Missouri S&T

Curated Portfolio

Illya Starikov

Selected Coursework and Teaching Materials

Bachelor of Science, Computer Science

Class of 2018

Preface

This carefully selected portfolio contains teaching materials and educational resources developed during my time as a teaching assistant at Missouri University of Science and Technology. These materials represent my commitment to knowledge transfer and my ability to distill complex technical concepts into accessible learning experiences for fellow students.

The resources presented here demonstrate both pedagogical understanding and technical expertise, showcasing how I've contributed to the academic community through structured labs, comprehensive demonstrations, and evaluation rubrics. This collection reflects my belief that true mastery of a subject comes not just from learning, but from the ability to effectively teach and mentor others in their educational journey.

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

Portfolio Overview

This curated portfolio showcases both academic coursework and teaching materials from my time at Missouri S&T. The collection includes materials I developed as a teaching assistant and guest lecturer, alongside significant assignments, examinations, and projects that demonstrate mastery across computer science, mathematics, and engineering disciplines.

Teaching Materials

The following materials were developed for instructional purposes as a teaching assistant and guest lecturer:

Modern C++ Lecture

A comprehensive lecture introducing CS1570 students to modern C++ features (C++11/14). This document covers advanced programming paradigms including type inference with `auto`, lambda expressions, constant expressions, and modern best practices. Designed to bridge the gap between traditional C++03 taught in introductory courses and contemporary industry standards.

Lab 8: Template Functions

A comprehensive lab assignment for CS1570 introducing template programming and the classic Battleship game implementation. Students learn generic programming concepts through function templates, develop a complete game with board management and AI opponents, and practice modular design with multiple compilation units. Features creative narrative elements and progressive difficulty levels.

Lab 15: Vectors

An engaging holiday-themed lab assignment for CS1570 focusing on dynamic data structures using C++ vectors. Students implement a party simulation where they manage guest lists, gift exchanges, and social interactions using vector operations. This lab reinforces object-oriented programming concepts while introducing STL containers in a creative context.

CS1570 Homework 10 Grading Rubric

A detailed grading rubric for CS1570's Homework 10, demonstrating assessment methodology for programming assignments. The rubric evaluates functional correctness, semantic accuracy, object-oriented design principles, documentation quality, coding style adherence, and presentation skills. Includes sections for partner collaboration and reflective questions about implementation challenges.

Git Tutorial for Missouri S&T Satellite Team

A comprehensive Git version control tutorial developed for the Missouri S&T Satellite Team. Includes interactive slides covering fundamental Git concepts, branching strategies, collaborative workflows, and best practices for aerospace software development. Features hands-on demonstrations with a Tic-Tac-Toe game implementation showing practical Git usage patterns for team projects.

iOS Development With Swift 4

A comprehensive ACM presentation introducing iOS development using Swift 4. Covers the Model-View-Controller (MVC) paradigm, Swift syntax fundamentals including variables, optionals, data structures, enums, control flow, functions, closures, and classes/structs. Features practical code examples, object-oriented programming concepts, and guidance on modern Swift development practices. Developed for the Academy of Computing Machinery at Missouri S&T.

11.2 Polar Coordinates Lecture

Personal lecture notes from when I served as a guest lecturer for Calculus II, covering the introduction and application of polar coordinates. The material includes coordinate system definitions, conversion between Cartesian and polar forms, graphing techniques, and practical applications in physics and engineering. These notes reflect my teaching approach emphasizing visual understanding and real-world connections.

Academic Coursework

The following documents represent significant assignments, examinations, and projects completed as a student, demonstrating comprehensive understanding across multiple disciplines:

CS5402 Data Mining Test 1

Solutions and analysis for Test 1 in graduate-level Data Mining course. Covers fundamental data mining concepts including data preprocessing, correlation measures, OLAP operations (slice, dice, roll-up, drill-down), and the 1-R classification method. Demonstrates understanding of ROC curves, attribute selection, and data warehouse operations with practical examples.

CS5400 AI Exercise 2

Problem formulation and solution for the classic Missionaries and Cannibals river-crossing puzzle. This exercise demonstrates state space representation, constraint satisfaction, and search problem formulation. Includes formal mathematical notation for state descriptions and transition rules, showcasing systematic problem-solving approaches in AI.

CS5400 AI Exercise 4

Search algorithm implementation and analysis exercise focusing on uninformed search strategies. Demonstrates practical application of search algorithms with complexity analysis and performance evaluation. Includes problem-solving using various search techniques in artificial intelligence contexts.

CS5200 Homework 8

Advanced algorithm analysis assignment covering probability theory applications (Bayes' Theorem), binary heap properties, and efficient k-way merge algorithms. Demonstrates mathematical rigor in analyzing algorithmic complexity and probability calculations, including the famous prisoner paradox problem.

CS5200 Final Exam

Comprehensive final examination for graduate Analysis of Algorithms course. Features rigorous mathematical proofs using Big-O notation definitions, asymptotic analysis without limits, and detailed algorithmic complexity demonstrations. Showcases deep understanding of theoretical computer science foundations.

CS3800 Exam II Cheatsheet

Concise reference sheet for Operating Systems Exam II, formatted for maximum information density. Covers advanced OS concepts in a landscape format, optimized for quick reference during examinations. Demonstrates ability to synthesize and organize complex technical information effectively.

CS2300 Databases Final Report

Comprehensive database system design and implementation report for ForFit, a fitness tracking application. Includes complete database design manual with conceptual, logical, and physical design phases, along with detailed user manual. Features EER diagrams, relational schemas, application program design, and extensive testing procedures. Demonstrates end-to-end database project development from problem statement to deployment.

