} \times \hat{\mathbf{j}}) = \hat{\mathbf{i}} \quad (38)$$

$$\hat{\mathbf{k}} \times \hat{\mathbf{i}} = -(\hat{\mathbf{i}} \times \hat{\mathbf{k}}) = \hat{\mathbf{j}} \quad (39)$$

$$\hat{\mathbf{i}} \times \hat{\mathbf{i}} = \hat{\mathbf{j}} \times \hat{\mathbf{j}} = \hat{\mathbf{k}} \times \hat{\mathbf{k}} = 0 \quad (40)$$

Evaluating the Cross Product

Let $\mathbf{u} = u_1\hat{\mathbf{i}} + u_2\hat{\mathbf{j}} + u_3\hat{\mathbf{k}}$ and $\mathbf{v} = v_1\hat{\mathbf{i}} + v_2\hat{\mathbf{j}} + v_3\hat{\mathbf{k}}$. Then

$$\mathbf{u} \times \mathbf{v} = \begin{vmatrix} \hat{\mathbf{i}} & \hat{\mathbf{j}} & \hat{\mathbf{k}} \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{vmatrix} = \begin{vmatrix} u_2 & u_3 \\ v_2 & v_3 \end{vmatrix} \hat{\mathbf{i}} - \begin{vmatrix} u_1 & u_3 \\ v_1 & v_3 \end{vmatrix} \hat{\mathbf{j}} + \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix} \hat{\mathbf{k}} \quad (41)$$

12.5 Lines and Curves in Space

Equation of a Line

An equation of the line passing through the point $P_0(x_0, y_0, z_0)$ in the direction of the vector $\mathbf{v} = \langle a, b, c \rangle$ is $\mathbf{r} = \mathbf{r}_0 + t\mathbf{v}$, or

$$\langle x, y, z \rangle = \langle x_0, y_0, z_0 \rangle + t\langle a, b, c \rangle, \quad \text{for } -\infty < t < \infty \quad (42)$$

Equivalently, the parametric equations of the line are

$$x = x_0 + at, \quad y = y_0 + bt, \quad z = z_0 + ct, \quad \text{for } -\infty < t < \infty \quad (43)$$

Limit of a Vector-Valued Function

A vector-valued function $\mathbf{r}$ approaches the limit $\mathbf{L}$ as t approaches a , written $\lim_{t \rightarrow a} \mathbf{r}(t) = \mathbf{L}$, provided $\lim_{t \rightarrow a} |\mathbf{r}(t) - \mathbf{L}| = 0$

12.6 Calculus of Vector-Valued Functions

Derivative and Tangent Vector

Let $\mathbf{r}(t) = f(t)\hat{\mathbf{i}} + g(t)\hat{\mathbf{j}} + h(t)\hat{\mathbf{k}}$, where f , g , and h are differentiable functions on (a, b) . Then $\mathbf{r}$ has a **derivative** (or is **differentiable**) on (a, b) and

$$\mathbf{r}'(t) = f'(t)\hat{\mathbf{i}} + g'(t)\hat{\mathbf{j}} + h'(t)\hat{\mathbf{k}} \quad (44)$$

Provided $\mathbf{r}'(t) \neq \mathbf{0}$, $\mathbf{r}'(t)$ is a **tangent vector** (or velocity vector) at the point corresponding to $\mathbf{r}$.

Unit Tangent Vector

Let $\mathbf{r} = f(t)\hat{\mathbf{i}} + g(t)\hat{\mathbf{j}} + h(t)\hat{\mathbf{k}}$ be a smooth parameterized curve, for $a \leq t \leq b$. The **unit tangent vector** for a particular value of t is

$$\mathbf{T}(t) = \frac{\mathbf{r}'(t)}{|\mathbf{r}'(t)|} \quad (45)$$

Derivative Rules

Let $\mathbf{u}$ and $\mathbf{v}$ be differentiable vector-valued functions and let f be a differentiable scalar-valued function, all at a point t . Let $\mathbf{c}$ be a constant vector. The following rules apply.

$$\frac{d}{dt}(\mathbf{c}) = \mathbf{0} \quad \text{Constant Rule} \quad (46)$$

$$\frac{d}{dt}(\mathbf{u}(t) + \mathbf{v}(t)) = \mathbf{u}'(t) + \mathbf{v}'(t) \quad \text{Sum Rule} \quad (47)$$

$$\frac{d}{dt}(f(t)\mathbf{u}(t)) = f'(t)\mathbf{u}(t) + f(t)\mathbf{u}'(t) \quad \text{Product Rule} \quad (48)$$

$$\frac{d}{dt}(\mathbf{u}(f(t))) = \mathbf{u}'(f(t))f'(t) \quad \text{Chain Rule} \quad (49)$$

$$\frac{d}{dt}(\mathbf{u}(t) \cdot \mathbf{v}(t)) = \mathbf{u}'(t) \cdot \mathbf{v}(t) + \mathbf{u}(t) \cdot \mathbf{v}'(t) \quad \text{Dot Product Rule} \quad (50)$$

$$\frac{d}{dt}(\mathbf{u}(t) \times \mathbf{v}(t)) = \mathbf{u}'(t) \times \mathbf{v}(t) + \mathbf{u}(t) \times \mathbf{v}'(t) \quad \text{Cross Product Rule} \quad (51)$$

Indefinite Integral of a Vector-Valued Function

Let $\mathbf{r} = f\hat{\mathbf{i}} + g\hat{\mathbf{j}} + h\hat{\mathbf{k}}$ be a vector function and let $\mathbf{R} = F\hat{\mathbf{i}} + G\hat{\mathbf{j}} + H\hat{\mathbf{k}}$, where F , G , and H are antiderivatives of f , g , and h , respectively. The **indefinite integral** of $\mathbf{r}$ is

$$\int \mathbf{r}(t) dt = \mathbf{R}(t) + \mathbf{C} \quad (52)$$

where $\mathbf{C}$ is an arbitrary constant vector.

Definite Integral of a Vector-Valued Function

Let $\mathbf{r}(t) = f(t)\hat{\mathbf{i}} + g(t)\hat{\mathbf{j}} + h(t)\hat{\mathbf{k}}$, where f , g , and h are integrable on the interval $[a, b]$.

$$\int \mathbf{r}(t) dt = \left[\int_a^b f(t) dt \right] \hat{\mathbf{i}} + \left[\int_a^b g(t) dt \right] \hat{\mathbf{j}} + \left[\int_a^b h(t) dt \right] \hat{\mathbf{k}} \quad (53)$$

12.7 Motion In Space

Position, Velocity, Speed, Acceleration

Let the position of an object moving in three-dimensional space be given by $\mathbf{r}(t) = \langle x(t), y(t), z(t) \rangle$, for $t \geq 0$. The **velocity** of the object is

$$\mathbf{v}(t) = \mathbf{r}'(t) = \langle x'(t), y'(t), z'(t) \rangle \quad (54)$$

The **speed** of the object is the scalar function

$$|\mathbf{v}(t)| = \sqrt{x'(t)^2 + y'(t)^2 + z'(t)^2} \quad (55)$$

The **acceleration** of the object is $\mathbf{a}(t) = \mathbf{v}'(t) = \mathbf{r}''(t)$.

Motion with Constant $|\mathbf{r}|$

Let $\mathbf{r}$ describe a path on which $|\mathbf{r}|$ is constant (motion on a circle or sphere centered at the origin). Then, $\mathbf{r} \cdot \mathbf{v} = 0$, which means the position vector and the velocity vector are orthogonal at all times for which the functions are defined.

Two-Dimensional Motion in a Gravitational Field

Consider an object moving in a plane with horizontal x -axis and a vertical y -axis, subject only to the force of gravity. Given the initial velocity $\mathbf{v}(0) = \langle u_0, v_0 \rangle$ and the initial position $\mathbf{r}(0) = \langle x_0, y_0 \rangle$, the velocity of the object, for $t \geq 0$, is

$$\mathbf{v}(t) = \langle x'(t), y'(t) \rangle = \langle u_0, -gt + v_0 \rangle \quad (56)$$

and the position is

$$\mathbf{r}(t) = \langle x(t), y(t) \rangle = \left\langle u_0 t + x_0, -\frac{1}{2}gt^2 + v_0 t + y_0 \right\rangle \quad (57)$$

Two-Dimensional Motion

Assume an object traveling over horizontal ground, acted on only by the gravitational force, has an initial position $\langle x_0, y_0 \rangle = \langle 0, 0 \rangle$ and initial velocity

$\langle u_0, v_0 \rangle = \langle |\mathbf{v}_0| \cos \alpha, |\mathbf{v}_0| \sin \alpha \rangle$. The trajectory, which is a segment or a parabola, has the following properties.

$$\text{time of flight} = T = \frac{2|\mathbf{v}_0| \sin \alpha}{g} \quad (58)$$

$$\text{range} = \frac{|\mathbf{v}_0| \sin 2\alpha}{g} \quad (59)$$

$$\text{maximum height} = y\left(\frac{T}{2}\right) = \frac{(|\mathbf{v}_0| \sin \alpha)^2}{2g} \quad (60)$$

12.8 Length of Curves

Consider the parameterized curve $\mathbf{r}(t) = \langle f(t), g(t), h(t) \rangle$, where f' , g' , and h' are continuous, and the curve is traversed once for $a \leq t \leq b$. The **arc length** of the curve between $(f(a), g(a), h(a))$ and $(f(b), g(b), h(b))$ is

$$L = \int_a^b \sqrt{f'(t)^2 + g'(t)^2 + h'(t)^2} dt = \int_a^b |\mathbf{r}'(t)| dt \quad (61)$$

Arc Length of a Polar Curve

Let f have a continuous derivative on the interval $[\alpha, \beta]$. The **arc length** of the polar curve $r = f(\theta)$ on $[\alpha, \beta]$ is

$$L = \int_{\alpha}^{\beta} \sqrt{f(\theta)^2 + f'(\theta)^2} d\theta. \quad (62)$$

Arc Length as a Function of a Parameter

Let $\mathbf{r}(t)$ describe a smooth curve, for $t \geq a$. The arc length is given by

$$s(t) = \int_a^t |\mathbf{v}(u)| du, \quad (63)$$

where $|\mathbf{v}| = |\mathbf{r}'|$. Equivalently, $\frac{ds}{dt} = |\mathbf{v}t| > 0$. If $|\mathbf{v}(t)| = 1$, for all $t \geq a$, then the parameter t corresponds to arc length.

Arc Length as a Function of a Parameter

Let $\mathbf{r}(t)$ describe a smooth curve, for $t \geq a$. The arc length is given by

$$s(t) = \int_a^t |\mathbf{v}(u)| du, \quad (64)$$

where $|\mathbf{v}| = |\mathbf{r}'|$. Equivalently, $\frac{ds}{dt} = |\mathbf{v}(t)| > 0$. If $|\mathbf{v}(t)| = 1$, for all $t \geq a$, then the parameter t corresponds to arc length.

12.9 Curvature and Normal Vectors

Curvature

Let $\mathbf{r}$ describe a smooth parameterized curve. If s denotes arc length and $\mathbf{T} = \frac{\mathbf{r}'}{|\mathbf{r}'|}$ is the unit tangent vector, the **curvature** is $\kappa(s) = \left| \frac{d\mathbf{T}}{ds} \right|$

Curvature Formula

Let $\mathbf{r}(t)$ describes a smooth parameterized curve, where t is any parameter. If $\mathbf{v} = \mathbf{r}'$ is the velocity and $\mathbf{T}$ is the unit tangent vector, then the curvature is

$$\kappa(t) = \frac{1}{|\mathbf{v}|} \left| \frac{d\mathbf{T}}{dt} \right| = \left| \frac{\mathbf{T}'(t)}{\mathbf{r}'(t)} \right| \quad (65)$$

Alternative Curvature Formula

Let $\mathbf{r}$ be the position of an object moving on a smooth curve. The **curvature** at a point on the curve is

$$\kappa = \frac{|\mathbf{v} \times \mathbf{a}|}{|\mathbf{v}|^3}, \quad (66)$$

where $\mathbf{v} = \mathbf{r}'$ is the velocity and $\mathbf{a} = \mathbf{v}'$ is the acceleration.

Principal Unit Normal Vector

Let $\mathbf{r}$ describe a smooth parameterized curve. The **principal unit normal vector** at a point P on the curve at which $\kappa \neq 0$ is

$$\mathbf{N}(s) = \frac{d\mathbf{T}/ds}{|d\mathbf{T}/ds|} = \frac{1}{\kappa} \frac{d\mathbf{T}}{ds} \quad (67)$$

In practice, we use the equivalent formula

$$\mathbf{N}(t) = \frac{d\mathbf{T}/dt}{|d\mathbf{T}/dt|} \quad (68)$$

evaluated at the value of t corresponding to P .

Properties of the Principal Unit Normal Vector

Let $\mathbf{r}$ describe a smooth parameterized curve with unit tangent vector $\mathbf{T}$ and principal unit normal vector $\mathbf{N}$.

1. $\mathbf{T}$ and $\mathbf{N}$ are orthogonal at all points of the curve; that is, $\mathbf{T}(t) \cdot \mathbf{N}(t) = 0$, at all points where $\mathbf{N}$ is defined.
2. The principal unit normal vector points to the inside of the curve—in the direction that the curve is turning.

Tangential and Normal Components of the Acceleration

The acceleration vector of an object moving in space along a smooth curve has the following representation in terms of its **tangential component** a_T (in the direction of $\mathbf{T}$) and its normal component a_N (in the direction of $\mathbf{N}$):

$$\mathbf{a} = a_N \mathbf{N} + a_T \mathbf{T}, \quad (69)$$

where $a_N = \kappa |\mathbf{v}|^2 = \frac{|\mathbf{v} \times \mathbf{a}|}{|\mathbf{v}|}$ and $a_T = \frac{d^2 s}{dt^2}$.

Unit Binormal Vector and Torsion

Let C be a smooth parameterized curve with unit tangent and principal unit normal vectors $\mathbf{T}$ and $\mathbf{N}$, respectively. Then, at each point of the curve at which the curvature is nonzero, the **unit binormal vector** is

$$\mathbf{B} = \mathbf{T} \times \mathbf{N} \quad (70)$$

and the **torsion** is

$$\tau = -\frac{d\mathbf{B}}{ds} \cdot \mathbf{N} \quad (71)$$

Formulas for Curves in Space

1. Position function: $\mathbf{r}(t) = \langle x(t), y(t), z(t) \rangle$
2. Velocity: $\mathbf{v} = \mathbf{r}'$
3. Acceleration: $\mathbf{a} = \mathbf{v}'$

4. Unit tangent vector: $\mathbf{T} = \frac{\mathbf{v}}{|\mathbf{v}|}$
5. Principal unit normal vector: $\mathbf{N} = \frac{d\mathbf{T}/dt}{|d\mathbf{T}/dt|}$ (provided $d\mathbf{T}/dt \neq \mathbf{0}$)
6. Curvature: $\kappa = \frac{d\mathbf{T}}{ds} = \frac{1}{|\mathbf{v}|} \left| \frac{d\mathbf{T}}{dt} \right| = \left| \frac{\mathbf{T}'(t)}{\mathbf{r}'(t)} \right| = \frac{|\mathbf{v} \times \mathbf{a}|}{|\mathbf{v}|^3}$
7. Components of acceleration: $\mathbf{a} = a_N \mathbf{N} + a_T \mathbf{T}$, where $a_N = \kappa |\mathbf{v}|^2 = \frac{|\mathbf{v} \times \mathbf{a}|}{|\mathbf{v}|}$ and $a_T = \frac{d^2 s}{dt^2} = \frac{\mathbf{v} \cdot \mathbf{a}}{|\mathbf{v}|}$
8. Unit binormal vector: $\mathbf{B} = \mathbf{T} \times \mathbf{N} = \frac{\mathbf{v} \times \mathbf{a}}{|\mathbf{v} \times \mathbf{a}|}$
9. Torsion $\tau = -\frac{d\mathbf{B}}{ds} \cdot \mathbf{N} = \frac{(\mathbf{v} \times \mathbf{a}) \cdot \mathbf{a}'}{|\mathbf{v} \times \mathbf{a}|^2} = \frac{(\mathbf{r}' \times \mathbf{r}'') \cdot \mathbf{r}'''}{(\mathbf{r}' \times \mathbf{r}'')^2}$

13 Functions of Several Variables

13.1 Planes and Surfaces

Plane in $\mathbb{R}^3$

Given a fixed point P_0 and a nonzero **normal vector** $\mathbf{n}$, the set of points P in $\mathbb{R}^3$ for which $\vec{P_0P}$ is orthogonal to $\mathbf{n}$ is called a **plane**.

General Equation of a Plane in $\mathbb{R}^3$

The plane passing through the point $P_0(x_0, y_0, z_0)$ with a nonzero normal vector $\mathbf{n} = \langle a, b, c \rangle$ is described by the equation

$$a(x - x_0) + b(y - y_0) + c(z - z_0) = 0 \quad \text{or} \quad ax + by + cz = d \quad (72)$$

where $d = ax_0 + by_0 + cz_0$.

Parallel and Orthogonal Planes

Two distinct planes are **parallel** if their respective normal vectors are parallel (that is, the normal vectors are scalar multiples of each other). Two planes are **orthogonal** if their respective normal vectors are orthogonal (that is, the dot product of the normal vectors is zero).

Cylinder

Given a curve C in a plane P and a line ℓ not in P , a **cylinder** is the surface consisting of all lines parallel to ℓ that pass through C .

Trace

A **trace** of a surface is the set of points at which the surface intersects a plane that is parallel to one of the coordinate planes. The trace in the coordinate planes are called **xy-trace**, the **xz-trace**, and the **yz-trace**.

Quadratic Surfaces

Name	Standard Equation	Features
Ellipsoid	$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$	All traces are ellipses.
Elliptic paraboloid	$z = \frac{x^2}{a^2} + \frac{y^2}{b^2}$	Traces with $z = z_0 > 0$ are ellipses. Traces with $x = x_0$ or $y = y_0$ are parabolas.
Hyperboloid of one sheet	$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1$	Traces with $z = z_0$ are ellipses for all z_0 . Traces with $x = x_0$ or $y = y_0$ are hyperbolas.
Hyperboloid of two sheets	$-\frac{x^2}{a^2} - \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$	Traces with $z = z_0$ with $ z_0 > c $ are ellipses. Traces with $x = x_0$ and $y = y_0$ are hyperbolas.
Elliptic cone	$-\frac{x^2}{a^2} + \frac{y^2}{b^2} = \frac{z^2}{c^2}$	Traces with $z = z_0 \neq 0$ are ellipses. Traces with $x = x_0$ or $y = y_0$ are hyperbolas or intersecting lines.
Hyperbolic paraboloid	$z = \frac{x^2}{a^2} - \frac{y^2}{b^2}$	Traces with $z = z_0 \neq 0$ are hyperbolas. Traces with $x = x_0$ or $y = y_0$ are parabolas.