CS2200 Theory of Computer Science Quiz 10

Solutions for context-free grammar derivations and language theory problems. Demonstrates understanding of formal language theory including production rules, derivation sequences, and language characterization. Shows mastery of theoretical computer science concepts including grammar transformations and formal language specifications.

CS1510 Data Structures Heap

Comprehensive notes on heap data structure implementation and operations. Covers binary heap properties, percolation algorithms, and array-based heap implementation in C++. Includes visual tree representations and detailed code examples for insertion and removal operations with complexity analysis.

CS1200 Discrete Math Test II Study Guide

Student-prepared comprehensive study guide for Discrete Mathematics Test II. Covers sequences, mathematical induction, recursion, and set theory with formal definitions and proofs. Includes summation formulas, binomial coefficients, geometric sequences, and element argument proof methods. Demonstrates ability to synthesize course material into effective study resources.

CPE3150 Homework 8

Embedded systems programming assignment for the 8051 microcontroller. Features four problems demonstrating timer/counter operations, interrupt handling, and waveform generation. Includes C and assembly language implementations showcasing frequency generation for 5 kHz and 500 Hz square waves with precise duty cycle control using timer interrupts.

CPE3150 Arduino Research Paper

Research paper on "Paradigms and Applications of the Arduino Uno" examining the microcontroller's impact across education, healthcare, and home automation industries. Discusses Arduino's open-source philosophy, technical specifications, community ecosystem, and practical implementations including PID controllers and remote laboratory systems for engineering education.

CPE3150 Project 3: Space Invaders

Team project report implementing a retro Space Invaders game on embedded hardware with custom peripherals. Features terminal-based graphics using a custom 8051 curses implementation, music synthesis through timer interrupts, interactive menu system with Penrose triangle visualization, seven-segment display integration, and keyboard input for real-time gameplay. Demonstrates creative problem-solving for memory constraints and serial communication challenges.

Physics II Lecture 3: Electric Fields

Comprehensive lecture notes covering electric field lines, electric dipoles, electric flux, and Gauss' Law. Includes detailed explanations of dipole moments, torque calculations, and the fundamental relationship between electric flux and enclosed charge. Features both book notes and recitation examples demonstrating practical applications of field theory and surface integrals in electrostatics.

Modern Physics Review

Comprehensive review document covering special relativity, energy-momentum relationships, and quantum phenomena. Includes derivations of time dilation and length contraction, relativistic energy equations, and detailed explanations of the photoelectric effect, Compton scattering, and de Broglie wavelength. Features mathematical proofs connecting classical and quantum mechanics through limiting cases.

Philosophy 3235 Test 2: Business Ethics

Examination covering utilitarian and Kantian ethical frameworks with practical applications. Analyzes the fundamental differences between universal and maximum benefit principles through thought experiments including the transplant problem and truth-telling scenarios. Features analysis of autonomous vehicle ethics through the MIT Moral Machine experiment, demonstrating real-world applications of ethical decision-making in technology.

TEACH

Teaching Materials

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++03[1]. 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.

$$\texttt{auto}\ \text{answer} = 42\ \texttt{;} \tag{2}$$

The actual syntax is

[1] Yes, 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 <
     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!

```
auto sort(std::function<bool(int, int)> compare, int array[], const int
    size) {
    // This is just a bubble sort
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size - i - 1; j++) {

            // Here's where we use compare
            if (compare(array[j], array[j + 1])) {
                int temporary = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temporary;
            }
        }
    }
}

int main(int argc, char *argv[]) {
    auto lessThan = [](int x, int y) { return x < y; };
    int array[5] = { 1, 2, 3, 4, 5 };
    sort(lessThan, array, 5);

    for (auto element : array) {
        cout << element << " ";
    }
}
```

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.

```
auto counter(const int start, const int incrementor) {
    return [=]() {
        static auto x = start;
        x += incrementor;

        return x;
    };
}

int main(int argc, char *argv[]) {
    auto count = counter(42, 2);

    cout << count() << "\n" << count() << "\n" << count();
    /* prints 44, 46, 48 */
}
```

We can use them exactly as we have used all variables — hold them in arrays

```
    auto styleChecker = []() {
        ⋮
        std::cout << "Style Checking...\n";
    };
    auto plagarismChecker = []() {
```

```
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,
        plagarismChecker, inputOutputChecker };
15
16      for (auto gradeFunction: graderStack) {
17          if (gradeFunction != NULL) {
18              gradeFunction();
19          }
20      }
```

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? Welp, 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 $1 \ldots 5$. (1 through 5)
  4. Assign attack a *random* number $15 \ldots 20$. (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 fleed 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 comprehensize; for more information you can check documenation 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)` method[1] 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.

[1] Method means member function; it's more object-oriented way of saying it.

```
vector<string> assistants;

assistants.push_back("Illya");
assistants.push_back("Grant");
assistants.push_back("Ian");

for (int i = 0; i < 3; i++) {
    // prints Illya Grant Ian
    cout << assistants.at(i) << " ";
}
```

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()`).

```
vector<string> assistants = { "Illya", "Ian", "Grant" };

/* Removes Grant from assistants. */
assistants.pop_back();

/* 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.**

```
vector<string> assistants = { "Illya", "Ian", "Grant" };

/* Erase Grant */
assistants.erase(assistants.begin() + 2);

/* Erase Ian */
assistants.erase(assistants.begin() + 1);

/* Erase Illya */
assistants.erase(assistants.begin());
```

Note the order is important! If we did this in reverse, we would get very different results.