13.2 Graphs and Level Curves

Function, Domain, and Range with Two Independent Variables

A **function** $z = f(x, y)$ assigns to each point (x, y) in a set D in $\mathbb{R}^2$ a unique real number z in a subset of $\mathbb{R}$. The set D is the **domain** of f . The **range** of f is the set of real numbers z that are assumed as the points (x, y) vary over the domain.

Function, Domain, and Range with n Independent Variables

The **function** $y = f(x_1, x_2, \dots, x_n)$ assigns a unique real number y to each point $(x_1, x_2, \dots, x_n)$ in a set D in $\mathbb{R}^n$. The set D is the **domain** of f . The **range** is the set of real numbers y that are assumed as the points $(x_1, x_2, \dots, x_n)$ vary over the domain.

13.3 Limits and Continuity

Limit of a Function of Two Variables

The function f has the **limit** L as $P(x, y)$ approaches $P_0(a, b)$ written

$$\lim_{(x, y) \rightarrow (a, b)} f(x, y) = L \quad (73)$$

if, given any $\varepsilon > 0$, there exists a $\delta > 0$ such that

$$|f(x, y) - L| < \varepsilon \quad (74)$$

whenever (x, y) is in the domain of f and

$$0 < |PP_0| = \sqrt{(x - a)^2 + (y - b)^2} < \delta \quad (75)$$

Limits of Constant and Linear Functions

Let a, b , and c be real numbers.

1. Constant functions $f(x, y) = c : \lim_{(x, y) \rightarrow (a, b)} c = c$
2. Linear function $f(x, y) = x : \lim_{(x, y) \rightarrow (a, b)} x = a$
3. Linear function $f(x, y) = y : \lim_{(x, y) \rightarrow (a, b)} y = b$

Limit Laws and Functions of Two Variables

Let L and M be real numbers and suppose that $\lim_{(x, y) \rightarrow (a, b)} f(x, y) = L$ and $\lim_{(x, y) \rightarrow (a, b)} g(x, y) = M$. Assume c is a constant, and $\forall m, n \in \mathbb{Z}$.

1 Sum $\lim_{(x, y) \rightarrow (a, b)} (f(x, y) + g(x, y)) = L + M$

2 Difference $\lim_{(x, y) \rightarrow (a, b)} (f(x, y) - g(x, y)) = L - M$

3 Constant multiple $\lim_{(x, y) \rightarrow (a, b)} cf(x, y) = cL$

4 Product $\lim_{(x, y) \rightarrow (a, b)} f(x, y) \cdot g(x, y) = L \cdot M$

5 Quotient $\lim_{(x, y) \rightarrow (a, b)} \left[\frac{f(x, y)}{g(x, y)} \right] = \frac{L}{M}$

6 Power $\lim_{(x,y) \rightarrow (a,b)} (f(x, y))^n = L^n$

7 m/n Power If m and n have no common factors and $n \neq 0$, then $\lim_{(x,y) \rightarrow (a,b)} [f(x, y)]^{m/n} = L^{m/n}$, where we assume $L > 0$ if n is even.

Interior and Boundary Points

Let R be a region in $\mathbb{R}^2$. An **interior point** P of R lies entirely within R , which means it is possible to find a disk centered at P that contains only points of R .

A **boundary point** Q of R lies on the edge of R in the sense that *every* disk centered at Q contains at least one point in R and at least one point not in R .

Open and Closed Sets

A region is **open** if it consists entirely of interior points. A region is **closed** if it contains all its boundary points.

Two-Path Test for Nonexistence of Limits

If $f(x, y)$ approaches two different values as (x, y) approaches (a, b) along two different paths in the domain of f , then $\lim_{(x,y) \rightarrow (a,b)} f(x, y)$ does not exist.

Continuity

A function f is continuous at the point (a, b) provided

1. f is defined at (a, b) .
2. $\lim_{(x,y) \rightarrow (a,b)} f(x, y)$ exists.
3. $\lim_{(x,y) \rightarrow (a,b)} f(x, y) = f(a, b)$.

Continuity of Composite Functions

If $u = g(x, y)$ is continuous at (a, b) and $z = f(u)$ is continuous at $g(a, b)$, then the composite function $z = f(g(x, y))$ is continuous at (a, b) .

13.4 Partial Derivatives

The **partial derivative of f with respect to x at the point (a, b)** is

$$f_x(a, b) = \lim_{h \rightarrow 0} \frac{f(a + h, b) - f(a, b)}{h}. \quad (76)$$

The **partial derivative of f with respect to y at the point (a, b)** is

$$f_y(a, b) = \lim_{h \rightarrow 0} \frac{f(a, b + h) - f(a, b)}{h}. \quad (77)$$

provided these limits exists.

Equality of Mixed Partial Derivatives

Assume that f is defined on an open set D of $\mathbb{R}^2$, and f_{xy} and f_{yx} are continuous throughout D . Then $f_{xy} = f_{yx}$ at all points of D .

Differentiability

The function $z = f(x, y)$ is **differentiable at (a, b)** provided $f_x(a, b)$ and $f_y(a, b)$ exist and the change $\Delta z = f(a + \Delta x, b + \Delta y) - f(a, b)$ equals

$$\Delta z = f_x(a, b)\Delta x + f_y(a, b)\Delta y + \varepsilon_1\Delta x + \varepsilon_2\Delta y, \quad (78)$$

where for fixed a and b , ε_1 and ε_2 are functions that depend only on Δx and Δy , with $(\varepsilon_1, \varepsilon_2) \rightarrow (0, 0)$ as $(\Delta x, \Delta y) \rightarrow (0, 0)$. A function is **differentiable** on an open set R if it is differentiable at every point on R .

Conditions for Differentiability

Suppose the function f has partial derivatives f_x and f_y defined on an open set containing (a, b) , with f_x and f_y continuous (a, b) . Then f is differentiable at (a, b) .

Differentiability Implies Continuity

If a function f is differentiable at (a, b) , then it is continuous at (a, b)

13.5 The Chain Rule

Chain Rule (One Independent Variable)

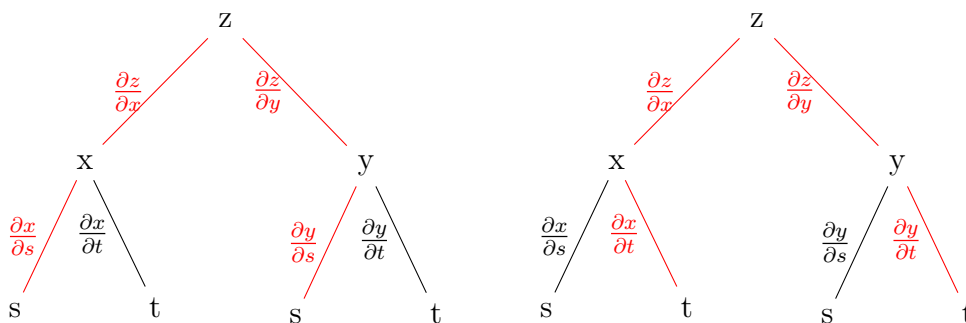
Let z be a differentiable function of x and y on its domain, where x and y are differentiable functions of t on an interval I . Then

$$\frac{dz}{dt} = \frac{\partial z}{\partial x} \frac{dx}{dt} + \frac{\partial z}{\partial y} \frac{dy}{dt}. \quad (79)$$

Chain Rule (Two Independent Variables)

Let z be a differentiable function of x and y , where x and y are differentiable functions of s and t . Then

$$\frac{dz}{ds} = \frac{\partial z}{\partial x} \frac{\partial x}{\partial s} + \frac{\partial z}{\partial y} \frac{\partial y}{\partial s} \quad \text{and} \quad \frac{dz}{dt} = \frac{\partial z}{\partial x} \frac{\partial x}{\partial t} + \frac{\partial z}{\partial y} \frac{\partial y}{\partial t} \quad (80)$$



Implicit Differentiation

Let F be differentiable on its domain and suppose that $F(x, y) = 0$ defines y as a differentiable function of x . Provided $F_y \neq 0$.

$$\frac{dy}{dx} = -\frac{F_x}{F_y} \quad (81)$$

13.6 Directional Derivatives and the Gradient

Directional Derivative

Let f be differentiable at (a, b) and let $\mathbf{u} = \langle \cos \theta, \sin \theta \rangle$ be a unit vector in the xy -plane. The **directional derivatives of f at (a, b) in the direction of $\mathbf{u}$** is

$$D_{\mathbf{u}}f(a, b) = \lim_{h \rightarrow 0} \frac{f(a + h \cos \theta, b + h \sin \theta) - f(a, b)}{h} \quad (82)$$

provided the limit exists.

Directional Derivative

Let f be differentiable on (a, b) and let $\mathbf{u} = \langle u_1, u_2 \rangle$ be a unit vector in the x, y -plane. The **directional derivative of f at (a, b) in the direction of $\mathbf{u}$** is

$$D_{\mathbf{u}}f(a, b) = \langle f_x(a, b), f_y(a, b) \rangle \cdot \langle u_1, u_2 \rangle \quad (83)$$

Gradient (Two Dimensions)

Let f be differentiable at the point (x, y) . The **gradient of f at (x, y)** is the vector-valued function

$$\nabla f(x, y) = \langle f_x(x, y), f_y(x, y) \rangle = f_x(x, y)\mathbf{i} + f_y(x, y)\mathbf{j} \quad (84)$$

Directions of Change

Let f be differentiable at (a, b) with $\nabla f(a, b) \neq \mathbf{0}$

1. f has its maximum rate of increase at (a, b) in the direction of the gradient $\nabla f(a, b)$. The rate of increase in this direction is $|\nabla f(a, b)|$
2. f has its maximum rate of decrease at (a, b) in the direction of $-\nabla f(a, b)$. The rate of decrease in this direction is $-|\nabla f(a, b)|$.
3. The directional derivative is zero in any direction orthogonal to $\nabla f(a, b)$.

The Gradient and Level Curves

Given a function f differentiable at (a, b) , the line tangent to the level curve of f at (a, b) is orthogonal to the gradient $\nabla f(a, b)$, provided $\nabla f(a, b) \neq \mathbf{0}$.

Gradient and Directional Derivative in Three Dimensions

Let f be differentiable at the point (x, y, z) . The **gradient** of f at (x, y, z) is the vector-valued function

$$\nabla f(x, y, z) = \langle f_x(x, y, z), f_y(x, y, z), f_z(x, y, z) \rangle \quad (85)$$

$$= f_x(x, y, z)\hat{\mathbf{i}} + f_y(x, y, z)\hat{\mathbf{j}} + f_z(x, y, z)\hat{\mathbf{k}} \quad (86)$$

The **directional derivative** of f in the direction of the unit vector $\mathbf{u} = \langle u_1, u_2, u_3 \rangle$ at the point (a, b, c) is $D_{\mathbf{u}}f(a, b, c) = \nabla f(a, b, c) \cdot \mathbf{u}$

13.7 Tangent Planes and Linear Approximation

Let F be differentiable at the point $P_0(a, b, c)$ with $\nabla F(a, b, c) \neq \mathbf{0}$. The plane tangent to the surface $F(x, y, z) = 0$ at P_0 , called the **tangent plane**, is the plane passing through P_0 orthogonal $\nabla F(a, b, c)$. An equation of the tangent plane is

$$F_x(a, b, c)(x - a) + F_y(a, b, c)(y - b) + F_z(a, b, c)(z - c) = 0. \quad (87)$$

Tangent Plane for $z = f(x, y)$

Let f be differentiable at the point (a, b) . An equation of the plane tangent to the surface $z = f(x, y)$ at the point $(a, b, f(a, b))$ is

$$z = f_x(a, b)(x - a) + f_y(a, b)(y - b) + f(a, b) \quad (88)$$

Linear Approximation

Let f be differentiable at (a, b) . The linear approximation to the surface $z = f(x, y)$ at the point $(a, b, f(a, b))$ is the tangent plane at that point, given by the equation

$$L(x, y) = f_x(a, b)(x - a) + f_y(a, b)(y - b) + f(a, b) \quad (89)$$

The Differential dz

Let f be differentiable at the point (a, b) . The change in $z = f(x, y)$ as the independent variables change from (a, b) to $(a + dx, b + dy)$ is denoted by the differential dz :

$$\Delta z \approx dz = f_x(a, b)dx + f_y(a, b)dy \quad (90)$$

13.8 Maximum/Minimum Problems

Local Maximum/Minimum Values

A function f has a **local maximum value** at (a, b) if $f(x, y) \leq f(a, b)$ for all (x, y) in the domain of f in some open disk centered at (a, b) . A function f has a **local minimum value** at (a, b) if $f(x, y) \geq f(a, b)$ for all (x, y) in the domain of f in some open disk centered at (a, b) . Local maximum and local minimum values are also called **local extreme values** or **local extrema**.

Derivatives and Local Maximum/Minimum Values

If f has a local maximum or minimum value at (a, b) and the partial derivatives f_x and f_y exist at (a, b) then $f_x(a, b) = f_y(a, b) = 0$.

Critical Point

An interior point (a, b) in the domain of f is a **critical point** of f if either

1. $f_x(a, b) = f_y(a, b) = 0$, or
2. one (or both) of f_x or f_y does not exist at (a, b)

Saddle Point

A function f has a **saddle point** at a critical point (a, b) if, in every open disk centered at (a, b) , there are points (x, y) for which $f(x, y) > f(a, b)$ and points for which $f(x, y) < f(a, b)$

Second Derivative Test

Suppose that the second partial derivative of f are continuous throughout an open disk centered at the point (a, b) , where $f_x(a, b) = f_y(a, b) = 0$. Let $D(x, y) = f_{xx}(x, y)f_{yy}(x, y) - (f_{xy}(x, y))^2$.

1. If $D(a, b) > 0$ and $f_{xx}(a, b) < 0$, then f has a local maximum value at (a, b) .
2. If $D(a, b) > 0$ and $f_{xx}(a, b) > 0$, then f has a local minimum value at (a, b) .

3. If $D(a, b) < 0$, then f has a saddle point at (a, b) .
4. If $D(a, b) = 0$, then the test is inconclusive.

Absolute Maximum/Minimum Values

If $f(x, y) \leq f(a, b)$ for all (x, y) in the domain of f , then f has an **absolute maximum value** at (a, b) . If $f(x, y) \geq f(a, b)$ for all (x, y) in the domain of f , then f has an **absolute minimum value** at (a, b) .

Finding Absolute Maximum/Minimum Values on Closed, Bounded Sets

Let f be continuous on a closed bounded set R in $\mathbb{R}^2$. To find the absolute maximum and minimum values of f on R :

1. Determine the values of f at all critical points in R .
2. Find the maximum and minimum values of f on the boundary of R .
3. The greatest function values found in Step 1 and 2 is the absolute maximum value of f on R , and the least function value found in Steps 1 and 2 is the absolute minimum value of f on R .

13.9 Lagrange Multipliers

Parallel Gradients (Ball Park Theorem)

Let f be a differentiable function in a region of $\mathbb{R}^2$ that contains the smooth curve C given by $g(x, y) = 0$. Assume that f has a local extreme value (relative to values of f on C) at a point $P(a, b)$ on C . Then $\nabla f(a, b)$ is orthogonal to the line tangent to C at P . Assuming $\nabla g(a, b) \neq 0$, it follows that there is a real number λ (called a **Lagrange multiplier**) such that $\nabla f(a, b) = \lambda \nabla g(a, b)$.

Method of Lagrange Multipliers in Two Variables

Let the objective function f and the constraint function g be differentiable on a region of $\mathbb{R}^2$ with $\nabla g(x, y) \neq 0$ on the curve $g(x, y) = 0$. To locate the maximum and minimum values of f subject to the constraint $g(x, y) = 0$, carry out the following steps.

1. Find the values of x, y and λ (if they exist) that satisfy the equations

$$\nabla f(x, y) = \lambda \nabla g(x, y) \quad \text{and} \quad g(x, y) = 0 \quad (91)$$

2. Among the values (x, y) found in Step 1, select the largest and smallest corresponding function values, which are the maximum and minimum values of f subject to the constraint.

Method of Lagrange Multipliers in Three Variables

Let f and g be differentiable on a region of $\mathbb{R}^3$ with $\nabla g(x, y, z) \neq 0$ on the surface $g(x, y, z) = 0$. To locate the maximum and minimum values of f subject to the constraint $g(x, y, z) = 0$, carry out the following steps.

1. Find the values of x, y, z and λ that satisfy the equations

$$\nabla f(x, y, z) = \lambda \nabla g(x, y, z) \quad \text{and} \quad g(x, y, z) = 0 \quad (92)$$

2. Among the points (x, y, z) found in Step 1, select the largest and smallest corresponding values of the objective function. These values are the maximum and minimum values of f subject to the constraint.

14 Multiple Integration

14.1 Double Integrals over Rectangular Regions

Volumes and Double Integrals

A function f defined on a rectangular region R in the xy -plane is **integratable** on R if $\lim_{\Delta \rightarrow 0} \sum_{k=1}^n f(x_k^*, y_k^*) \Delta A_k$ exists for all partitions of R and for all choices of (x_k^*, y_k^*) within those partitions. The limit is the **double integral of f over R** , which we write

$$\iint_R f(x, y) dA = \lim_{\Delta \rightarrow 0} \sum_{k=1}^n f(x_k^*, y_k^*) \Delta A_k \quad (93)$$

If f is nonnegative on R , then the double integral equals the volume of the solid bounded by $z = f(x, y)$ and the xy -plane over R .