```
vector<string> assistants = { "Illya", "Ian", "Grant" };

/* Erase Illya */
assistants.erase(assistants.begin());

/* Erase Grant */
assistants.erase(assistants.begin() + 1);

/* segfault! There's only one element, Ian */
assistants.erase(assistants.begin() + 2);
```

Last, we can clear all the contents of the array with `clear`.

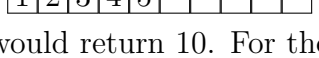
```
vector<string> assistants = { "Illya", "Ian", "Grant" };
string newAssistant;
/* Erase Grant */
assistants.erase(assistants.begin() + 2);

for (int i = 0; i < 1000; i++) {
    /* Assistants size is 1002 */
    newAssistant = getInferiorTA();
    assistants.push_back(newAssistant);
}

/* Assistant size is now 0 */
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 

, 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 lost[2] 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].

[2] hint, hint: check the Hints section.

[3] As in, "Starikov"'s gift is "Socks" and "Ian"'s gift is a "Tesla", the arrays should be |Starikov|Ian| and |Socks|Tesla|

[4] As 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 vector[5].

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.

[5] hint, hint: actually use `erase`

[6] hint, 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 -
     lower + 1);
4    }
```

- `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 you 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 uppercased, 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

Fundamentals of Git

Missouri Satellite Team

Presented by: Illya Starikov

Getting Started

There'll be some setting up before you can start using git. It'll depend on your operating system — you can refer to the installation guide here. Below are the more popular methods of installation.

macOS `brew install git`[1] or installing command lines tools.

Linux Depends on your distro. If Ubuntu use `sudo apt-get install git-all`, if Arch Linux then `pacman -S git`, if others refer here.

Windows Download a `.exe` from here.

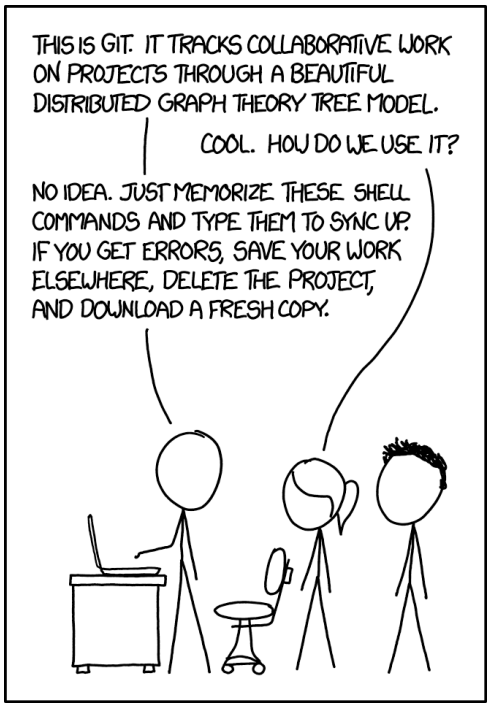
Getting Started

If you would like to follow along, navigate to https://github.com/IllyaStarikov → repositories → msat-git-tutorial → clone or download[2]. From here, you can either download as you normally would, or you use the super-cool terminal way:

1. Copy the link provided (something like https://github.com/IllyaStarikov/msat-git-tutorial.git).
2. Open your terminal emulator of choice.
3. Navigate to desired directory (using `cd DIRECTORY`)
4. Type in `git clone LINK`, where link is the copied link from Step #1.

What is Git?

- Git is nothing more than Directed Acyclic Graph of objects compressed and identified by an SHA-1 hash.
- Git works in snapshots, not differences.
- Git is local.
- Git has data integrity.
- Git is parallelizable.



The Five Stages of Git

1. Working Directory
2. Staging Area
3. Git Directory
4. ...
5. Profit

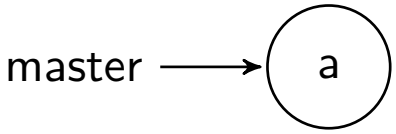
`Git`ting Good

If the "stages" didn't make sense, that's alright. It's better to go through a workflow apposed to the formalities.

1. Start a new repository with `git init`
2. Work on project in bite sized chunks, and add files that were changed with `git add file(s)`
3. Commit your changes with `git commit`
4. ...
5. Profit.

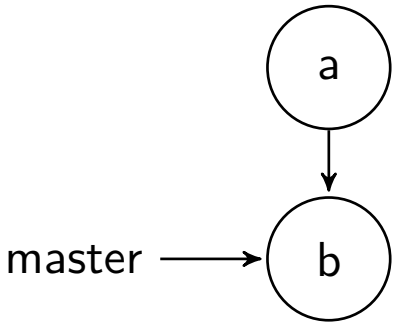
Committing I

```
git commit -m ``a''
```



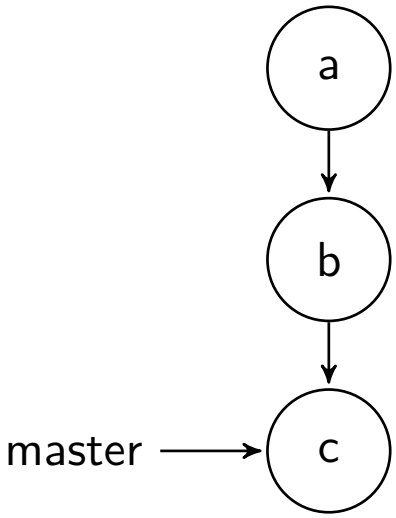
Committing II

```
git commit -m ``b''
```



Committing III

```
git commit -m ``c''
```



The Commit Message

This is the easiest part of git — and also the easiest to mess up. Here are seven rules of a great commit message.

1. Separate subject from body with a blank line.
2. Limit the subject line to 50 characters.
3. Capitalize the subject line.
4. Do not end the subject line with a period.
5. Use the imperative mood in the subject line.
6. Wrap the body at 72 characters.
7. Use the body to explain *what* and *why* vs. *how*.

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

The Commit Message (Example)

```
Summarize changes in around 50 characters or less

More detailed explanatory text, if necessary. Wrap it to about 72
characters or so. In some contexts, the first line is treated as the
subject of the commit and the rest of the text as the body. The
blank line separating the summary from the body is critical (unless
you omit the body entirely); various tools like `log', `shortlog'
and `rebase' can get confused if you run the two together.

Explain the problem that this commit is solving. Focus on why you
are making this change as opposed to how (the code explains that).
Are there side effects or other unintuitive consequences of this
change? Here's the place to explain them.

Further paragraphs come after blank lines.

 - Bullet points are okay, too

 - Typically a hyphen or asterisk is used for the bullet, preceded
   by a single space, with blank lines in between, but conventions
   vary here
```

Demo

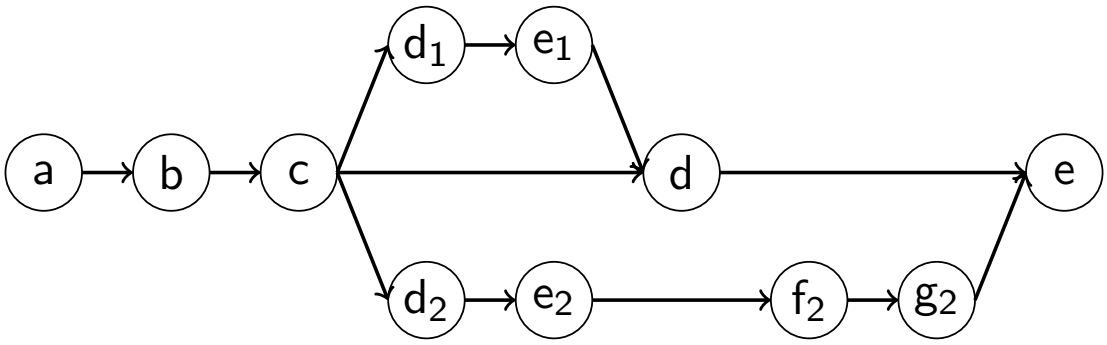
`Git`ting Better

Some more advanced commands to make your job easier.

- The wildcard `*` expands to whatever can fit a certain pattern.
  - `git add *.cpp` stages all files with a `cpp` extension.
  - `git add damon.*` add all document types with the name of `damon`, whether it be `cpp`, `txt`, or (unfortunately) `jpg`.
- `git add -A` stages new, modified, and removed files.
- `git commit -m ``<msg>''` commits with the commit message `<msg>`. **Only use if you absolutely know what you're doing.**

Branching I

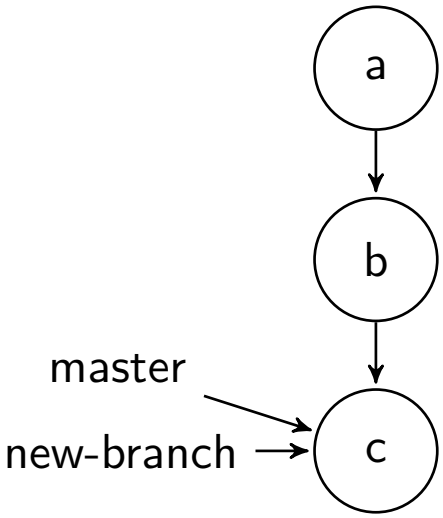
Branching is what allows for multiple people to work on multiple parts of the project.



So $D_1 \cdots E_1$ could be a bug fixing branch while $D_2 \cdots G_2$ could be a feature branch.

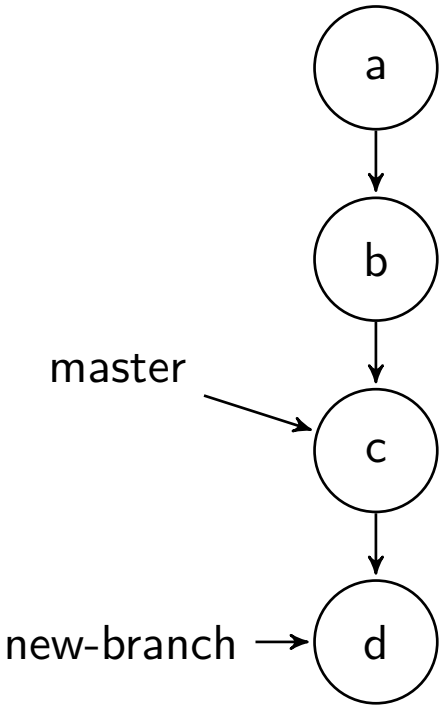
Branching II