Double Integrals on Rectangular Regions

Let f be continuous on the rectangular region $R = \{(x, y) : a \leq x \leq b, c \leq y \leq d\}$. The double integral of f over R may be evaluated by either of two iterated integrals:

$$\iint_R f(x, y) dA = \int_c^d \int_a^b f(x, y) dx dy = \int_a^b \int_c^d f(x, y) dy dx \quad (94)$$

Average Value of a Function over a Plane Region

The **average value** of an integrable function f over a region R is

$$\bar{f} = \frac{1}{\text{area of } R} \iint_R f(x, y) dA \quad (95)$$

14.2 Double Integrals over General Regions

Let R be a region bounded below and above by the graphs of the continuous functions $y = g(x)$ and $y = h(x)$, respectively, and by the lines $x = a$ and $x = b$. If f is continuous on R , then

$$\iint_R f(x, y) dA = \int_a^b \int_{g(x)}^{h(x)} f(x, y) dy dx \quad (96)$$

Let R be a region bounded on the left and right by the graphs of the continuous functions $x = g(y)$ and $x = h(y)$, respectively, and the lines $y = c$ and $y = d$. If f is continuous on R , then

$$\iint_R f(x, y) dA = \int_c^d \int_{g(y)}^{h(y)} f(x, y) dx dy \quad (97)$$

Areas of Regions by Double Integrals

Let R be a region in the xy -plane. Then

$$\text{area of } R = \iint_R 1 dA \quad (98)$$

14.3 Double Integrals in Polar Coordinates

Double Integrals over Polar Rectangular Region

Let f be continuous on the region in the xy -plane $R = \{(r, \theta) : 0 \leq a \leq r \leq b, \alpha \leq \theta \leq \beta\}$, where $\beta - \alpha \leq 2\pi$. Then

$$\iint_R f(r, \theta) dA = \int_{\alpha}^{\beta} \int_a^b f(r, \theta) r dr d\theta \quad (99)$$

Double Integrals over More General Polar Regions

Let f be continuous on the region in the xy -plane

$$R = \{(r, \theta) : 0 \leq g(\theta) \leq r \leq h(\theta), \alpha \leq \theta \leq \beta\} \quad (100)$$

where $0 < \beta - \alpha \leq 2\pi$. Then

$$\iint_R f(r, \theta) dA = \int_{\alpha}^{\beta} \int_{g(\theta)}^{h(\theta)} f(r, \theta) r dr d\theta. \quad (101)$$

Area of Polar Regions

The area of the region $R = \{(r, \theta) : 0 \leq g(\theta) \leq r \leq h(\theta), \alpha \leq \theta \leq \beta\}$, where $0 < \beta - \alpha \leq 2\pi$, is

$$A = \iint_R dA = \int_{\alpha}^{\beta} \int_{g(\theta)}^{h(\theta)} r dr d\theta \quad (102)$$

14.4 Triple Integrals

Let f be continuous over the region

$$D = \{(x, y, z) : a \leq x \leq b, g(x) \leq y \leq h(x), G(x, y) \leq z \leq H(x, y)\} \quad (103)$$

where g, h, G , and H are continuous functions. Then f is integrable over D and the triple integral is evaluated as the iterated integral

$$\iiint_D f(x, y, z) dV = \int_a^b \int_{g(x)}^{h(x)} \int_{G(x, y)}^{H(x, y)} f(x, y, z) dz dy dx. \quad (104)$$

Average Value of a Function of Three Variables

If f is continuous on a region D of $\mathbb{R}^3$, then the average value of f over D is

$$\bar{f} = \frac{1}{\text{volume}(D)} \iiint_D f(x, y, z) dV \quad (105)$$

14.5 Triple Integrals in Cylindrical and Spherical Coordinates

Transformations Between Cylindrical and Rectangular Coordinates

Rectangular $\rightarrow$ Cylindrical Cylindrical $\rightarrow$ Rectangular

$$r^2 = x^2 + y^2$$

$$\tan \theta = y/x$$

$$z = z$$

$$x = r \cos \theta$$

$$y = r \sin \theta$$

$$z = z$$

Triple Integrals in Cylindrical Coordinates

Let f be continuous over the region

$$D = \{(r, \theta, z) : g(\theta) \leq r \leq h(\theta), \alpha \leq \theta \leq \beta, G(x, y) \leq z \leq H(x, y)\} \quad (106)$$

Then f is integrable over D and the triple integral of f over D in cylindrical coordinates is

$$\iiint_D f(r, \theta, z) dV = \int_{\alpha}^{\beta} \int_{g(\theta)}^{h(\theta)} \int_{G(r \cos \theta, r \sin \theta)}^{H(r \cos \theta, r \sin \theta)} f(r, \theta, z) dz r dr d\theta \quad (107)$$

Transformations Between Spherical and Rectangular Coordinates

Rectangular $\rightarrow$ Spherical Spherical $\rightarrow$ Rectangular

$$\rho^2 = x^2 + y^2 + z^2$$

Use trigonometry to find φ and θ

$$x = \rho \sin \varphi \cos \theta$$

$$y = \rho \sin \varphi \sin \theta$$

$$z = \rho \cos \varphi$$

Triple Integrals in Spherical Coordinates

Let f be continuous over the region

$$D = \{(\rho, \varphi, \theta) : g(\varphi, \theta) \leq \rho \leq h(\varphi, \theta), a \leq \varphi \leq b, \alpha \leq \theta \leq \beta\} \quad (108)$$

Then f is integrable over D , and the triple integral of f over D in spherical coordinates is

$$\iiint_D f(\rho, \varphi, \theta) dV = \int_{\alpha}^{\beta} \int_a^b \int_{g(\varphi, \theta)}^{h(\varphi, \theta)} f(\rho, \varphi, \theta) \rho^2 \sin \varphi \, d\rho \, d\varphi \, d\theta \quad (109)$$

14.6 Integrals for Mass Calculations

Center of Mass in One Dimension

Let ρ be an integrable density function on the interval $[a, b]$ (which represents a thin rod or wire). The **center of mass** is location at the point $\bar{x} = \frac{M}{m}$, where the **total moment** M and mass m are

$$M = \int_a^b x\rho(x) dx \quad \text{and} \quad m = \int_a^b \rho(x) dx \quad (110)$$

Center of Mass in Two Dimensions

Let ρ be integrable density function defined over a closed bounded region R in $\mathbb{R}^2$. The coordinates of the center of mass of the object represented by R are

$$\bar{x} = \frac{M_y}{m} = \frac{1}{m} \iint_R x\rho(x, y) dA \quad \text{and} \quad \bar{y} = \frac{M_x}{m} = \frac{1}{m} \iint_R y\rho(x, y) dA \quad (111)$$

Center of Mass in Three Dimensions

Let ρ be integrable density function on a closed bounded region D in $\mathbb{R}^3$. The coordinates of the center of mass of the region are

$$\bar{x} = \frac{M_{yz}}{m} = \frac{1}{m} \iiint_R x\rho(x, y, z) dV \quad (112)$$

$$\bar{y} = \frac{M_{xz}}{m} = \frac{1}{m} \iiint_R y\rho(x, y, z) dV \quad (113)$$

$$\bar{z} = \frac{M_{xy}}{m} = \frac{1}{m} \iiint_R z\rho(x, y, z) dV \quad (114)$$

where $m = \iiint_D \rho(x, y, z) dV$ is the mass, and M_{yz} , M_{xz} and M_{xy} are the moments with respect to the coordinate planes.

15 Vector Calculus

15.1 Vector Fields

Vector Fields in Two Dimensions

Let f and g be defined on a region R of $\mathbb{R}^2$. A **vector field** in $\mathbb{R}^2$ is a function $\mathbf{F}$ that assigns to each point in R a vector $\langle f(x, y), g(x, y) \rangle$. The vector field is written as

$$\mathbf{F}(x, y) = \langle f(x, y), g(x, y) \rangle \quad \text{or} \quad \mathbf{F}(x, y) = f(x, y)\hat{\mathbf{i}} + g(x, y)\hat{\mathbf{j}} \quad (115)$$

A vector field $\mathbf{F} = \langle f, g \rangle$ is continuous or differentiable on a region R of $\mathbb{R}^2$ if f and g are continuous or differentiable on R , respectively.

Radial Vector Fields in $\mathbb{R}^2$

Let $\mathbf{r} = \langle x, y \rangle$. A vector field of the form $\mathbf{F} = f(x, y)\mathbf{r}$, where f is a scalar-valued function, is a **radial vector field**. Of specific interest are the radial vector field

$$\mathbf{F}(x, y) = \frac{\mathbf{r}}{|\mathbf{r}|^p} = \frac{\langle x, y \rangle}{|\mathbf{r}|^p} \quad (116)$$

where p is a real number. At every point (except the origin), the vectors of this field are directed outward from the origin with the magnitude of $|\mathbf{F}| = \frac{1}{|\mathbf{r}|^{p-1}}$.

Vector Fields in Three Dimensions

Let f , g , and h be defined on a region D of $\mathbb{R}^3$. A **vector field** in $\mathbb{R}^3$ is a function $\mathbf{F}$ that assigns to each point in D a vector $\langle f(x, y, z), g(x, y, z), h(x, y, z) \rangle$. The vector field is written as

$$\mathbf{F}(x, y, z) = \langle f(x, y, z), g(x, y, z), h(x, y, z) \rangle \quad \text{or} \quad (117)$$

$$\mathbf{F}(x, y, z) = f(x, y, z)\hat{\mathbf{i}} + g(x, y, z)\hat{\mathbf{j}} + h(x, y, z)\hat{\mathbf{k}} \quad (118)$$

A vector field $\mathbf{F} = \langle f, g, h \rangle$ is continuous or differentiable on a region D of $\mathbb{R}^3$ if f , g , h are continuous or differentiable on R , respectively. Of particular importance are the **radial vector fields**

$$\mathbf{F}(x, y, z) = \frac{\mathbf{r}}{|\mathbf{r}|^p} = \frac{\langle x, y, z \rangle}{|\mathbf{r}|^p} \quad (119)$$

where p is a real number.

Gradient Fields and Potential Functions

Let $z = \varphi(x, y)$ and $w = \varphi(x, y, z)$ be differentiable functions on regions of $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively. The vector field $\mathbf{F} = \nabla\varphi$ is **gradient field**, and the function φ is a **potential function** for $\mathbf{F}$.

15.2 Line Integrals

Scalar Line Integral in the Plane, Arc Length Parameter

Suppose the scalar-valued function f is defined on the smooth curve $C : \mathbf{r}(s) = \langle x(s), y(s) \rangle$, parameterized by the arc length s . The **line integral of f over C** is

$$\int_C f(x(s), y(s)) ds = \lim_{\Delta \rightarrow 0} \sum_{k=1}^n f(x(s_k^*), y(s_k^*)) \Delta s_k, \quad (120)$$

provided this limit exists over all partitions of C . When the limit exists, f is said to be **integrable** on C .

Evaluating Scalar Line Integrals in $\mathbb{R}^2$

Let f be continuous on a region containing a smooth curve $C : \mathbf{r}(t) = \langle x(t), y(t) \rangle$, for $a \leq t \leq b$. Then

$$\int_C f ds = \int_a^b f(x(t), y(t)) |\mathbf{r}'(t)| dt \quad (121)$$

$$= \int_a^b f(x(t), y(t)) \sqrt{x'(t)^2 + y'(t)^2} dt \quad (122)$$

Evaluating the Line Integral $\int_C f ds$

1. Find a parametric description of C in the form $\mathbf{r}(t) = \langle x(t), y(t) \rangle$, for $a \leq t \leq b$
2. Computer $|\mathbf{r}'(t)| = \sqrt{x'(t)^2 + y'(t)^2}$
3. Make substitutions for x and y in the integrand and evaluate an ordinary integral

$$\int_C f ds = \int_a^b f(x(t), y(t)) |\mathbf{r}'(t)| dt \quad (123)$$

Evaluating Scalar Line Integrals in $\mathbb{R}^3$

Let f be continuous on a region containing a smooth curve $C : \mathbf{r}(t) = \langle x(t), y(t), z(t) \rangle$, for $a \leq t \leq b$. Then

$$\int f \, ds = \int_a^b f(x(t), y(t), z(t)) |\mathbf{r}'(t)| \, dt \quad (124)$$

$$= \int_a^b f(x(t), y(t), z(t)) \sqrt{x'(t)^2 + y'(t)^2 + z'(t)^2} \, dt \quad (125)$$

Line Integral of a Vector Field

Let $\mathbf{F}$ be a vector field that is continuous on a region containing a smooth oriented curve C parameterized by arc length. Let $\mathbf{T}$ be the unit tangent vector at each point of C consistent with the orientation. The line integral of $\mathbf{F}$ over C is $\int_C \mathbf{F} \cdot \mathbf{T} \, ds$.

Different Forms of Line Integrals of Vector Fields

The line integral $\int_C \mathbf{F} \cdot \mathbf{T} \, ds$ may be expressed in the following forms, where $\mathbf{F} = \langle f, g, h \rangle$, for $a \leq t \leq b$:

$$\int_a^b \mathbf{F} \cdot \mathbf{r}'(t) \, dt = \int_a^b (f x'(t), g y'(t), h z'(t)) \, dt \quad (126)$$

$$= \int_C f \, dx + g \, dy + h \, dz \quad (127)$$

$$= \int_C \mathbf{F} \cdot d\mathbf{r} \quad (128)$$

For line integrals in the plane, we let $\mathbf{F} = \langle f, g \rangle$ and assume C is parameterized in the form $\mathbf{r}(t) = \langle x(t), y(t) \rangle$, for $a \leq t \leq b$. Then

$$\int_C \mathbf{F} \cdot \mathbf{T} \, ds = \int_a^b (f x'(t) + g y'(t)) \, dt = \int_C f \, dx + g \, dy = \int_C \mathbf{F} \cdot d\mathbf{r} \quad (129)$$

Work Done in a Force Field

Let $\mathbf{F}$ be a continuous force field in a region D of $\mathbb{R}^3$ and let $C : \mathbf{r}(t) = \langle x(t), y(t), z(t) \rangle$, for $a \leq t \leq b$, be a smooth curve in D with a unit tangent vector $\mathbf{T}$ consistent with the orientation. The work done in moving an object C in the positive direction is

$$W = \int_C \mathbf{F} \cdot \mathbf{T} \, ds = \int_a^b \mathbf{F} \cdot \mathbf{r}'(t) \, dt \quad (130)$$

Circulation

Let $\mathbf{F}$ be a continuous vector field on a region D of $\mathbb{R}^3$ and let C be a closed smooth oriented curve in D . The **circulation** of $\mathbf{F}$ on C is $\int_C \mathbf{F} \cdot \mathbf{T} \, ds$, where $\mathbf{T}$ is the unit vector tangent to C consistent with the orientation.

Flux

Let $F = \langle f, g \rangle$ be continuous vector field on a region R of $\mathbb{R}^2$. Let $C : \mathbf{r}(t) = \langle x(t), y(t) \rangle$, for $a \leq t \leq b$, be a smooth oriented curve in R that does not intersect itself. The **flux** of the vector field across C is

$$\int_C \mathbf{F} \cdot \mathbf{n} \, ds = \int_a^b (f y'(t) - g x'(t)) \, dt, \quad (131)$$

where $\mathbf{n} = \mathbf{T} \times \hat{\mathbf{k}}$ is the unit normal vector and $\mathbf{T}$ is the unit tangent vector consistent with the orientation. If C is a closed curve with counterclockwise orientation, $\mathbf{n}$ is the outward normal vector and the flux integral gives the **outward flux** across C .

15.3 Conservative Vector Fields

Simple and Closed Curves

Suppose a curve C (in $\mathbb{R}^2$ and $\mathbb{R}^3$) is described parametrically by $\mathbf{r}(t)$, where $a \leq t \leq b$. Then C is a **simple curve** if $\mathbf{r}(t_1) \neq \mathbf{r}(t_2)$ for all t_1 and t_2 , with $a < t_1 < t_2 < b$; that is, C never intersects itself between its endpoints. The curve C is **closed** if $\mathbf{r}(a) = \mathbf{r}(b)$; that is, the initial and terminal points of C are the same.

Connected and Simply Connected Regions

An open region R in $\mathbb{R}^2$ (or D in $\mathbb{R}^3$) is **connected** if it is possible to connect any two points of R by a continuous curve lying in R . An open region R is **simply connected** if every closed simple curve in R can be deformed and contracted to a point in R .

Conservative Vector Field

A vector field F is said to be **conservative** on a region (in $\mathbb{R}^2$ or $\mathbb{R}^3$) if there exists a scalar function φ such that $\mathbf{F} = \nabla\varphi$ on that region.

Test for Conservative Vector Fields

Let $\mathbf{F} = \langle f, g, h \rangle$ be a vector field defined on a connected and simply connected region D of $\mathbb{R}^3$, where f , g , and h have continuous first partial derivatives on D . Then $\mathbf{F}$ is a conservative vector field on D (there is a potential function φ such that $\mathbf{F} = \nabla\varphi$) if and only if

$$\frac{\partial f}{\partial y} = \frac{\partial g}{\partial x}, \quad \frac{\partial f}{\partial z} = \frac{\partial h}{\partial x}, \quad \text{and} \quad \frac{\partial g}{\partial z} = \frac{\partial h}{\partial y} \quad (132)$$

For vector fields in $\mathbb{R}^2$, we have the single condition $\frac{\partial f}{\partial y} = \frac{\partial g}{\partial x}$.

Finding Potential Functions in $\mathbb{R}^3$

Suppose $\mathbf{F} = \langle f, g, h \rangle$ is a conservative vector field. To find φ such that $\mathbf{F} = \nabla\varphi$, take the following steps:

1. Integral $\varphi_x = f$ with respect to x to obtain φ , which includes an arbitrary function $c(y, z)$.

2. Compute φ_y and equate it to g to obtain an expression for $c_y(y, z)$.
3. Integrate $c_y(y, z)$ with respect to y to obtain $c(y, z)$, including an arbitrary function $d(z)$.
4. Compute φ_z and equate it to h to get $d(z)$.

Beginning the procedure with $\varphi_y = g$ or $\varphi_z = h$ maybe be easier in some cases.

Fundamental Theorem for Line Integrals

Let $\mathbf{F}$ be a continuous vector field on an open connected region R in $\mathbb{R}^2$ (or D in $\mathbb{R}^3$). There exists a potential function φ with $\mathbf{F} = \nabla\varphi$ (which means that $\mathbf{F}$ is conservative) if and only if

$$\int_C \mathbf{F} \cdot \mathbf{T} \, ds = \int_C \mathbf{F} \cdot d\mathbf{r} = \varphi(B) - \varphi(A) \quad (133)$$

for all points A and B in R and all smooth oriented curves C from A to B .

Line Integrals on Closed Curves

Let R in $\mathbb{R}^2$ (or D in $\mathbb{R}^3$) be an open region. Then $\mathbf{F}$ is a conservative vector field on R if and only if $\oint_C \mathbf{F} \cdot d\mathbf{r} = 0$ on all simple closed smooth oriented curves C in R .

15.4 Green's Theorem

Let C be a simple closed smooth curve, oriented counterclockwise, that encloses a connected and simply connected region R in the plane. Assume $\mathbf{F} = \langle f, g \rangle$, where f and g have continuous first partial derivatives in R . Then

$$\oint_C \mathbf{F} \cdot d\mathbf{r} = \oint_C f dx + g dy = \iint_R \left(\frac{\partial g}{\partial x} - \frac{\partial f}{\partial y} \right) dA. \quad (134)$$

Two-Dimensional Curl

The **two-dimensional curl** of the vector field $\mathbf{F} = \langle f, g \rangle$ is $\frac{\partial g}{\partial x} - \frac{\partial f}{\partial y}$. If the curl is zero throughout a region, the vector field is said to be **irrotational** on that region.

Area of a Plane Region by Line Integrals

Under the conditions of Green's Theorem, the area of a region R enclosed by a curve C is

$$\oint_C x dy = - \oint_C y dx = \frac{1}{2} \oint_C (x dy - y dx) \quad (135)$$

Green's Theorem, Flux Form

Let C be a simple closed smooth curve, oriented counterclockwise, that encloses a connected and simply connected region R in the plane. Assume $\mathbf{F} = \langle f, g \rangle$, where f and g have continuous first partial derivatives in R . Then

$$\oint_C \mathbf{F} \cdot \mathbf{n} ds = \oint_C f dy - g dx = \iint_R \left(\frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} \right) dA \quad (136)$$

where $\mathbf{n}$ is the outward unit normal vector on the curve.