```
git checkout -b new-branch
```

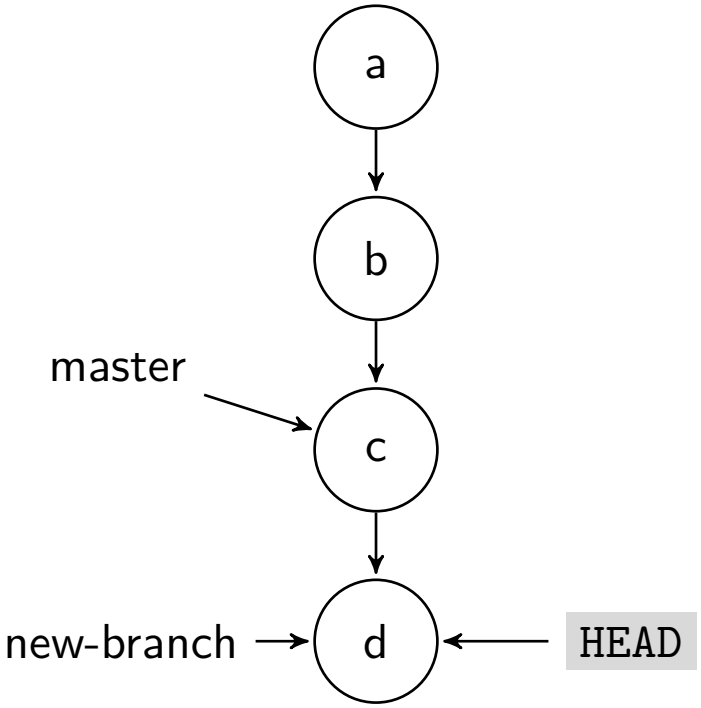


Branching III

```
git commit -m ``d''
```

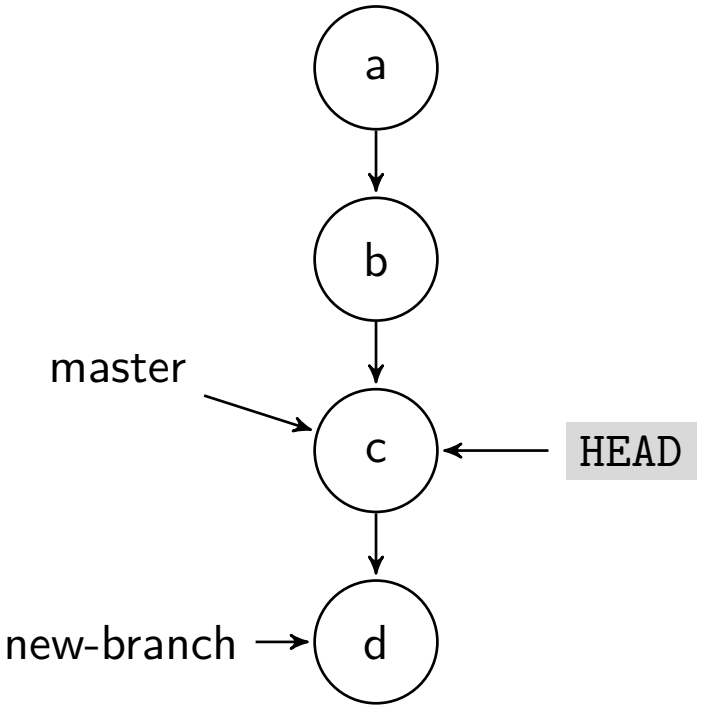


Branching IV



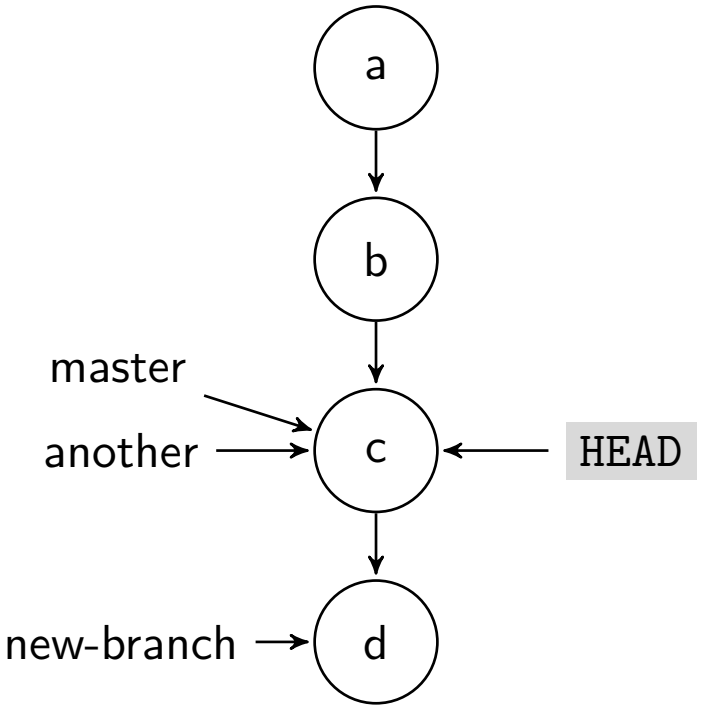
Branching V