Two-Dimensional Divergence

The **two-dimensional divergence** of the vector field $\mathbf{F} = \langle f, g \rangle$ is $\frac{\partial f}{\partial x} + \frac{\partial g}{\partial y}$. If the divergence is zero throughout a region, the vector field is said to be **source free** on that region.

15.5 Divergence and Curl

Divergence of a Vector Field

The **divergence** of a vector field $\mathbf{F} = \langle f, g, h \rangle$ that is differentiable on a region of $\mathbb{R}^3$ is

$$\operatorname{div} \mathbf{F} = \nabla \cdot \mathbf{F} = \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} + \frac{\partial h}{\partial z} \quad (137)$$

If $\nabla \cdot \mathbf{F} = 0$, the vector field is **source free**.

Divergence of Radial Vector Fields

For a real number p , the divergence of the radial vector field

$$\mathbf{F} = \frac{\mathbf{r}}{|\mathbf{r}|^p} = \frac{\langle x, y, z \rangle}{(x^2 + y^2 + z^2)^{\frac{p}{2}}} \text{ is } \nabla \cdot \mathbf{F} = \frac{3-p}{|\mathbf{r}|^p} \quad (138)$$

Curl of a Vector Field

The **curl** of a vector field $\mathbf{F} = \langle f, g, h \rangle$ that is differentiable on a region of $\mathbb{R}^3$ is

$$\nabla \times \mathbf{F} = \operatorname{curl} \mathbf{F} = \left(\frac{\partial h}{\partial y} - \frac{\partial g}{\partial z} \right) \hat{\mathbf{i}} + \left(\frac{\partial f}{\partial z} - \frac{\partial h}{\partial x} \right) \hat{\mathbf{j}} + \left(\frac{\partial g}{\partial x} - \frac{\partial f}{\partial y} \right) \hat{\mathbf{k}} \quad (139)$$

If $\nabla \times \mathbf{F} = \mathbf{0}$, the vector field is **irrotational**.

Curl of a Conservative Vector Field

The **general rotation vector field** is $\mathbf{F} = \mathbf{a} \times \mathbf{r}$, where the nonzero constant vector $\mathbf{a} = \langle a_1, a_2, a_3 \rangle$ is the axis of rotation and $\mathbf{r} = \langle x, y, z \rangle$. For all nonzero choices of $\mathbf{a}$, $|\nabla \times \mathbf{F}| = 2|\mathbf{a}|$ and $\nabla \cdot \mathbf{F} = 0$. The constant angular speed of the vector field is

$$\omega = |\mathbf{a}| = \frac{1}{2} |\nabla \times \mathbf{F}| \quad (140)$$

Curl of a Conservative Vector Field

Suppose that $\mathbf{F}$ is a conservative vector field on an open region D of $\mathbb{R}^3$. Let $\mathbf{F} = \nabla\varphi$, where φ is a potential function with continuous second partial derivatives on D . Then $\nabla \times \mathbf{F} = \nabla \times \nabla\varphi = \mathbf{0}$; that is, the curl of the gradient is the zero vector and $\mathbf{F}$ is irrotational.

Divergence of the Curl

Suppose that $\mathbf{F} = \langle f, g, h \rangle$, where f , g , and h have continuous second partial derivatives. Then $\nabla \cdot (\nabla \times \mathbf{F}) = 0$: The divergence of the curl is zero.

Product Rule for the Divergence

Let u be a scalar-valued function that is differentiable on a region D and let $\mathbf{F}$ be a vector field that is differentiable on D . Then

$$\nabla \cdot (u\mathbf{F}) = \nabla u \cdot \mathbf{F} + u(\nabla \cdot \mathbf{F}) \quad (141)$$

Properties of a Conservative Vector Field

Let $\mathbf{F}$ be a conservative vector field whose components have continuous second partial derivatives on an open connected region D in $\mathbb{R}^3$. Then $\mathbf{F}$ has the following equivalent properties.

1. There exists a potential function φ such that $\mathbf{F} = \nabla\varphi$
2. $\int_C \mathbf{F} \cdot d\mathbf{r} = \varphi(B) - \varphi(A)$ for all points A and B in D and all smooth oriented curves C from A and B .
3. $\oint_C \mathbf{F} \cdot d\mathbf{r} = 0$ on all simple smooth closed oriented curves C in D .
4. $\nabla \times \mathbf{F} = \mathbf{0}$ at all points of D .

15.6 Surface Integrals

Surface Integrals of Scalar-Valued Functions on Parameterized Surface

Let f be a continuous function on a smooth surface S given parametrically by $\mathbf{r}(u, v) = \langle x(u, v), y(u, v), z(u, v) \rangle$, where $R = \{(u, v) : a \leq u \leq b, c \leq v \leq d\}$. Assume also that the tangent vectors $\mathbf{t}_u = \frac{\partial \mathbf{r}}{\partial u} = \langle \frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u} \rangle$, and $\mathbf{t}_v = \frac{\partial \mathbf{r}}{\partial v} = \langle \frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v} \rangle$ are continuous on R and the normal vectors $\mathbf{n} = \mathbf{t}_u \times \mathbf{t}_v$ is nonzero on R . Then the **surface integral** of the scalar-valued function f over S is

$$\iint_S f(x, y, z) dS = \iint_R f(x(u, v), y(u, v), z(u, v)) |\mathbf{t}_u \times \mathbf{t}_v| dA \quad (142)$$

If $f(x, y, z) = 1$, the integral equals the surface area of S .

Evaluation of Surface Integrals of Scalar-Valued Functions on Explicitly Defined Surfaces

Let f be a continuous function on a smooth surfaces S given by $z = g(x, y)$, for (x, y) in a region R . The surface integral of f over S is

$$\iint_S f(x, y, z) dS = \iint_R f(x, y, g(x, y)) \sqrt{z_x^2 + z_y^2 + 1} dA \quad (143)$$

If $f(x, y, z) = 1$, the surface integral equals the area of the surface.

Surface Integral of a Vector Field

Suppose $\mathbf{F} = \langle f, g, h \rangle$ is a continuous vector field on a region of $\mathbb{R}^3$ containing a smooth oriented surface S . If S is defined parametrically as $\mathbf{r}(u, v) = \langle x(u, v), y(u, v), z(u, v) \rangle$, for (u, v) is a region R , then

$$\iint_S \mathbf{F} \cdot \mathbf{n} dS = \iint_R \mathbf{F} \cdot (\mathbf{t}_u \times \mathbf{t}_v) dA \quad (144)$$

where $\mathbf{t}_u = \frac{\partial \mathbf{r}}{\partial u} = \langle \frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u} \rangle$ and $\mathbf{t}_v = \frac{\partial \mathbf{r}}{\partial v} = \langle \frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v} \rangle$ are continuous on R , the normal vector $\mathbf{n} = \mathbf{t}_u \times \mathbf{t}_v$ is nonzero on R , and the direction of $\mathbf{n}$ is

consistent with the orientation of S . If S is defined in the form $z = g(x, y)$, for (x, y) in a region R , then

$$\iint_S \mathbf{F} \cdot \mathbf{n} \, dS = \iint_R (-fz_x - gz_y + h) \, dA \quad (145)$$

15.7 Stokes' Theorem

Let S be a smooth oriented surface in $\mathbb{R}^3$ with a smooth closed boundary C whose orientation is consistent with that of S . Assume that $\mathbf{F} = \langle f, g, h \rangle$ is a vector field whose components have continuous first partial derivatives on S . Then

$$\oint_C \mathbf{F} \cdot d\mathbf{r} = \iint_S (\nabla \times \mathbf{F}) \cdot \mathbf{n} \, dS \quad (146)$$

where $\mathbf{n}$ is the unit vector normal to S determined by the orientation of S .

Curl $\mathbf{F} = \mathbf{0}$ Implies $\mathbf{F}$ is Conservative

Suppose that $\nabla \times \mathbf{F} = \mathbf{0}$ throughout an open simply connected region D of $\mathbb{R}^3$. Then $\oint_C \mathbf{F} \cdot d\mathbf{r} = 0$ on all closed simple smooth curves C in D and $\mathbf{F}$ is a conservative vector field on D .

15.8 Divergence Theorem

Let $\mathbf{F}$ be a vector field whose components have continuous first partial derivatives in a connected and simply connected region D in $\mathbb{R}^3$ enclosed by a smooth oriented surface S . Then

$$\iint_S \mathbf{F} \cdot \mathbf{n} \, dS = \iiint_D \nabla \cdot \mathbf{F} \, dV \quad (147)$$

where $\mathbf{n}$ is the unit outward normal vector on S .

Divergence Theorem for Hollow Regions

Suppose the vector field $\mathbf{F}$ satisfies the conditions of the Divergence Theorem on a region D bounded by two smooth oriented surfaces S_1 and S_2 , where S_1 lies within S_2 . Let S be the entire boundary of D ($S = S_1 \cup S_2$) and let $\mathbf{n}_1$ and $\mathbf{n}_2$ be the outward unit normal vectors for S_1 and S_2 , respectively. Then

$$\iiint_D \nabla \cdot \mathbf{F} \, dV = \iint_S \mathbf{F} \cdot \mathbf{n} \, dS = \iint_{S_2} \mathbf{F} \cdot \mathbf{n}_2 \, dS + \iint_{S_1} \mathbf{F} \cdot \mathbf{n}_1 \, dS \quad (148)$$

STAT3117

Statistics

S&TTM

Confidence Intervals for a Population Mean

Let $X_1, \dots, X_n$ be a simple random sample from a population with mean μ and variance σ^2 . Let $\bar{X}$ be the sample mean, and S_n be the sum of sample observation. If n is **sufficiently large**,

X ~ N (mu, sigma^2/n)

and

Sn ~ N (nmu, n sigma^2)

Let $X_1, \dots, X_n$ be a **large** ($n > 30$) random sample from a population with mean μ and standard deviation σ , so that $\bar{X}$ is approximately normal. Then a level $100(1 - \alpha)\%$ confidence interval for μ is

Xbar +/- z_alpha/2 sigma_X

where $\sigma_{\bar{X}} = \frac{\sigma}{\sqrt{n}}$. When the value of σ is unknown, it can be replaced with the sample standard deviation s .

Small Sample Confidence Intervals for a Population Mean

Let $X_1, \dots, X_n$ be a small ($n < 30$) sample from a *normal* population with mean μ . Then the quantity

(Xbar - mu) / (s/sqrt(n))

has a Student's t distribution with $n - 1$ degrees of freedom, denoted t_{n-1} .

When n is large, the distribution of quantity $\frac{\bar{X}-\mu}{\frac{s}{\sqrt{n}}}$ is very close to normal, so the normal curve can be used, rather than the Student's t .

Hypothesis Testing

Large-Sample Tests for a Population Mean

Let $X_1, \dots, X_n$ be a **large** ($n > 30$) sample form a population with mean μ and standard deviation σ .

To test a null hypothesis of the form $H_0 : u \leq u_0, H_0 : u \geq \mu_0, H_0 := \mu_0$:

- Compute the z -score:

z = (Xbar - mu_0) / (s/sqrt(n))

If σ is unknown it may be approximated with s .

- Computer the P -value. The P -value is an area under the normal curve, which depends on the alternate hypothesis as follows:

Alternate Hypothesis	P-Value
$H_1 : \mu > \mu_0$	Area to the right of z
$H_1 : \mu < \mu_0$	Area to the left of z
$H_1 : \mu = \mu_0$	Sum of the areas cut off by z and $-z$

Drawing Conclusions from the Results of Hypothesis Tests

Let α be any value between 0 and 1. Then, if $P \leq \alpha$,

- The result of the test is said to be statistically significant at the $100\alpha\%$ level.
- The null hypothesis is rejected at the $100\alpha\%$ level.
- When reporting the result of the hypothesis test, report the P -value, rather than just comparing it to the 5% or 1%.

Small-Sample Tests for a Population Mean

Let $X_1, \dots, X_n$ be a **small** ($n \leq 30$) random sample from a *normal* population with mean μ (unknown) and a standard deviation σ .

To test a null hypothesis of the form $H_0 : \mu \leq \mu_0, H_0 : \mu \geq \mu_0$, or $H_0 : \mu = \mu_0$:

- Compute the test statistic

t* = (Xbar - mu_0) / (s/sqrt(n))

- Compute the P -value. The P -value is an area under the Student's t curve with $n - 1$ degrees of freedom, which depends on the alternate hypothesis as follows

Alternate Hypothesis	P-Value
$H_1 : \mu > \mu_0$	Area to the right of z
$H_1 : \mu < \mu_0$	Area to the left of z
$H_1 : \mu = \mu_0$	Sum of the areas cut off by z and $-z$

- If σ is known, the test statistic is $z = \frac{\bar{X}-\mu_0}{\frac{\sigma}{\sqrt{n}}}$

Large-Sample Tests for the Difference Between Two Means

Let $X_1, \dots, X_n$ and $Y_1, \dots, Y_n$ be **large** ($n_x > 30$ and $n_y > 30$) independent random samples from populations with mean μ_x and μ_y and standard deviation σ_x and σ_y , respectively.

The test statistic is as follows:

z* = (Xbar - Ybar - delta_0) / sqrt(sigma_x^2/n_x + sigma_y^2/n_y)

If σ_X and σ_Y are unknown they may be replaced by s_X and s_Y , respectively

Null Hypothesis	Alternative Hypothesis	p-value
$H_0 : \mu_x - \mu_y \leq \delta_0$	$H_1 : \mu_x - \mu_y > \delta_0$	$P(Z \geq z^*)$
$H_0 : \mu_x - \mu_y \geq \delta_0$	$H_1 : \mu_x - \mu_y < \delta_0$	$P(Z \leq z^*)$
$H_0 : \mu_x - \mu_y = \delta_0$	$H_1 : \mu_x - \mu_y \neq \delta_0$	$2 \times P(Z \geq z^*)$

Small-Sample Tests for the Difference Between Two Means

Population Variances Are Not Equal

Let $X_1, \dots, X_{n,x}$ and $Y_1, \dots, Y_{n,y}$ be samples from *normal* populations with means μ_X and μ_Y and standard deviations σ_X and σ_Y , respectively. Assume the samples are drawn independently of each other.

If σ_x and σ_Y are **not known to be equal**, then, to test a null hypothesis of the form $H_0 : \mu_X - \mu_Y \leq \Delta_0, H_0 : \mu_X - \mu_Y \geq \Delta_0$, or $H_0 : \mu_X - \mu_Y = \Delta_0$.

- Rounding down to the nearest integer, calculate

nu = ((s_x^2/n_x + s_y^2/n_y) / ((s_x^2/n_x)^2 / (n_x - 1) + (s_y^2/n_y)^2 / (n_y - 1)))^2

- Compute the test statistic

t = (Xbar - Ybar - Delta_0) / sqrt(Sx^2/n_x + Sy^2/n_y)

- Compute the P -value. The P -value is an area under the Student's t curve with ν degrees of freedom, which depends on the alternate hypothesis as follows:

Alternate Hypothesis	P-value
$H_1 : \mu_X - \mu_Y > \Delta_0$	Area to the right of t
$H_1 : \mu_X - \mu_Y < \Delta_0$	Area to the left of t
$H_1 : \mu_X - \mu_Y \neq \Delta_0$	Sum of the areas in the tails cut off

Population Variances Are Equal

Let $X_1, \dots, X_{n,x}$ and $Y_1, \dots, Y_{n,y}$ be samples from *normal* populations with means μ_X and μ_Y and standard deviations σ_X and σ_Y , respectively. Assume the samples are drawn independently of each other.

If σ_X and σ_Y are known to be equal, then, to test a null hypothesis of the form $H_0 : \mu_X - \mu_Y \leq \Delta_0, H_0 : \mu_X - \mu_Y \geq \Delta_0$, or $H_0 : \mu_x - \mu_y = \Delta_0$:

- Compute

sp = sqrt((nX - 1)sX^2 + (nY - 1)sY^2) / (nX + nY - 2)

- Compute the test statistic

t = (Xbar - Ybar - Delta_0) / (sp * sqrt(1/nX + 1/nY))

- Compue the P -value. The P -value is an area under the Student's t curve with $n_X + n_Y - 2$ degrees of freedom, which depends on the alternate hypothesis as follows.

Alternate Hypothesis	P-value
$H_1 : \mu_X - \mu_Y > \Delta_0$	Area to the right of t
$H_1 : \mu_X - \mu_Y < \Delta_0$	Area to the left of t
$H_1 : \mu_X - \mu_Y \neq \Delta_0$	Sum of the areas in the tails cut off

Correlation vs. Causation

Correlation

A correlation coefficient (denoted r) measures the strength and direction of a linear relationship between two variables. Let $(x_1, y_1), \dots, (x_n, y_n)$ represent bivariate data, then the correlation coefficient is

r = (1/(n-1) * sum((xi - xbar)/sx * (yi - ybar)/sy)) = (sum(xi*yi - n*xbar*ybar) / sqrt(sum(xi^2 - n*xbar^2) * sum(yi^2 - n*ybar^2))) = SSR/SST

The Least-Squares Line

For an equation of the form

y1 = beta_0 + beta_1 x_i + epsilon_i

y_i is called the **dependent variable**, x_i is called the **indepedent variable**, β is called the **regression coefficients** (the least squares coefficients), and ϵ_i is called the **error**.

Also, r^2 is the **proportion of variance in y explained by regression**.

e1 = yi - y-hat_i = yi - beta-hat_0 - beta-hat_1 xi

beta-hat_1 = beta_1 = (sum yi * xi - n * xbar * ybar) / (sum xi^2 - n * xbar^2)

beta-hat_0 = y-bar - beta-hat_1 x-bar

Uncertainties in the Least-Squares Coefficients

Using some assumptions,

- The quantity $\hat{\beta}$ is *normally distributed* random variables.
- The means of $\hat{\beta}$ is the true values of $\hat{\beta}$.
- The *standard deviations* of $\hat{\beta}$ is estimated with

s_beta_0 = s * sqrt(1/n + xbar^2 / (sum (xi - xbar)^2))

s_beta_1 = s / sqrt(sum (xi - xbar)^2)

where

$$s = \sqrt{\frac{(1-r^2) \sum_{i=1}^n (y_i - \bar{y})^2}{n-2}}$$

is an estimate of the error standard deviation σ .

Confidence Intervals for Coefficients

Under assumptions, the quantities $\frac{\hat{\beta}_1 - \beta_1}{s_{\hat{\beta}_1}}$ and $\frac{\hat{\beta}_1 - \beta_1}{s_{\hat{\beta}_1}}$ have Student's t distributions with $n - 2$ degrees of freedom. Level $100(1 - \alpha)\%$ confidence intervals for β_0 and β_1 are given by

$$\hat{\beta}_0 \pm t_{n-2} \times s_{\hat{\beta}_0} \quad \hat{\beta}_1 \pm t_{n-2} \times s_{\hat{\beta}_1}$$

Level $100(1 - \alpha)\%$ confidence intervals for the quantity $\beta_0 + \beta_1 x$ is given by

$$\widehat{\beta}_0 + \widehat{\beta}_1 x \pm t_{n-2, \alpha/2} \times s_{\hat{y}}$$

where

$$s_{\hat{y}} = s \sqrt{\frac{1}{n} + \frac{(x - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2}}$$

Checking Assumptions

If the plot of residuals versus fitted values

- Shows no substantial trend or curve, and
- Is **homoscedastic**, that is, the vertical spread does not vary too much along the horizontal length of the plot, except perhaps near the edges,

then it is *likely*, but not certain, that the assumptions of the linear model hold.