```
git checkout master
```



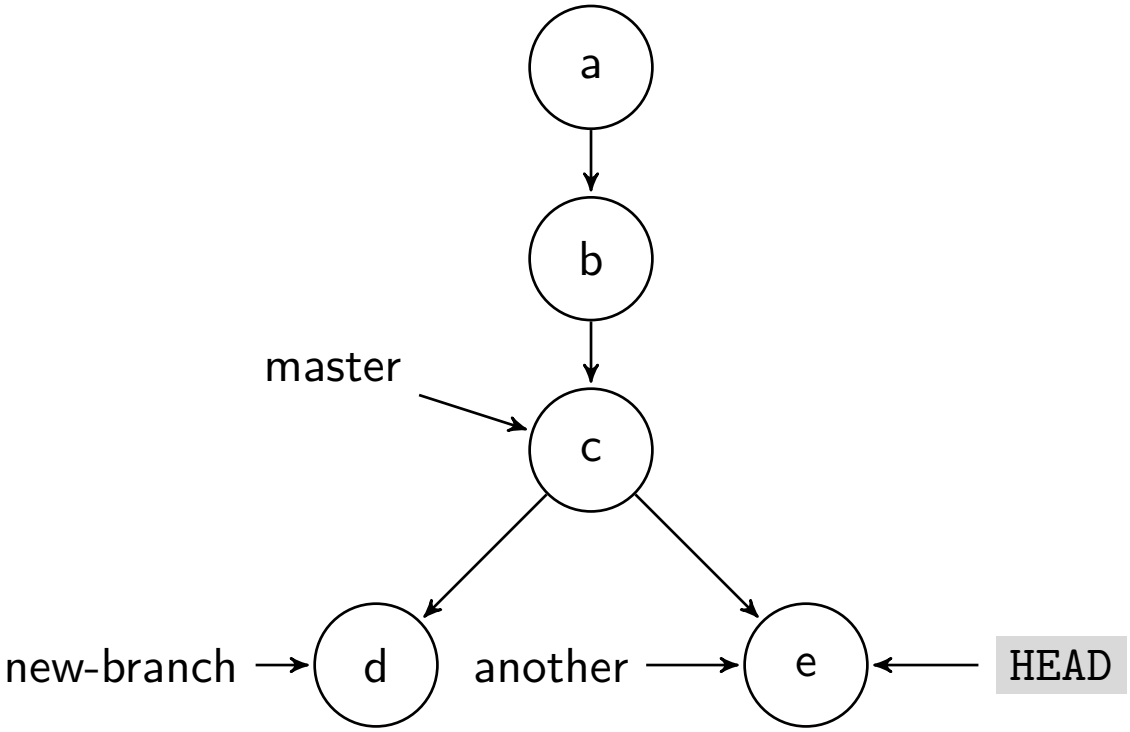
Branching VI

```
git checkout -b another
```



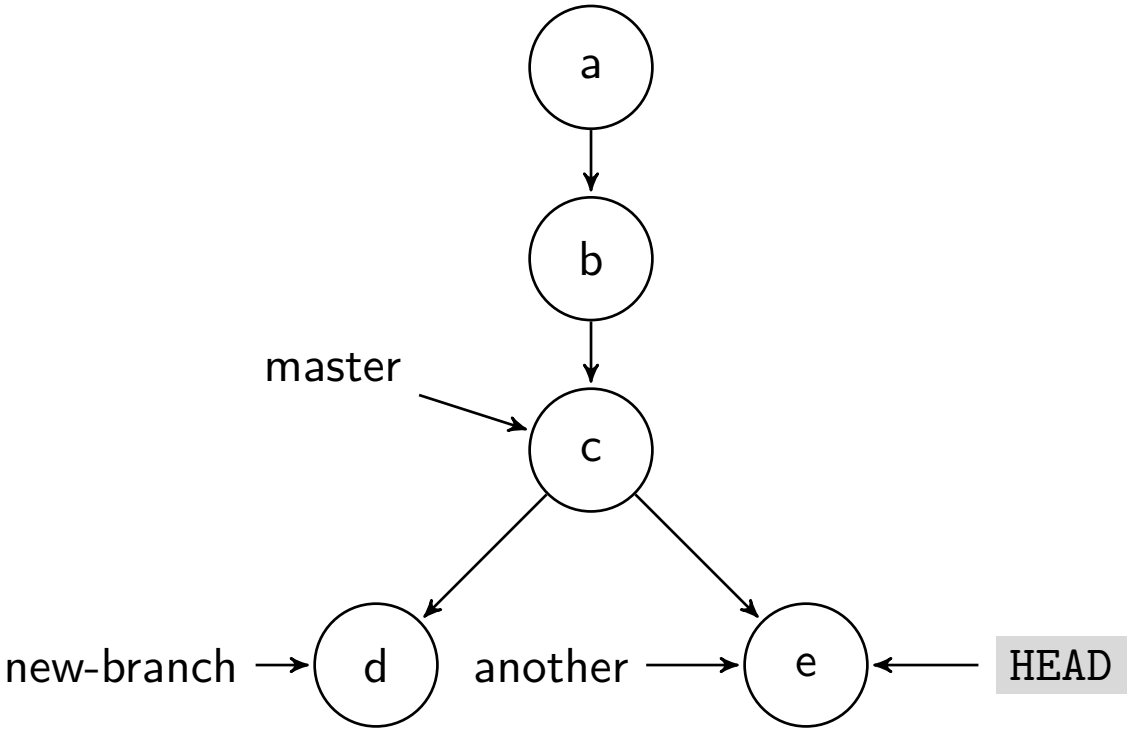
Branching VII

```
git commit -m ``e''
```



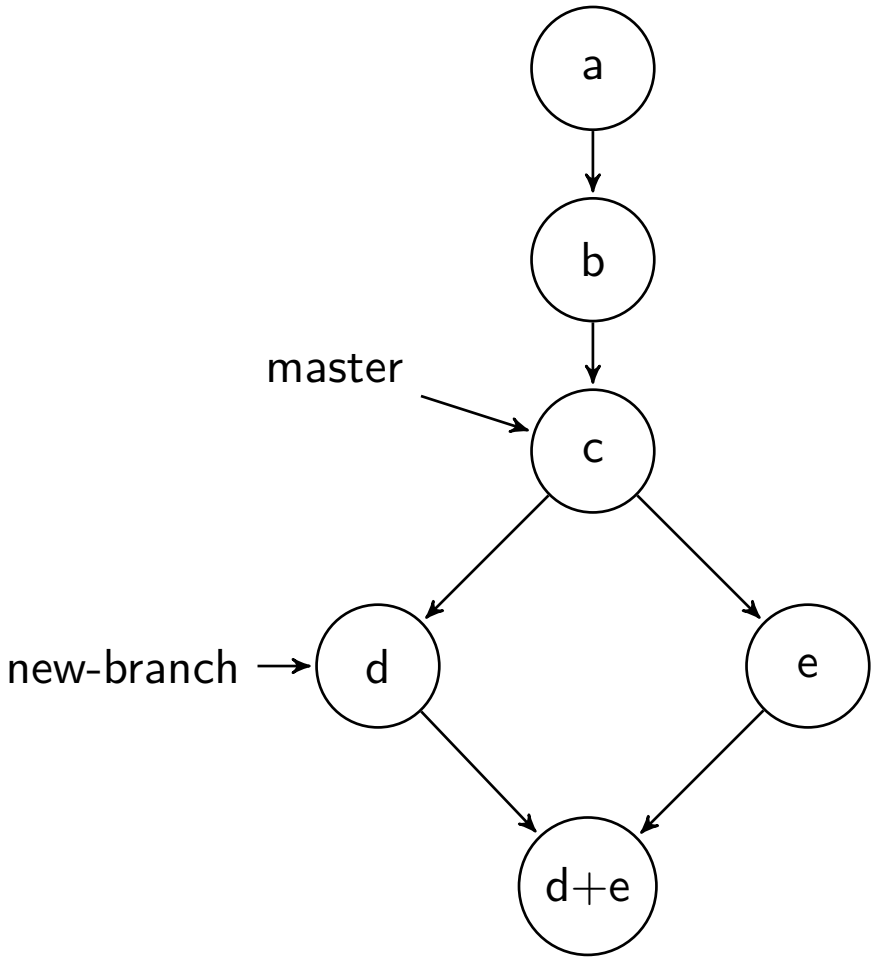
Merging I

```
git merge new-branch
```



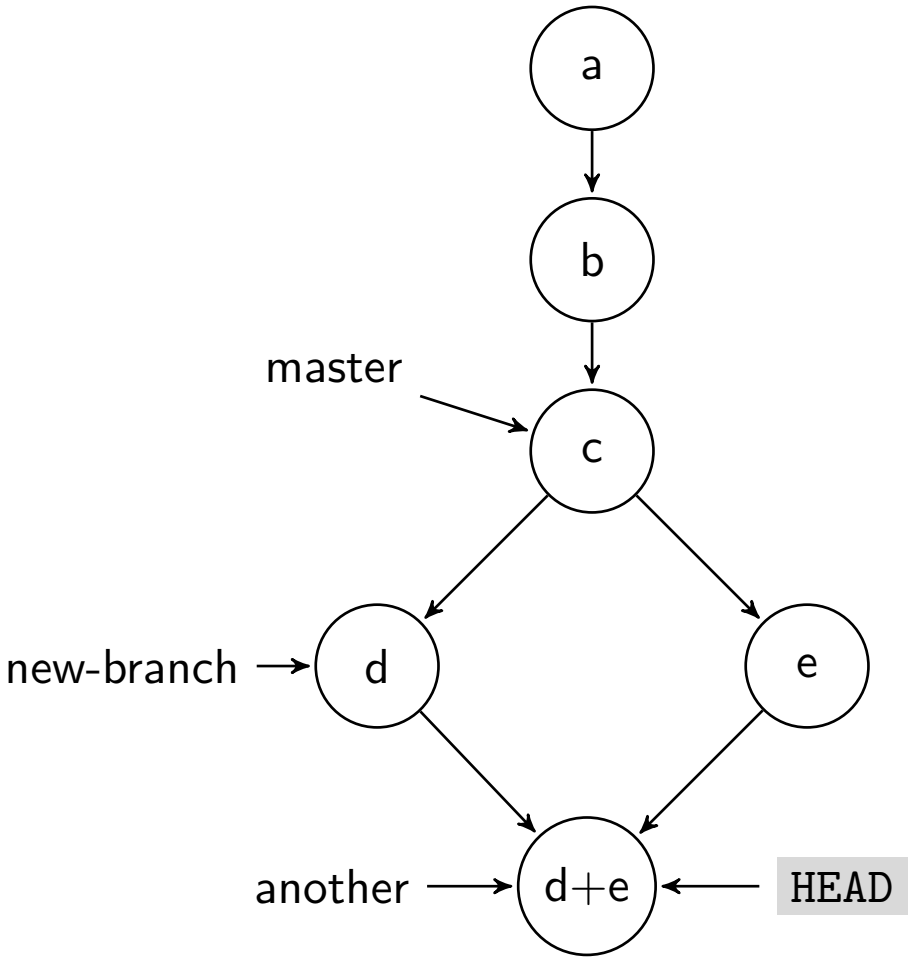
Merging II