However, if the residual plot *does* show a substantial trend or curve, or is **heteroscedastic**, it is certain that the assumptions of the linear plot *do not* hold.

Miscellaneous Notes

- A **test statistic** is a function of the sample data whose value is used to test a hypothesis
- A **p-value** is a measure of the disagreement between a sample and H_0 .
- The smaller the P -value, the more certain we can be that H_0 is false and vice versa.
- For **large samples**, we approximate the population standard deviation σ using the sample standard deviation s .
- The correlation coefficient is called the **sample correlation (r)**, and the it is an estimate of the population correlation (ρ).
- Some properties of the correlation coefficient (r):
 1. $-1 \leq r \leq 1$, r is unitless.
 2. If the points lie exactly on a horizontal or vertical line, the correlation coefficient is undefined, because one of the standard deviations is equal to zero.
 3. Whenever $r \neq 0$, x and y are said to be correlated. If $r = 0$, x and y are said to be uncorrelated.
 4. Correlation coefficient is unaffected by the units in which the measurements are made.
- For **small samples**, s may be far from σ , which invalidates this large-sample method. However, when the population is approximately normal, the Student's t distribution can be used.

- The **pooled** sample variance is

$$s_p^2 = \frac{(n_X - 1)s_X^2 + (n_Y - 1)s_Y^2}{n_X + n_Y - 2}$$

- The correlation coefficient remains unchanged under each of the following operations
 - Multiplying each value of a variable by a positive constant.
 - Adding a constant to each of a variable.
 - Interchanging the values of x and y .
- A **goodness-of-fit statistic** measures how well a model explains a given set of data.
- SST = total sum of squares = $\sum_{i=1}^n (y_i - \bar{y})^2$
- SSE = error sum of squares = $\sum_{i=1}^n (y_i - \hat{y})^2$
- SSR = regression sum of squares = $SST - SSE$
- The following assumptions are satisfied.
 - The errors $\epsilon_1, \dots, \epsilon_n$ are random and independent. In particular, the magnitude of any error ϵ_i does not influence the value of the next error ϵ_{i+1} .
 - The errors $\epsilon_1, \dots, \epsilon_n$ all have mean 0.
 - The errors $\epsilon_1, \dots, \epsilon_n$ all have the same variance, which we denote by σ^2 .
 - The errors $\epsilon_1, \dots, \epsilon_n$ are normally distributed.
- Margin of error = $t_{\frac{\alpha}{2}, \sqrt{n}} \times \frac{s}{\sqrt{n}}$
- The sample variance is calculated

$$\frac{1}{N-1} \sum_{i=0}^n (x - \bar{x})^2$$

Project Report

STAT 3113 — Engineering Statistics

Caroline Ketterer, Sam McGraw, Illya Starikov, Timothy Ott

Due Date: November 30th, 2017

Contents

1 Objective

A useful tool in cooking is determining roughly how long it will take to boil water. There are several factors that might affect how fast water boils, including:

- Amount of salt in the water.
- The size of the pot.

This experiment will specifically determine how said factors affect how long it takes water to reach 100 °C.

2 Experiment Design

Below we will lay out the design of our experiment.

2.1 Factors and Levels

We design a two factor, three level experiment — a 3^2 factorial experiment. Please refer to Table ?? for a complete description.

Table 1: Factor and Level Definition

Factor	Level 1	Level 2	Level 3
<i>Pan Size</i>	153.94 cm ²	254.47 cm ²	314.16 cm ²
<i>Salt Weight</i>	No Salt	17 g	34 g

2.2 Response Variable

In our experiment, the response variable is the amount of time it takes for two cups of water to reach 100 °C.

2.3 Blocking Factors

There is only one possible blocking factor for our experiment: the initial temperature of the pot. We circumvent this by placing cold water in all of the pans initially and letting all the pans reach thermal equilibrium before starting the experiment.

2.4 Number of Replicants

Each treatment combination is replicated three times.

3 Experiment Plan

The following section will describe the layout of the experiment.

3.1 Letting Pans Reach Thermal Equilibrium

The first step is to ensure that all pans have the same initial temperature.

We accomplish this by placing cold water (5 °C) in both of the pans for 5:00 minutes. After this, we can proceed with the experiment.

3.2 Measure Out Salt

Note: This step does not apply to the treatment with no salt.

We measure out the salt on a scale to the desired weight (whether it be 17 g or 34 g).

After pouring said salt into all of the pans individually, we stir until we reach a homogeneous, saltwater mixture.

3.3 Conduct The Experiment

We proceed to set the temperature of all heating elements “high”. We continually measure all pots to see if they have reached 100 °C.



Figure 1: Timothy Ott Conducting The Experiment

If they have reached the desired temperature, take the pot off the heater and mark the time. Do so for all pots.

After all pans have finished, mark the recorded data — these notes can be found in Section ??

4 Data Collection

All data collected can be summarized Table ??; additionally, Section ?? contains the data as it was collected during the experiment.

Table 2: Number of Minutes To Reach 100 °C (No Salt)

	No Salt			17 g Salt			34 g Salt		
Big	6:00.18	6:29.03	6:36.66	4:57.91	5:32.9	5:41.30	4:55.35	5:20.36	5:01.71
Medium	7:36.48	5:41.17	6:56.42	5:22.78	6:17.04	5:55.60	5:33.12	5:34.46	5:35.75
Small	8:15.79	6:57.90	7:13.85	6:08.14	6:34.02	6:47.33	6:03.25	6:08.36	6:04.19

5 Data Analysis

We entered the data of our experiment into the JMP software as shown below. The data can be viewed in Figure??

	Salt Weight	Pan Size	Boil Time (Seconds)
1	0	153.94	495.79
2	0	153.94	417.5
3	0	153.94	433.85
4	17	153.94	368.14
5	17	153.94	394.02
6	17	153.94	407.33
7	34	153.94	363.25
8	34	153.94	368.36
9	34	153.94	364.17
10	0	254.47	456.48
11	0	254.47	341.17
12	0	254.47	416.42
13	17	254.47	322.78
14	17	254.47	317.04
15	17	254.47	355.6
16	34	254.47	333.12
17	34	254.47	334.46
18	34	254.47	335.75
19	0	314.16	360.18
20	0	314.16	389.03
21	0	314.16	396.66
22	17	314.16	297.97
23	17	314.16	332.9
24	17	314.16	341.3
25	34	314.16	295.35
26	34	314.16	320.86
27	34	314.16	301.71

Figure 2: The Input Used By JMP.

To analyze the data we first needed to test for the presence of interaction effect.

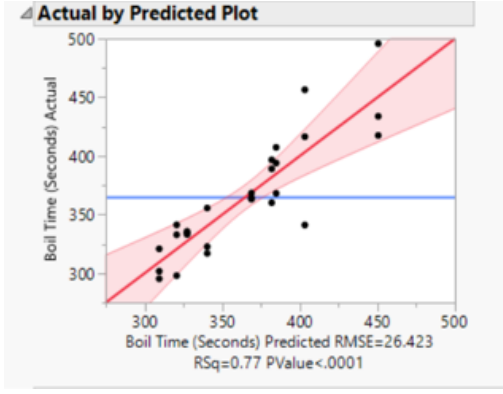
$$H_0 : \gamma_{11} = \gamma_{12} = \dots = \gamma_{IJ} = 0$$

$$H_1 : \text{at least one of the } \gamma_{ij} \text{ is nonzero}$$

Because the p -value of Salt Water*Pan Size is 0.7625 (Figure ??), which is greater than our significance level of 0.05, we fail to reject our null hypothesis and conclude that the interaction effect is not significant. Therefore, we can test for the main effects.

6 Conclusion

To test the main effects we first need to define our null and alternative hypotheses. We will first consider pan size.



Effect Tests					
Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
Salt Weight(0.34)	1	1	26445.132	37.8780	<.0001*
Pan Size(153.94,314.16)	1	1	18460.811	26.4419	<.0001*
Salt Weight*Salt Weight	1	1	3749.167	5.3700	0.0307*
Pan Size*Pan Size	1	1	104.126	0.1491	0.7032
Salt Weight*Pan Size	1	1	65.435	0.0937	0.7625

(a) The Actual by Predicted Plot

(b) The Main Effects Table

Figure 3: The Results

$$H_0 : \alpha_1 = \alpha_2 = \dots = \alpha_I = 0$$

$$H_1 : \text{at least one of the } \alpha_i \text{ is nonzero}$$

The p -value for pan size is a number smaller than 0.0001 which is smaller than our significance level of 0.05. Therefore, we fail to reject our null hypothesis and conclude that pan size does have an effect on the time it takes for water to reach 100 °C.

We will now test for the main effect of salt weight.

$$H_0 : \beta_1 = \beta_2 = \dots = \beta_J = 0$$

$$H_1 : \text{at least one of the } \beta_j \text{ is nonzero}$$

The p -value for salt weight is a number smaller than 0.0001 which is again smaller than our significance level of 0.05. Therefore, we fail to reject our null hypothesis and conclude that salt weight does have an effect on the time it takes for water to reach 100 °C.

7 Improvement Suggestions

There are still things that can be improved in our experiment; specifically, they are:

Filtered Water Using filtered water, as opposed to tap water, would also improve the accuracy of the experiment. Tap water naturally contains various minerals, ions, and other particles that vary greatly between samples. Filtered water would remove the interference of those particles, making the salt the only additive in the water.

Better Temperature Measurement Creating a rig to hold the thermometer in the water would also improve the accuracy of our results. Manually dipping the thermometer between the pots adds some variation between measurements, while having some method for holding the thermometer in the center of the pot would ensure consistent, unbiased results.

Atmospheric Pressure Controlling the atmospheric pressure at which the experiment was conducted would improve the accuracy of this experiment. The air pressure above the water changes the temperature/amount of time required to boil water, and since this experiment took place over the course of several hours, the atmospheric pressure is liable to change. Conducting the experiment in a controlled, pressurized environment would eliminate the variation in boiling pressures, thus producing consistent results.

8 Appendix

The following are notes collected from the experiment.

PHYS1135

Physics I



0.1 State Variable

C_v is specific heat for constant volume.

p, V, T, n, U . Cannot write Q , because you can pull heat in and out. The relationships of these:

$$\begin{aligned}pV &= nRT \\U &= \frac{3}{2}nRT \\&= N_{\text{molecules}} E_{\text{kinetic}}\end{aligned}$$

We can describe this in a pV diagram. When you see a pV diagram that has an area under the curve, you can calculate it with:

$$W = \int p dv$$

How this is derived:

$$\begin{aligned}W &= \int F dv \\&= Ap \frac{dv}{A} \\&= \int p dv\end{aligned}$$

Now talking about non state variable:

$$U = Q - W \tag{1}$$

0.2 Processes

isobaric $p = \text{constant}$

isochoric $V = \text{constant}$

isothermic $T = \text{constant}$

adiabatic $Q = \text{constant}$

To maintain the same pressure (isobaric), apply heat and increase temperature. To maintain the same volume (isochoric), cool it down. To keep the temperature the same (isothermal), keep it in thermodynamic equilibrium while changing the pressure / volume via a heat bath. To keep the heat the same, insulate the system.

1. Isobaric.

- $W = \int p dV = p \Delta V = nR \Delta T.$
- $Q = nC_p \Delta T$
- $\Delta U = Q - W = n(c_p - R) \Delta T = nc_v \Delta T$

2. Isochoric.

- $W = \int p dV = 0$
- $Q = nc_v \Delta T$
- $\Delta U = Q = nc_v \Delta T = \frac{3}{2} nRT \rightarrow c_v = \frac{3}{2} R$

3. Isothermal.

- $W = \int p dV = \int_{v_i}^{v_f} (nRT) dV = nRT \ln(\frac{V_f}{V_i})$
- $\Delta U = 0 \rightarrow Q - W = 0 \rightarrow Q = W$

4. Adiabatic

- $\Delta Q = 0 \rightarrow \Delta U = -\Delta W \rightarrow du = -dW$
- Just look in lecture notes for derivation.
- $pV^\gamma = pV^{\frac{c_p}{c_v}}$ constant

PHYS2135

Physics II



1 Electric Charge, Coulomb's Law, Electric Field, Motion of a Charge in Electric Field

1.1 Book Notes

- When charges are at rest in our frame of reference, they exert electrostatic forces on each other.
- Electrostatic forces are governed by a simple relationship known as Coulomb's law and are most conveniently described by using the concept of electric field.
- Two positive charges or two negative charges repel each other. A positive charge and a negative charge attract each other.
- The algebraic sum of all the electric charges in any closed system is constant.
- The magnitude of charge of the electron or proton is a natural unit of charge.
- The electric force on a charged body is exerted by the electric field created by other charged bodies.

1.2 Lecture Notes

- Law of conservation of charge: the net amount of electric charge produced in any process is zero.

1.3 Recitation

- Office is 115
 - Tue/Thu: 10-Noon, 2-4:30
 - Wed: 8-2, 2-4:30
 - Fri: 8-Noon

2 Electric Field of a Charge Distribution

2.1 Lecture Notes

- $\lambda \neq$ to wavelength, but linear charge density (charge per length).
- Arc length is angle (radians) times radius ($S = r\theta$).
- No two field lines can cross.

2.2 Recitation

- Because in nature, point charges are not common, we do a distributed charges.
 - We do this via superposition (i.e. add up vectorily to get the total charge).
- Suppose you have a rod, and an origin at the center, and a point h distance at the origin.
 - $dE_x = dE \sin \theta$
 - Likewise, $dE_y = dE \cos \theta$

$$dE_y = dE \cos \theta \tag{1}$$

$$= \frac{k\lambda dx}{r^2} \frac{h}{r} \tag{2}$$

- This is one of two charge distribution we often encounter.
 - Still the same process.

3 Electric Field Lines, Electric Dipoles, Electric Flux, Gauss' Law

3.1 Book Notes

- The direction of $\tilde{\mathbf{p}}$ is from the negative to the positive charge.

3.2 Lecture Notes

- The electric field always depends on qd .
 - dipole moment vector $\vec{p} = q\vec{d}$
 - Torque on the dipole is exactly the same as classical mechanics.
- Remember, zero potential energy does not mean minimum potential energy!
- The **electric flux** passing through a surface is the number of electric field lines that pass through it.
- For a closed surface, $d\mathbf{A}$ is normal to the surface and always points away from the inside.
- The electric field is a vector field, so a constant electric field is one that does not change with position or time.
- If a conductor is in electrostatic equilibrium, any excess charge must lie on its surface, so for the charge to be uniformly distributed throughout the volume, the object must be an insulator.

3.3 Recitation

- Electric field lines just give an easy way to imagine what the force would be.
- The distance $\vec{d}$ points from the negative to the positive.
 - That where the dipole moment comes from.
- $U = -\vec{p}\vec{E}$
- $\phi_E = \oint \vec{E} \cdot d\vec{A}$
 - Area vector points outward.

$$\phi_E = \oint \vec{E} \cdot d\vec{A} \tag{1}$$

$$= \oint E \, da \cos \theta \tag{2}$$

Cross product expansions

$$= E \oint dA \tag{3}$$

If E is constant, pull it out

$$= E \times \text{Surface Area} \tag{4}$$

(5)

4 Gauss' Law Calculations, Conductors and Electric Fields

4.1 Book Notes

- Whether there is a net outward or inward electric flux through a closed surface depends on the sign of the enclosed charge.
- Charges outside the surface do not give a net electric flux through the surface.
- The net electric flux is directly proportional to the net amount of charge enclosed within the surface but is otherwise independent of the size of the closed surface.
- Gauss's law states that the total electric flux through any closed surface (a surface enclosing a definite volume) is proportional to the total (net) electric charge inside the surface.
- When excess charge is placed on a solid conductor and is at rest, it resides entirely on the surface, not in the interior of the material.
- Electrostatic equilibrium means there is no net motion of the charges inside the conductor.
 - The electric field inside the conductor must be zero.
 - If this were not the case, charges would accelerate.
- Any excess charge must reside on the outside surface of the conductor.
- The electric field just outside a charged conductor must be perpendicular to the conductor's surface.
- The magnitude of the electric field just outside a charged conductor is equal to $\frac{|\sigma|}{\epsilon_0}$, where $|\sigma|$ is the magnitude of the local surface charge density.

4.2 Lecture Notes

- The electric field inside the conductor must be zero.
 - If not, the system would accelerate.

4.3 Recitation

- We're doing some vector review.
 - $\vec{A} \cdot \vec{B} = AB \cos \theta$
 - * max at $\theta = 0$
 - * min at $\theta = \pi$

- $|\vec{A} \times \vec{B}| = |AB \sin \theta|$

- * max at $\theta = \frac{\pi}{2}$

- * min at $\theta = 0$

- Back to physics. $\vec{F} = q\vec{E}$
- Inside conductor, $E = 0$. Otherwise electrons would still be moving.
 - This brings us to Gauss's law.
- A good way to calculate charge would be to start at the inner most surface and work on the way out.

5 Electric Potential, Electric Potential Energy

5.1 Book Notes

- The potential-energy difference $U_a - U_b$ equals the work that is done by the electric force when the particle moves from a to b . When U_a is greater than U_b , the field does positive work on the particle as it “falls” from a point of higher potential energy (a) to a point of lower potential energy (b).
- The potential-energy difference $U_a - U_b$ is then defined as the work that must be done by an external force to move the particle slowly from b to a against the electric force.
- **Potential** is potential energy per unit charge.
- SI unit of potential is called one volt (1V).
- The potential difference between two points is often called **voltage**.
- V_{ab} , the potential of a with respect to b , equals the work that must be done to move a UNIT charge slowly from b to a against the electric force.
- The electric potential at a certain point is the potential energy that would be associated with a unit charge placed at that point. That’s why potential is measured in joules per coulomb, or volts. Keep in mind, too, that there doesn’t have to be a charge at a given point for a potential V to exist at that point. (In the same way, an electric field can exist at a given point even if there’s no charge there to respond to it.)
- Moving with the direction of $\tilde{\mathbf{E}}$ means moving in the direction of *decreasing* V , and moving against the direction of $\tilde{\mathbf{E}}$ means moving in the direction of *increasing* V .

5.2 Lecture Notes

- $\Delta U = -[W_{\text{conservative}}]_{i \rightarrow f}$
 - Always ask yourself which work you are calculating.
- Don’t fall into the trap of making r_{12} a square.
- Potential energies are defined relative to some configuration of objects that you are free to choose.
 - For example, it often makes sense to define the gravitational potential energy of a ball to be zero when it is resting on the surface of the earth, but you don’t have to make that choice.

- Our equation for the electric potential energy of two charged particles uses the convention that the potential energy is zero when the particles are infinitely far apart.
- Protons fall down, electrons fall up.
- An electron volt (eV) is the energy acquired by a particle of charge e when it moves through a potential difference of 1 volt.

6 Electric Potentials of Charge Distributions, Equipotentials, Potential Gradient

6.1 Book Notes

- An equipotential surface is a three-dimensional surface on which the electric potential V is the same at every point.
- On a given equipotential surface, the potential V has the same value at every point. In general, however, the electric-field magnitude E is not the same at all points on an equipotential surface.
- When all charges are at rest, the surface of a conductor is always an equipotential surface
- In an electrostatic situation, if a conductor contains a cavity and if no charge is present inside the cavity, then there can be no net charge anywhere on the surface of the cavity.
- surface charge density on the wall of the cavity is zero at every point.
- Don't confuse equipotential surfaces with the Gaussian surfaces we encountered in Chapter 22. Gaussian surfaces have relevance only when we are using Gauss's law, and we can choose any Gaussian surface that's convenient. We cannot choose equipotential surfaces; the shape is determined by the charge distribution.

6.2 Lecture Notes

- The disk of charge will be on exam.
- The "Ed" equation does not require rectangular plates, or any plates at all. It works as long as E is uniform and parallel or antiparallel to d .

6.3 Recitation

- Potential is a scalar, which awesommmmmee.
 - But we can't exploit symmetry because of no directions.

7 Capacitance, Capacitors in Series and Parallel

7.1 Book Notes

- Any two conductors separated by an insulator (or a vacuum) form a capacitor.
- Don't confuse the symbol C for capacitance (which is always in italics) with the abbreviation C for coulombs (which is never italicized).
- Thus capacitance is a measure of the ability of a capacitor to store energy.
- The reciprocal of the equivalent capacitance of a series combination equals the sum of the reciprocals of the individual capacitances.
- The magnitude of charge is the same on all plates of all the capacitors in a series combination; however, the potential differences of the individual capacitors are not the same unless their individual capacitances are the same. The potential differences of the individual capacitors add to give the total potential difference across the series combination: $V_{\text{total}} = V_1 + V_2 + V_3 + \dots$
- The equivalent capacitance of a parallel combination equals the sum of the individual capacitances.
- The potential differences are the same for all capacitors in a parallel combination; however, the charges on individual capacitors are not the same unless their individual capacitances are the same. The charges on the individual capacitors add to give the total charge on the parallel combination: $Q_{\text{total}} = Q_1 + Q_2 + Q_3 + \dots$

7.2 Recitation

- The simplest capacitor is just two plates, both with a charge $\pm Q$ (they different).
 - Capacitance, by definition, has to be positive.
 - $C = \frac{\sigma A \epsilon_0}{\sigma d} = \frac{\kappa \epsilon_0}{d}$
 - * Only depends on the geometry.
- Does example from lecture, except with sphere..
 - Except specifies direction with $\hat{r}$
- With series, $Q_1 = Q_2$.
 - $\frac{1}{C_{eq}} = \frac{1}{C_1} + \frac{1}{C_2} + \dots + \frac{1}{C_n}$

- Opposite is true
 - $V_1 = V_2$
 - $C_{eq} = C_1 + C_2 + \dots + C_n$

9 Energy stored in Capacitors and Electric Fields; Dielectrics

9.1 Lecture Notes

- Charge goes up (conceptual example).
- Lets you apply higher voltages (so more charge).
- Lets you place the plates closer together (make d smaller).
- Increases the value of C because $\kappa > 1$.

9.2 Recitation

- Before the exam, we had a capacitance equation with a κ .
 - Suppose we disconnect from battery, so Q is constant.
 - Now put an insulator inside capacitor.
 - * Tada, there's your κ
 - * Insulator means electrons are stuck where they're at.
 - * $\kappa > 1$, usually.
 - This causes an electric field inside, causes a polarization.
- We can express the energy in three different forms, depending on what we know about the system.
- Energy conservation is inapplicable to these problems.

10 Electric Current; Current Density; Resistance

10.1 Book Notes

- An electric current consists of charges in motion from one region to another. If the charges follow a conducting path that forms a closed loop, the path is called an electric circuit.
- In electrostatic situations (discussed in Chapters 21 through 24) the electric field is zero everywhere within the conductor, and there is no current. However, this does not mean that all charges within the conductor are at rest.
- Although we refer to the direction of a current, current as defined by Eq. (25.1) is not a vector quantity. In a current-carrying wire, the current is always along the length of the wire, regardless of whether the wire is straight or curved. No single vector could describe motion along a curved path. We'll usually describe the direction of current either in words (as in "the current flows clockwise around the circuit") or by choosing a current to be positive if it flows in one direction along a conductor and negative if it flows in the other direction.
- Current density $\vec{J}$ is a vector, but current I is not — the difference is that the current density $\vec{J}$ describes how charges flow at a certain point, and the vector's direction tells you about the direction of the flow at that point.
- The greater the resistivity (ρ), the greater the field needed to cause a given current density, or the smaller the current density caused by a given field.

10.2 Lecture Notes

- Current goes down to milli-Amps — **Remember for test.**
- Current is a scalar — but can be negative.

10.3 Recitation

- Ohm's Law, in general form, $\vec{J} = \sigma \vec{E}$
- In this class, Ohm's law will just be current through a wire.
- We can assume the current density is proportional $\frac{I}{A}$
- The electric field should be uniform!
- $V = IR$ serves as our definition of resistance
- Some materials actually have a $-\alpha$

11 EMF; Electric Power

11.1 Book Notes

- For a conductor to have a steady current, it must be part of a path that forms a closed loop (or a complete circuit).
- A battery is not a current source — You might have thought that a battery or other source of emf always produces the same current, no what circuit it's used in. $\mathcal{E} - Ir = IR \implies I = \frac{\mathcal{E}}{R+r}$ says it isn't so. The greater the resistance R of the external circuit the less the current the source will produce.

11.2 Recitation

- Rate of energy conversion: $P = I\mathcal{E}$
- Energy lost $P = I^2r_1$
- Total power supplied $P = I\mathcal{E}_1 - I^2r_1$

12 Resistors in Series and Parallel, Kirchoff's Rules

12.1 Book Notes

For capacitors in series,

$$V_{ax} = IR_1 \quad V_{xy} = IR_2 \quad V_{yb} = IR_3 \quad (1)$$

$$V_{ab} = V_{ax} + V_{xy} + V_{yb} = I(R_1 + R_2 + R_3) \quad (2)$$

$$\frac{V_{ab}}{I} = R_1 + R_2 + R_3 \quad (3)$$

$$R_{eq} = R_1 + R_2 + R_3 \quad (4)$$

A similar argument can be made for resistors in parallel, except I is unknown, which add.

- A junction in a circuit is a point where three or more conductors meet. A loop is any closed conducting path.

13 1

13.1 Book Notes

13.2 Lecture Notes

13.3 Recitation

- The magnetic field is directly analogous to the electric field
 - However, the fields lines are circular (because no monopoles)
 - The force is cross product

Exam _num₁*QuickReference*

Illya Starikov

August 7, 2025

1 Prefix

Factor	Name	Symbol
10^{-1}	deci	d
10^{-2}	centi	c
10^{-3}	milli	m
10^{-6}	micro	μ
10^{-9}	nano	n
10^{-12}	pico	p
10^{-15}	femto	f

Test II Study Guide

Illya Starikov

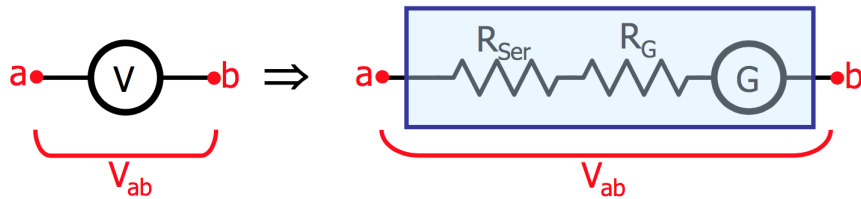
August 7, 2025

1 Notes

- If a dielectric fills one half of the space between plates, it's the same as two capacitors in parallel — one with a dielectric, one not.
- Current is in the direction of flow of **positive charge**.
- Materials that are ohmic have a linear I vs. V graph.
 - Anything else (like quadratic) are nonohmic.
- For resistors **in series**,
 - $R_{eq} = \sum_i R_i$ (opposite of capacitors)
 - Currents I is the same (same as capacitors)
 - V 's add (same as capacitors)
- For resistors **in parallel**,
 - $\frac{1}{R_{eq}} = \sum_i \frac{1}{R_i}$ (opposite of capacitors)
 - Currents I is the same (same as capacitors)
 - V 's add (same as capacitors)
- Power = $\frac{\text{Energy Transformed}}{\text{Time}}$
- Kirchhoff's Junction Rule: at any junction point, the sum of all currents entering the junction must equal the sum of all currents leaving the junction.

- Kirchhoff's Loop Rule: the sum of the changes of potential around any closed path of a circuit must be zero.

To reduce the percent error, the device being used as a voltmeter must have a very large resistance, so a voltmeter can be made from galvanometer in series with a large resistance.

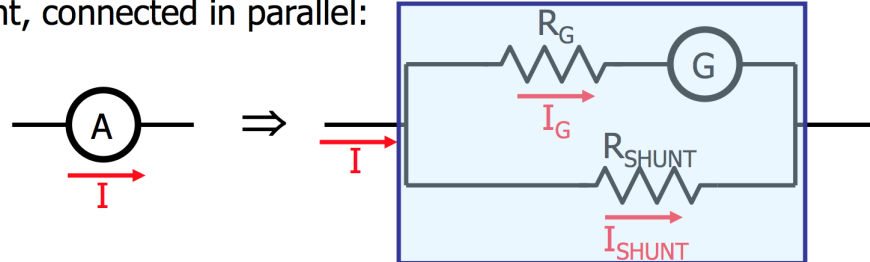


Everything inside the blue box is the voltmeter.

Homework hints: "the **galvanometer** reads 1A full scale" would mean a current of $I_G=1A$ would produce a full-scale deflection of the galvanometer needle.

If you want the **voltmeter** shown to read 10V full scale, then the selected R_{ser} must result in $I_G=1A$ when $V_{ab}=10V$.

A galvanometer-based ammeter uses a galvanometer and a shunt, connected in parallel:



Everything inside the blue box is the ammeter.

The resistance of the ammeter is

$$\frac{1}{R_A} = \frac{1}{R_G} + \frac{1}{R_{SHUNT}}$$

$$R_A = \frac{R_G R_{SHUNT}}{R_G + R_{SHUNT}}$$

- For charging a circuit, $I(t) = \frac{\varepsilon}{R} e^{-\frac{t}{RC}}$
- For discharging a circuit, $I(t) = I_0 e^{-\frac{t}{RC}}$
 - I_0 for charging is equal to I_0 for discharging only if the discharging capacitor was fully charged.
- In a series RC circuit, the same current I flows through both the capacitor and the resistor. Sometimes this fact comes in handy.
- Ohm's law **only** applies to resistors, not capacitors.
- Magnetic field lines point away from north, towards south.
 - Same notation as electric fields!
- In a uniform magnetic field, force is always radially outward. Therefore, $a = v^2/r$.

- Period $T = \frac{2\pi r}{v}$. (However, easier to remember distance = velocity · time $\implies T = \frac{\text{distance}}{\text{velocity}}$)
- frequency = $\frac{1}{T}$

2 Units

- Resistance: $1\Omega = \frac{1V}{1A}$
- Resistivity: $\rho = \Omega \cdot m$
- Magnetic field: $1T = \frac{1\text{kg}}{\text{C}\cdot\text{s}}$
- Older unit: $1\text{ G} = 10^{-4}\text{T}$

3 Unit Prefixes

10^{-15}	femto (f)
10^{-12}	pico (p)
10^{-9}	nano (n)
10^{-6}	micro (μ)
10^{-3}	milli (m)
10^{-2}	centi (c)
10^{-1}	deci (d)

PHYS2305

Modern Physics



Modern Physics Review

Illya Starikov

August 7, 2025

1 Special Relativity

From relativity, we know

1. All inertial reference frames are equivalent.
2. The speed of light is the same in all inertial reference frames.

From this, we notice that

1. Time and space depend on velocity ($\vec{v}$).
2. Moving clocks appear to run slow.
3. The length of a moving object, in the direction of motion, will appear shorter.

Eloquently, this can be described as

$$t = \gamma t_0$$

$$L = \frac{L_0}{\gamma}$$

Where $\gamma = \frac{1}{\sqrt{1-v^2/c^2}}$, t is the time according to the outside observer, t_0 to be proper time (i.e. the time measured by the moving object), L is length to outside observer, and L_0 is proper length.

2 Energy and Momentum

Relativistic energy and momentum can be defined as

$$E = \gamma m_0 c^2 \quad \vec{p} = \gamma m_0 \vec{v}$$

From the this, we can derive the more fundamental equation:

$$E^2 = m_0^2 c^4 + p^2 c^2 \quad (1)$$

For kinetic energy, we can eloquently describe it as $(\gamma - 1)m_0 c^2$, this is simply the rest mass energy ($m_0 c^2$) subtracted from the total energy ($\gamma m_0 c^2$). At low speeds (i.e. $v \ll c$), we can simply use a Taylor Series expansion to get $E \approx \frac{1}{2}m_0 v^2 + \frac{3m_0 v^4}{8c^2} + \dots$. Ignoring the other terms, we get $E \approx \frac{1}{2}m_0 v^2$, classical energy!

If we take m_0 to be 0, this implies $E = pc$ (From Equation 1). Using de Broglie wavelength ($\lambda = h/p$), this implies $E = h\nu$. This can be combined with the fact that $\lambda\nu = c$, which allows many permutations of the equations. We can also show that the kinetic energy $KE = \frac{p^2}{2m}$. To summarize, for $m_0 = 0$,

$$E = h\nu = \frac{hc}{\lambda} = pc$$

3 The Three Experiments

There were three experiments done to prove the quantum nature of light and particles.

3.1 The Photoelectric Effect

Suppose we have a sea of electrons within a metal. It takes some work to escape the sea, described by the work function Φ . We can relate the energy of the photon, the work function, and

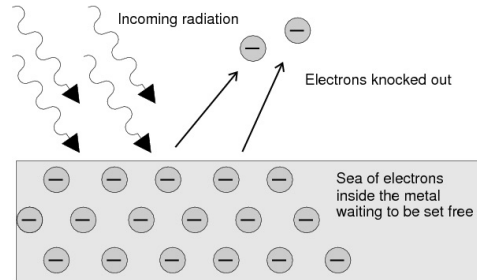


Figure 1 – The Photoelectric Effect.

the resulting energy of the electrons
by $h\nu = \Phi + KE_{\max}$.

3.2 Compton Scattering

We can scatter a photon off an electron, inelastically, and after some tedious math we can realize $\lambda_2 - \lambda_1 = \frac{h}{m_0c}(1 - \cos\theta)$, where λ_2 is the wavelength after scattering, λ_1 is the wavelength before the scattering, and m_0 the electron rest mass. From this we realize that there is a decrease in the energy of the photon, resulting in an increase of wavelength, which we know as the Compton effect.

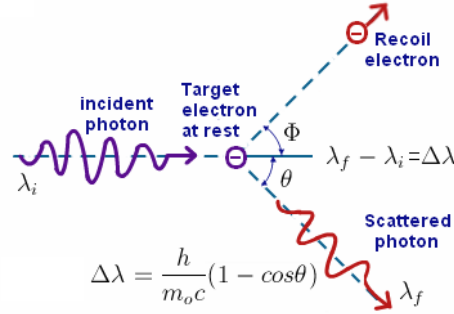


Figure 2 – Compton scattering.

3.3 Blackbody Radiation

Blackbody radiation refers to an object or system which absorbs all radiation incident upon it and re-radiates it. This effect can be characterized by the radiating system alone; it does not depend on the type of radiation incident upon it. The radiated energy can be considered to be produced by standing wave or resonant modes of the cavity which is radiating. The major effect of this is that the modes must be quantized. We can define the energy per unit volume per unit frequency

$$U(\nu) = \frac{8\pi\nu^2}{c^3} \frac{h\nu}{e^{\frac{h\nu}{kT}} - 1}$$

where k is the Boltzmann constant and T is the absolute temperature of the

body. The entity $\frac{h\nu}{e^{h\nu/kT}-1}$ is the average energy per mode, and $\frac{8\pi\nu^2}{c^3}$ counts the number of modes available.

4 Wave Nature of Massive Particles

$|\psi(x)|^2 dx$ = the probability of finding the
particle in the range of x to $x + dx$

Because $|\psi(x)|^2$ is a continuous probability distribution, we must normalize the wavefunction so that the probability has to add up to 100%,

$$\int_{-\infty}^{\infty} |\psi(x)|^2 dx = 1$$

4.1 Uncertainty Principle

It can be derived that for an particular measurement,

$$\Delta t \Delta E \geq \frac{\hbar}{2} \quad \hbar = \frac{h}{2\pi}$$

Or, more famously,

$$\Delta x \Delta p \geq \frac{\hbar}{2} \quad \hbar = \frac{h}{2\pi}$$

4.2 Infinite Potential Well

Imagine a potential wall where the wall go to infinity, with a distance L between the walls. Our wave function ψ must have the boundary conditions $\psi(x = 0) = 0$ and $\psi(x = L) = 0$. From this we realize we can only fit half-wavelengths of the wave into the box; that is, $n\lambda/2 = L$. To calculate the energy,

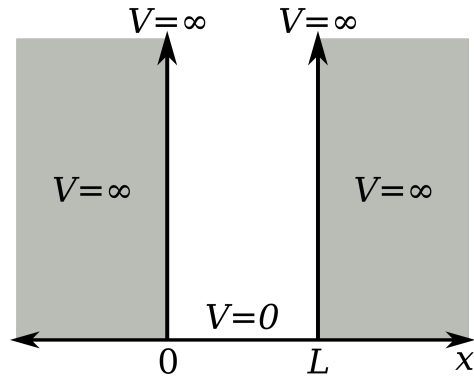


Figure 3 – Particle in a box (or infinite potential well).

$$E = \frac{p^2}{2m} = \frac{n^2 h^2}{8mL^2}$$

because $\lambda = \frac{2L}{n}$ and $p = \frac{h}{\lambda}$.

4.3 Wave Motion

We know we can describe just about any wave by $\cos(kx - \omega t)$, where $k = \frac{2\pi}{\lambda}$ and $\omega = 2\pi\nu$. Consequently, $p = h/\lambda = \hbar k$ and $E = h\nu = \hbar\omega$.

Furthering our wave mathematics, we can consider the phase velocity as

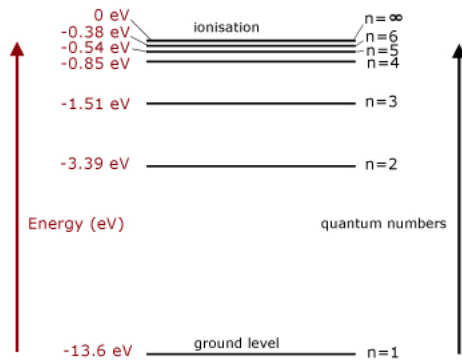
$$v_p = \frac{\omega}{k} = \frac{E}{p} = \nu\lambda$$

When considering a group, we can calculate the velocity of the group (or the packet), v_g , as

$$v_g = \frac{\partial\omega}{\partial k} = \frac{\partial E}{\partial p}$$

4.4 Hydrogen Atom

As we have proven with our three experiments, energy states are quantized. This implies that the energy levels in an atom come in integer levels (i.e. energy level $n = 1, 2, 3, \dots, \infty$, where ∞ is ionization).



From this, we can determine that the energy at any level $E_n = \frac{E_1}{n^2}$, where $E_1 < 0$. For photo-absorption,

$$\begin{aligned} h\nu &= E_f - E_i = \frac{E_1}{n_f^2} - \frac{E_1}{n_i^2} \\ &= E \left(\frac{1}{n_f^2} - \frac{1}{n_i^2} \right) \end{aligned}$$

Similarly, for emission,

Figure 4 – Energy levels of hydrogen atom.

$$\begin{aligned}
 h\nu = E_i - E_f &= \frac{E_1}{n_i^2} - \frac{E_1}{n_f^2} \\
 &= E \left(\frac{1}{n_i^2} - \frac{1}{n_f^2} \right)
 \end{aligned}$$

PSYC1101

Psychology 101

S&TTM

A Beautiful Mind

Illya Aleksandrovich Starikov

July 27, 2025

1 The Role of Psychology

The movie is centered around John Nash, who, as find out later in the story, has schizophrenia. In the remainder of this movie we are taken through Nash's life as he copes with his hallucinations and illusions.

2 Main Character

John Nash experiences an aggressive case of schizophrenia. His extraordinary intelligence (aside from his contribution to economics, he also worked on complex manifolds and the Riemann hypothesis, two very difficult areas of mathematics) causes him to experience very vivid episodes of hallucinations. A possible reasoning for the different characters is a coping mechanism — the roommate for his loneliness, the child for his fear of being a new father, the cryptography for having to do something great.

3 Psychological Treatment

John Nash first received regular treatments of *Insulin shock therapy*, in which he was repeatedly injected with insulin to induce comas. Because this is not a sustainable form of treatment outside of medical facilities, he was then given a regular prescription of an anti-psychotic. However, the side effects were too detrimental to Nash's mathematical work and his love life, he decides to mentally confront it. Instead of blocking it out, he decides to accept his hallucinations.

Famous People Extra Credit

Illya Aleksandrovich Starikov

July 27, 2025

1. Charles Darwin
2. Dorothea Dix
3. Sigmund Freud
4. G. Stanley Hall
5. William James
6. Ivan Pavlov
7. Jean Piaget
8. Carl Rogers
9. B. F. Skinner
10. Sigmund Freud

TEACH

Teaching Materials

S&TTM

Modern C++ (The Good Parts)

CS1570 — Introduction to Programming

Illya Starikov

July 27, 2025

Since its creation, C++ has gone through many different “versions” — more formally, known as standards. With `fg++`, you’ve been using C++03¹. Since then, there have been two major standards released: C++11 and C++14.

C++11 introduced many modern programming paradigms from other programming languages, and C++14 built on these principles. Namely, some of these features are type inference, lambda expressions, constant expressions, `default` and `deleted` keywords, and much more!

To compile C++11 or C++14 code, the `-std=c++11` or `-std=c++14` flags must be added (respectively). So, to compile with C++14, the command would be

```
g++ -std=c++14 *.cpp
```

1 Type Inference (`auto`)

Type inference is a way for a language to systematically “choose” the type you are trying to use. This can intuitively be thought of as you, the programmer, guessing the type of a math equation:

$$\text{answer} = 42 \tag{1}$$

Mathematically, we know `answer` to be an integer — subsequently, the compiler (GCC) should guess its type to be an `int`. Wouldn’t it be great if C++ could just magically do that? Well, now it can.

$$\text{auto } \text{answer} = 42 ; \tag{2}$$

The actual syntax is

¹Yes, that is “hella” old


```
auto variable = inference;
```

Now, we no longer have to type out those pesky return types (very useful if it's constantly changing). Let's do an example.

```
1 template <typename T>
2 T oneUp(const T value) {
3     return value + 1;
4 }
5
6 int main(int argc, char *argv[]) {
7     auto integer = oneUp(1);
8     auto character = oneUp('P');
9     auto string = oneUp("vs NP");
10    auto boolean = oneUp(false);
11 }
```

Compare `main` to it's pre-C++11/14 version.

```
1 template <typename T>
2 T oneUp(const T value) {
3     return value + 1;
4 }
5
6 int main(int argc, char *argv[]) {
7     int integer = oneUp(1);
8     char character = oneUp('P');
9     string std_string = oneUp("vs NP");
10    bool boolean = oneUp(false);
11 }
```

So there it is efficient for the programmer. How efficient is it for the program? Let's take another example.

```
1     char charArray[42] = "Hello";
2     auto length = strlen(charArray);
```

Can you guess the type of `length`? Spoiler alert, it's not `int`. It's actually `size_t`, a different type leftover from the C era. It's not vital knowing what `size_t` is or how it works, it's just vital knowing *it is not an* `int`. So, every time `strlen` it is used as an `int`, i.e.,

```
1     for (int i = 0; i < strlen(charArray); i++) {
2         /* do something here */
3     }
```

this has a performance cost. Why? `size_t` has to be downcast to an integer.

A note, in C++14, `auto` can be used as the return type of a function.

1.1 Range Based For Loops (`for (auto)`)

With `auto`, range based for loops are possible! What are these strange loops? They are just a way to iterate through all the element of a collection type (i.e., arrays).

```
1     auto coolTeachers = { "Illya", "Andrea", "Not Price" };
2
3     for (auto teacher : coolTeachers) {
4         cout << teacher << " ";
5     }
```

As you can guess, the output is “Illya Andrea Not Price”. Raaad. Some reasons it might be useful.

- Iterating through more complex data structures becomes much easier.
- It's more readable.
- You don't have to worry about proper indexing.

2 Lambda Expressions

Such a scary name for a scary topic. The simplest definition that will make thinking about them easier:

Functions as variables and types.

That's it. Now, the syntax:

```
[ capture ] ( parameters ) { functionBody }
```

For now, ignore the `capture` part. We will not be using it for this course. But now here is where shit gets crazy. Let's do an example.

```
1     std::function<bool(int, int)> lessThan = [](int x, int y) { return x <
2         y; };
```

First, notice the ugly return type `std::function<bool(int, int)>` — how can we get rid of it? `auto`! Second, it's not so bad! It sort of like a function but we are assigning it to a variable called `lessThan`.

```
1     auto lessThan = [](int x, int y) { return x < y; };
2
3     cout << lessThan(128, 256); /* Prints 1 for true */
```

How else did we use variables? Functions!

```

1  auto sort(std::function<bool(int, int)> compare, int array[], const int
    size) {
2      // This is just a bubble sort
3      for (int i = 0; i < size; i++) {
4          for (int j = 0; j < size - i - 1; j++) {
5
6              // Here's where we use compare
7              if (compare(array[j], array[j + 1])) {
8                  int temporary = array[j];
9                  array[j] = array[j + 1];
10                 array[j + 1] = temporary;
11             }
12         }
13     }
14 }
15
16 int main(int argc, char *argv[]) {
17     auto lessThan = [](int x, int y) { return x < y; };
18     int array[5] = { 1, 2, 3, 4, 5 };
19     sort(lessThan, array, 5);
20
21     for (auto element : array) {
22         cout << element << " ";
23     }
24 }

```

We can also return lambda expressions! Don't mind the `[]`, again it's an advanced feature. For this, all you have to know is it signifies the variable is "held" in temporary memory.

```

1  auto counter(const int start, const int incrementor) {
2      return [=]() {
3          static auto x = start;
4          x += incrementor;
5
6          return x;
7      };
8 }
9
10 int main(int argc, char *argv[]) {
11     auto count = counter(42, 2);
12
13     cout << count() << "\n" << count() << "\n" << count();
14     /* prints 44, 46, 48 */
15 }

```

We can use them exactly as we have used all variables — hold them in arrays

```

1  auto styleChecker = []() {
2      :
3      std::cout << "Style Checking...\n";
4  };
5  auto plagiarismChecker = []() {

```

```

6         :
7         std::cout << "Checking For Plagarism (Caught 2)...\n";
8     };
9     auto inputOutputChecker = []() {
10        :
11        std::cout << "Checking Input/Output...\n";
12    };
13
14    std::function<void()> graderStack[5] = { styleChecker,
15    plagiarismChecker, inputOutputChecker };
16
17    for (auto gradeFunction: graderStack) {
18        if (gradeFunction != NULL) {
19            gradeFunction();
20        }
21    }

```

Notice the checking of line `gradeFunction != NULL`. We have three functions in the array, but the size is five. If try to call the function, without a function in there, *it will cause a runtime error*. So be careful.

And finally, we can add them as variables to classes. I won't give an example here, because it's pretty straightforward.

3 Constant Expressions `constexpr`

`constexpr` is a way to move calculations from runtime to compile time. So, if you have a function that's going to compute **known** values ahead of time. So, if you're going to be calculating $\ln 2$ ahead of time often,

```

1  constexpr double ln(const int x) {
2      double sum = 0;
3
4      for (int n = 1; n <= 100; n++) {
5          sum += 1.0/n/(4*n - 2);
6      }
7
8      return sum;
9  }

```

Why might this be useful? Performance.

4 The `default` And `delete` Keywords

Remember when you implemented one constructor, and suddenly you lost the copy constructor and the assignment operator? Well, it took roughly 30 years to figure out

this was a pain in the ass for everyone. Now, if you overload the copy constructor, you can get the default constructor by prototyping it, then putting `= default`.

```
1 class Zombie {
2     string nameOfGradStudent;
3
4 public:
5     Zombie(const string name) {
6         nameOfGradStudent = name;
7     }
8
9
10 };
11
12 int main(int argc, char *argv[]) {
13     Zombie Fred; // compiler error
14 }
```

This won't compile, Fred doesn't have a default constructor. But, if did something like

```
1 class Zombie {
2     string nameOfGradStudent;
3
4 public:
5     Zombie(const string name) {
6         nameOfGradStudent = name;
7     }
8
9     Zombie() = default;
10 };
11
12 int main(int argc, char *argv[]) {
13     Zombie Fred; // compiler error
14 }
```

`delete` works the same way, except the opposite. Instead of gaining the default, it simply deletes the function. So, if there is no way possible to have a default constructor for the Zombie class, something like this is possible:

```
1 class Zombie {
2     string nameOfGradStudent;
3
4 public:
5     Zombie(const string name) {
6         nameOfGradStudent = name;
7     }
8
9     Zombie() = delete;
10 };
11
12 int main(int argc, char *argv[]) {
```

```
13     Zombie Fred; // compiler error
14 }
```

This is useful when you're creating a class for someone else to use. It makes the deletion of certain functionality more explicit.

5 Further Topics

If time permitted, some topics of further discussion:

- Preventing Exception Propagation (`noexcept`)
- Smart Pointers
- Move Semantics
- Variadic Templates
- There's no way we'd get here.

Template Functions, Function Overloading, Structs

Sannihitha Muppidi

October 11th, 2016

1 Problem Statement

As the title of this lab implies, the lab will cover template functions, function overloading, and structs (also a few other miscellaneous topics). This will be a longer lab, so please use time wisely. I recommend reading through the entirety of this lab, especially to the hints.

The scope of this lab is to create a simplistic attack-and-defend game. Two battleships start with the same health, but continuously attack each other until either one gets sunk or one flees from taking too much damage.

The implementation is laid out as follows.

Required Files

There will be *five* files to submit, namely

1. `main.cpp`
2. `battleship.h`
3. `functions.cpp`
4. `functions.h`
5. `templates.hpp`

And an optional constants file.

Required Structs

There will be one struct to be implemented, called `BattleShip`. It will have the following variables:

- A health variable of type `int`. This holds the health points (hp) of the ship.
- A defense variable of type `short`. It will be used when calculating relative damage.
- An attack variable of type `short`. It will also be used to calculate relative damage.

Required Functions

The following are all the *required* functions that must be implemented — this does not include any maintenance functions you so choose to have.

- `T randomNumber(const T from, const T to)` A **templated** random number generator (of typename `T`) that returns a random number `from...to` (inclusive). This **does not** work for floating point numbers, that is okay.
- `readyTheShip(BattleShip & ship, const int shipNumber)` This function prepares the ship for battle. Note that the
 1. Ask for input on what the name of the ship is.
 2. Assign health to a value of 100.
 3. Assign defense a *random* number 1...5. (1 through 5)
 4. Assign attack a *random* number 15...20. (1 through 5)
- `void attack(BattleShip & attacker, BattleShip & attackee)` The attackee ship loses health depending on the attack value. The attack value is a *random* value `attacker.defense...attackee.attack`. Output the damage.
- `bool attemptToFlee()` The flee is completely up to chance. At first the percentage chance is 15%, but with every successive attempt to flee, the change goes down a 1% until it's fixed at 1%.

- `void winner(const string winner)` Outputs a single winner.
- `void winner()` Outputs that there was a fleeing ship, no one wins!
- `int main()` Because there are several ways to play a game like this, there is provided pseudocode below. If you find a clever or cool different way to play, ask any of the student TAs or the TA if you are allowed to implement it.

```
ready ship one and two

while no one fled and no ship dies:
    ship two attacks ship one
    ship one attacks ship two

    if the health of ship one is < 20:
        ship one attempts to flee

    if the health of ship two is < 20:
        ship two attempts to flee

if someone flee,
    display flee message
else
    present the winner
```

Note that we do not consider ties, if two lose at the same time, arbitrarily choose the loser (I recommend choosing Price, if possible).

2 Submission

When using `cssubmit`, please use the following sample input and verify your solution with the sample output.

Sample Input

```
Jarus
Price
```

Sample Output

```
Jarus dealt 9 damage to Price
Price dealt 10 damage to Jarus
Jarus dealt 11 damage to Price
Price dealt 5 damage to Jarus
Jarus dealt 12 damage to Price
Price dealt 16 damage to Jarus
Jarus dealt 6 damage to Price
Price dealt 16 damage to Jarus
Jarus dealt 14 damage to Price
Price dealt 17 damage to Jarus
Jarus dealt 11 damage to Price
Price dealt 8 damage to Jarus
Jarus dealt 15 damage to Price
Price dealt 15 damage to Jarus
Jarus dealt 13 damage to Price
Price dealt 8 damage to Jarus
Jarus dealt 11 damage to Price
Price dealt 17 damage to Jarus

Jarus won!
```

3 Hints

- Compile early and compile often.
- If you don't know/aren't quite sure how to generate a random number $x_0 \dots x_n$, ask one of the TAs for a full explanation.
 - Make sure your random number generator is *inclusive*.
- There is a bare minimum of 6 constants in this assignment.

Lab #15: Vectors

Maria Bosco

Due Date: December 7th, 2016

It's holidays, and some of the Teaching Assistants and Professors decide to get together for a holiday extravaganza. Seeing as your star student, you decide to have them over. With the assistance of a newly-learned topic (vectors), your job is to write a C++ program to invite the guests, welcome them to your home, exchange gifts, and kick them out.

1 Preliminaries

Fundamentally, a vector is a C++ array. Meaning everything you did with an array to this point, you can do with a vector. After including the vector header (`#include <vector>`) and specifying a type (`vector<type>`, where `type` can be primitives like `char` or `double` or even classes like `string`). We see they're not much different from arrays.

```
1     vector<int> vector = { 1, 2, 3, 4, 5, 6 };
2     int array[6] = { 1, 2, 3, 4, 5, 6 };
3
4     /* Both print 2 */
5     cout << "Vector: " << vector[1] << " Array: " << array[1];
6
7     /* answer = 42 */
8     int answer = vector[5]*array[5] + vector[5];
9
10    /* walking off array */
11    answer = vector[42];
12    answer = array[42];
```

Note that we never specified a size; this is not a mistake. Vectors are **dynamic**, meaning as you use them, they grow larger. No more guessing

how large your array might have to be! So something like this is perfectly legal.

```
1     vector<double> someDoubles = { 1.1, 2.2, 3.3, 4.4, 5.5 };
2
3     someDoubles.push_back(6.6);
4     someDoubles.push_back(7.7);
5
6     for (int i = 0; i < rand(); i++) {
7         someDoubles.push_back(i + i/10.0);
8     }
```

We started with `someDoubles` being of size 5 and growing up to sizes greater than 10 000. Below I will list some of the more important parts of vectors. This won't be comprehensive; for more information you can check documentation [here](#), [here](#), or [here](#).

1.1 Accessing Elements

As we have already seen, accessing an element in a vector can be done with the accessor (the `[]` operator). Alternatively, you can use the `at(i)` method¹ instead.

In addition, there are two methods that might come in handy: `front()` and `back()`. They return the front element and the back element, respectively.

```
1     vector<string> assistants = { "Ian", "Illya", "Grant" };
2
3     cout << "The front assistant is " << assistants.front() << "\n"; // Ian
4     cout << "The back assistant is " << assistants.back() << "\n"; // Grant
5     cout << "The middle assistant is " << assistants.at(1) << "\n"; // Illya
```

1.2 Adding Elements

We have already seen `push_back(element)`, and for the purposes of this lab, this will suffice.

¹Method means member function; it's more object-oriented way of saying it.

```

1     vector<string> assistants;
2
3     assistants.push_back("Illya");
4     assistants.push_back("Grant");
5     assistants.push_back("Ian");
6
7     for (int i = 0; i < 3; i++) {
8         // prints Illya Grant Ian
9         cout << assistants.at(i) << " ";
10    }

```

1.3 Removing Elements

There are only two ways we are going to talk about removing elements: `pop_back()` and `erase()`. `pop_back()` is relatively straightforward; it deletes the last element (exact opposite of `push_back()`).

```

1     vector<string> assistants = { "Illya", "Ian", "Grant" };
2
3     /* Removes Grant from assistants. */
4     assistants.pop_back();
5
6     /* Array now contains [Illya, Ian] */

```

This is useful, but it doesn't allow for arbitrary deletion. To do this, we use `erase`; however, there is a catch. In true C++ fashion, you cannot say `erase(index)` or `erase(element)`. To delete *by index*, the syntax is `erase(vector.begin() + index)` where `vector` is the name of your vector variable.

```

1     vector<string> assistants = { "Illya", "Ian", "Grant" };
2
3     /* Erase Grant */
4     assistants.erase(assistants.begin() + 2);
5
6     /* Erase Ian */
7     assistants.erase(assistants.begin() + 1);
8
9     /* Erase Illya */
10    assistants.erase(assistants.begin());

```

Note the order is important! If we did this in reverse, we would get very different results.

```

1     vector<string> assistants = { "Illya", "Ian", "Grant" };
2
3     /* Erase Illya */
4     assistants.erase(assistants.begin());
5
6     /* Erase Grant */
7     assistants.erase(assistants.begin() + 1);
8
9     /* segfault! There's only one element, Ian */
10    assistants.erase(assistants.begin() + 2);

```

Last, we can clear all the contents of the array with `clear`.

```

1     vector<string> assistants = { "Illya", "Ian", "Grant" };
2     string newAssistant;
3     /* Erase Grant */
4     assistants.erase(assistants.begin() + 2);
5
6     for (int i = 0; i < 1000; i++) {
7         /* Assistants size is 1002 */
8         newAssistant = getInferiorTA();
9         assistants.push_back(newAssistant);
10    }
11
12    /* Assistant size is now 0 */
13    assistants.clear();

```

For deleting all occurrences, that's a more advanced topic. You can read about it [here](#).

1.4 Size and Max Size

In addition, there are two methods that are quite useful: `size()` and `max_size()`. So, if you had the following vector

1	2	3	4	5				

, calling `size()` would return 5, but `max_size()` would return 10. For the purposes of this lab, you should not need `max_size()`.

2 The Assignment

For this assignment, you will be required to implement four functions:

1. Invite Guests

2. Welcome Guests
3. Exchange Presents
4. Kick Out

Any maintenance or helper functions you might want/need are up to you.

2.1 Invite Guests

Invite guests should **take no input and return a vector of type string**.

First, print out a string to signify you are taking in guests, and have `q` stop the input. Do not stop until you have input two guests. Simply take in strings (you can assume no spaces, using `cout`) that signify the guest's names.

Because the postal service is.. super fast.. on the holidays, one person might not get the invitation in time. At random, decide if the invitation is lost² and print out that the person's invitation (`cout` the name) was lost, but the other x number of people got their invitation (`cout` the number of people that got the invitation). If the invitation didn't get lost, just output something like "Everyone got the invitation!".

Return the guests's name vector.

2.2 Welcome Guests

Welcome guests should **take in the guests vector and return a vector of type string**.

Similarly to invite guests, this will take also take input; however, for every guest, there will be one gift. Continuously take in strings to signify the present's names into its own vector *in the order of the guests*³. Return it after every guest has input their presents.

For iterating through all the guests, **you must use size**⁴.

²hint, hint: check the Hints section.

³As in, "Starikov"'s gift is "Socks" and "Ian"'s gift is a "Tesla", the arrays should be `Starikov` `Ian` and `Socks` `Tesla`

⁴As in, you can't pass in counters from input, no global variables, etc. You will solely rely on the vector's information size to iterate through the guests.

2.3 Exchange Presents

Exchange guests should **take in the guests vector and the presents vector by value, and not return anything**.

You will generate two random numbers, `from` and `to`, ranging from 0 to the number of presents $- 1$. You will use these values to determine the gift-giving relationship. `from` will be giving the gift, `to` will be receiving a gift. Afterward, erase `from` from the presents and gifts vector⁵.

And yes, you must use `size`.

2.4 Kick Out

Kick out should **take in the guests vector and the presents vector by reference, and not return anything**.

Announce you are kicking people out individually (iterating through all of them), and `clear`⁶ both the arrays.

2.5 Main

In main, simply call these functions in the order of

1. Invite Guests
2. Welcome Guests
3. Exchange Presents
4. Kick Out

so that the program acts as expected.

3 Deliverables

Please submit your file(s) with `cssubmit` using the provided sample input; you may check up against the sample output if you so wish.

⁵hint, hint: actually use `erase`

⁶hint, hint: use `clear`

3.1 Sample Input

```
q
Jarus
q
Price
Leopold
Wisely
Armita
Rhodes
q
Beer
Cow
AppleWatch
MoreBeer
Nothing
HandGrenade
```

3.2 Sample Output

```
Wisely gave Jarus a MoreBeer
Rhodes gave Armita a HandGrenade
Armita gave Price a Nothing
Leopold gave Jarus a AppleWatch
Jarus gave Price a Beer
Price gave Price a Cow
You kicked Jarus out!
You kicked Price out!
You kicked Leopold out!
You kicked Wisely out!
You kicked Armita out!
You kicked Rhodes out!
```

4 Hints

- Because it's not the point of the lab, random value generation is given. Don't forget `srand(time(NULL))` at the top of main and the relevant

header files.

```
1  template<typename T, typename S>
2  T randomArbitrary(const T lower, const S upper) {
3      return lower > upper ? 0 : lower + rand() % (upper -
4  lower + 1);
5  }
```

- `randomArbitrary(false, true)` will return either true or false; however it must be in the order `false, true`.
- The solution contains 58 lines of C++ code and 16 lines of template/header code (excluding comments, blank lines, etc.). If your solution is significantly more (or less!), there may be a problem.

Happy Holidays from the Computer
Science Department!

Homework #10 Grading Rubric

Introduction To Programming

1 Partners

Author #1

Author #2

2 Grading Rubric

Description	Points Possible	Grade #1	Grade #2	Notes
The program works as expected. All classes are properly implemented. Milhouse is chased by phantom pants. Bullies are scattered. Milhouse doesn't run over stuff. A clean, 17x17 grid is displayed. Colors are required.	40			
The program is semantically correct. No segmentation faults, infinite loops, program properly terminates, etc.	20			
Proper use of object oriented programming. Use of abstraction and encapsulation. Implementation details are marked private, vital functionality is marked public.	10			
Proper documentation. Preconditions, postconditions, and description are articulated.	10			
Style guide is adhered to. Braces are on new lines, no tabs are used, constants are upper-cased, etc.	10			
Presentation. Prepared for presentation, able to properly answer questions, brought snacks.	10			

3 Additional Questions

- What extra things did you implement?
- What was the work split? (For this, **you do not have to be honest**. If one person did all the work, and you want equal credit, say equal work. However, keep in mind I will grade more leniently those who did more work.)
- How did you handle Milhouse getting randomly placed on the map?
- How did you handle the phantom pants chasing Milhouse?
- How are your classes structured? As in, how did you choose what was public or private?
- Did issues did you run into during coding this assignment?

4 Notes

iOS Development With Swift 4

Academy of Computing Machinery

Illya Starikov

What We'll Be Covering

1. The Model View Controller (MVC) Paradigm
2. Swift 4
3. Xcode
4. Interface Builder

Massive View Controller I

Err, Model View Controllers*

Model An object representing the data or user activity.

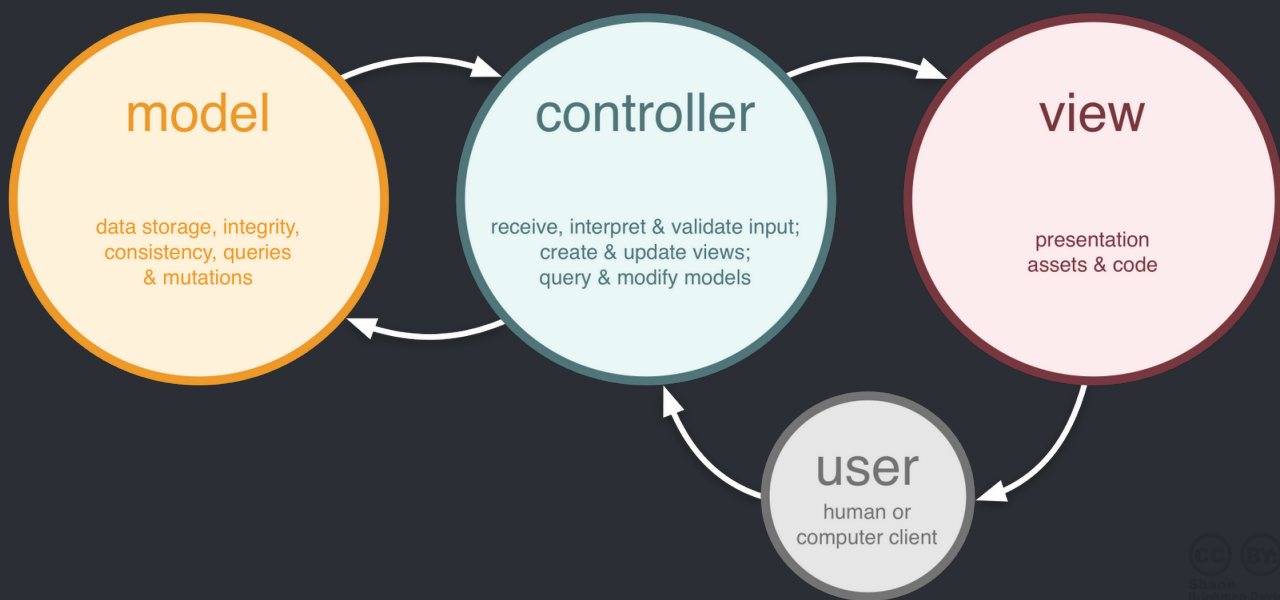
View A visual representation of the model, for user interaction.

Controller The interface between the model and the view.

Note the model should never talk to the view, and vice versa. Everything goes through the controller.

View Controller II

A more visual explanation



Swift 4

A brief introduction

```
print("Hello, Swift")
```


Swift 4

A brief Introduction

- Swift is a modern programming language with modern programming paradigms.
 - Closures
 - Tuples/multiple return values
 - First class functional programming support (i.e. , map, reduce, filter)
- Swift is open source, with all contents on available on <https://github.com/apple/swift>.
- Source compatible from Swift 3.
- Designed for safety (i.e. , automatic memory management (via reference counting), overflow checking, everything is utilized before use).

Variables

Swift Syntax I

- Variables in swift are declared with the keyword `var`.
- Constants are declared with the keyword `let`.
 - All variables that do not change *should be marked as constant*.
 - There are compiler warning for not using constants.
- Type inference is not only supported, but highly recommended.
- Swift is a *strongly typed* language.
 - This means when you declare a type, it stays that type.
 - Arithmetic operations should be performed on the same type. (i.e., `Int × Int` is okay, but `Int × Double` is not.)
 - Although Swift does support the idea of casting, it is often better to use constructors to get a new variable of desired type.

Variables (Example)

Swift Syntax I

```
let pi = 3.14 // Double
```

```
var answer = 420 // Int
```

```
answer /= 10 // oops
```

```
var piIsAnswerInt: Int // explicit type
```

```
piIsAnswerInt = Int(pi) * answer
```

```
print(piIsAnswerInt) // prints 126
```

Optionals

Swift Syntax II

Optionals are another type in Swift. There are two cases for optionals, either they are `nil` or a value. The syntax for an optional is `type?`, with the question mark signifying the optional. To unwrap an optional, i.e., get its value, the syntax is `variable!`.

*Unwrapping optionals **will cause a runtime error.***

Optionals (Example)

Swift Syntax II

```
var ACMPresident: String? = "Dalton Cole"

// this is unsafe, causes several warnings
print(ACMPresident) // prints Optional("Dalton Cole")

ACMPresident = nil
print(ACMPresident) // prints nil

print(ACMPresident!) // causes runtime error

ACMPresident = "Eric Michalak"
print(ACMPresident!) // prints Eric Michalak
```

Optionals

Swift Syntax III

Optionals are so prevalent in Swift, there is even special syntax for unwrapping them, `if let`. Most optionals will get unwrapped this way.

```
var ACMPresident: String? = nil
```

```
if let president = ACMPresident {  
    print("This current president is \(president).")  
} else {  
    print("ACM Elections are probably underway.")  
}
```

Data Structures

Swift Syntax IV

Swift supports all basic data structures you might need:

- Strings (`String`)
- Arrays (`Array<Character>`)
- Sets (`Set<Int>`)
- Dictionaries (`Dictionary<Int>`)
- Tuples (`Tuple<Int>`)

Enumerated Types (Enums)

Swift Syntax V

Enumerated types are just like like enums in other languages.

```
enum Direction {  
    case north  
    case south  
    case east  
    case west  
}
```


Enumerated Types (Example)

Swift Syntax VI

But because of Swift's type inference, enums have some brevity to them.

```
let northPoleDirection = Direction.north

if northPoleDirection == .north {
    print("Hello, Cold.")
} else if northPoleDirection == .south {
    print("When did penguins get here?")
} else {
    print("This definetly can't be right.")
}
```

Control Flow

Swift Syntax VII

Swift supports all paradigm of control flow, along with more modern loop methods.

- If-Else with `if...else`
- Switch-Cases with `switch { case ...: }`. Note that *switches must be exhaustive*, as in all switch cases must either go through all possible values or have a default.
- For-In loops via `for ... in ...`.
- Enumerated for loops via `for index in start...end`, where start is the start index and end is the end index (inclusive).
- While loops with `while`.
- Do-While loops with `repeat...while`

Control Flow (Example)

Swift Syntax VII

```
let answer = 42
```

```
if answer == 42 {  
    // prints "The answer to the question of life,  
    the universe and everything is 42"  
    print("The answer to the question of life, the  
    universe and everything is \(answer)")  
}
```

```
for i in 1...answer {  
    print(i) // prints 1 to 42  
}
```

Control Flow (Example)

Swift Syntax VII

```
let bestPokemon = ["pikachu", "wobbuffet", "mewtwo"]

for pokemon in bestPokemon {
    print(pokemon) // prints pikachu, wobbuffet,
    mewtwo
}
```

Control Flow (Example)

Swift Syntax VII

```
for pokemon in bestPokemon {  
    switch pokemon {  
        case "pikachu":  
            print("Pika Pika")  
  
        case "wobbuffet":  
            print("WOBBUFFET!")  
  
        default:  
            print("Pokedex 404")  
    }  
}
```

Functions

Swift Syntax VIII

Functions have the following syntax

```
func function(external internal: Type) ->  
    return values {  
    ...  
    return 42  
}
```

Where **external** and **internal** are the parameter names. Internal parameter names are used inside the function, while external parameter names are used when calling the function. This will make more sense in a second.

Functions (Example)

Swift Syntax VIII

```
func calculateRadian(fromAngle angle: Int) -> Double
{
    let pi = 3.14
    return Double(angle)*(pi/180.0)
}
```

```
let radian = calculateRadian(fromAngle: 180)
```

Closures

Swift Syntax IX

Closures are just functions that act like variables. Because they act like variables, they can get additional data from the environment (this will become more clear in the example). Just as variables have types, so do closures: (parameter types) -> return type. The general syntax of a Swift closures is

```
{ (parameters) -> return type in  
    ...  
}
```


Closures (Examples)

Swift Syntax IX

Take the following function.

```
func createCounter(startingAt start: Int,  
    incrementing incrementor: Int) -> () -> Int {  
    var value = start  
  
    return { () -> Int in  
        value += incrementor  
        return value  
    }  
}
```

Closures (Examples)

Swift Syntax IX

It's a function that takes in a start value and how much to increment by, and returns a closure (or, another function) that increments by the value passed in. So it would be used like so.

```
let counter = createCounter(startingAt: 42,  
    incrementing: 3)  
  
for _ in 1...5 {  
    print(counter()) // prints 45, 48, 51, 54, 57  
}
```

Note the underscore just represents that the program doesn't require that value.

Classes and Structs

Swift Syntax X

Swift is a purely object-oriented language, naturally classes and structs are present. Classes and structs adhere to the four pillars of object oriented programming:

1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

The only difference between structs and classes is:

struct Value type. All instances are copied when being passed around.

class Reference type, i.e., pass by reference. A pointer to all instances.

Classes and Structs

Swift Syntax X

Some notes on classes and structs.

- Classes require all values to be initialized upon instantiation, whether it be through the constructor or default property values. Structs are not required to do this.
- Both classes and structs can have properties and methods.
- Structs and classes have very different rules on initializers.

Classes and Structs (Example)

Swift Syntax X

```
class Zombie {  
    var health = 100  
    var attackBoost = 2 // valued from 1-5  
    var defenseBoost = 1 // valued from 1-5  
  
    func getAttackValue() -> Int {  
        return 4 // chosen by fair dice roll  
    }  
  
    func attack(_ zombie: Zombie) {  
        zombie.health -= getAttackValue() * attackBoost  
    }  
}
```

Classes and Structs

Swift Syntax X

Everything is familiar, even using this `Zombie` object is similar to most programming language.

```
let rick = Zombie(), morty = Zombie()  
rick.attack(morty)
```

However, this isn't good programming practice. Some things should be marked as `private`, and some `public`. In swift `public` and `private` are reserved words to mark things as public and private. In Swift, getters and setters get first class support with computed properties.

Classes and Structs (Public & Private)

Swift Syntax X

```
class Zombie {  
    var health = 100  
    private var attackBoost = 2 // valued from 1-5  
    private var defenseBoost = 1 // valued from 1-5  
  
    private func getAttackValue() -> Int {  
        return 4 // chosen by fair dice roll  
    }  
  
    public func attack(_ zombie: Zombie) {  
        zombie.health -= getAttackValue() * attackBoost  
    }  
}
```

Classes and Structs (Computed Properties)

Swift Syntax X

```
class Zombie {  
    private var myHealth = 100  
  
    public var health: Int {  
        get {  
            return myHealth  
        }  
    }  
    ...  
    public func attack(_ zombie: Zombie) {  
        let damage = getAttackValue() * attackBoost  
        zombie.take(damage: damage)  
    }  
  
    public func take(damage: Int) {  
        myHealth -= damage  
    }  
}
```


Classes and Structs (Constructors)

Swift Syntax X

As in most programming language, Swift has different rules about different constructors. Very rarely will writing a new constructor be necessary, but to write a convenience constructor. This has the syntax:

```
convenience init(parameters) {  
    self.init()  
    ...  
}
```

Classes and Structs (Example)

Swift Syntax X

```
class Zombie {  
    private var attackBoost = 2 // valued from 1-5  
    private var defenseBoost = 1 // valued from 1-5  
  
    convenience init(attackBoost: Int, defenseBoost: Int)  
    {  
        self.init()  
  
        self.attackBoost = attackBoost  
        self.defenseBoost = defenseBoost  
    }  
    ...  
}
```

Classes and Structs (Example)

Swift Syntax X

So now there is the default constructor, and the convenience initializer.

```
let rick = Zombie(attackBoost: 5, defenseBoost: 1)
let morty = Zombie()
```

```
rick.attack(morty)
```

The Rest of Swift

Swift Syntax XI

This was just a brief introduction to Swift — it still has many more powerful features:

- Type Casting
- Nesting Type
- Extensions
- Protocols
- Generics

To learn more about these features, and to go more in depth about the previous talking points, check out The Swift Programming Language on [Apple's Developer Site](#).

In Closing

Spam Illya with all questions, comments, and insults.

 @IllyaStarikov

 IllyaStarikov

 starikov@mst.com

Credits

- Model View Controller graphic courtesy of Basics of Web Development at <https://basicsofwebdevelopment.wordpress.com>