```
git merge new-branch
```



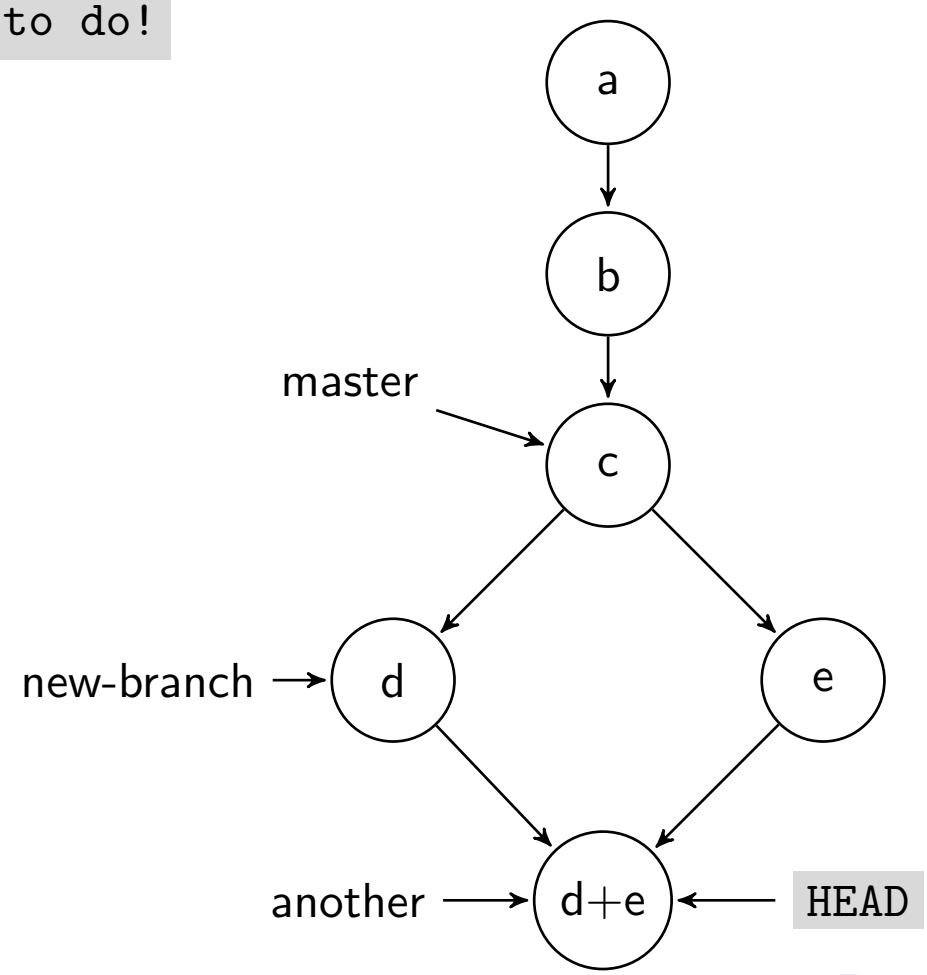
Merging III

```
git merge new-branch
```



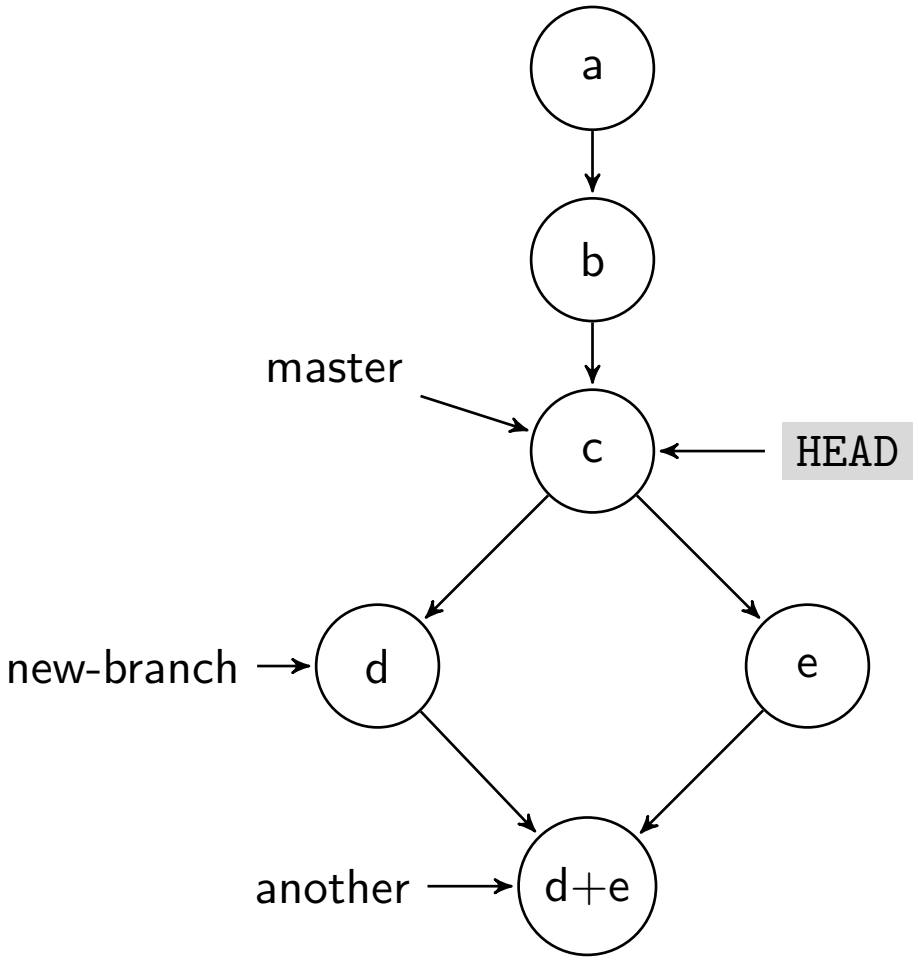
Merging IV

```
git merge master
Nothing to do!
```



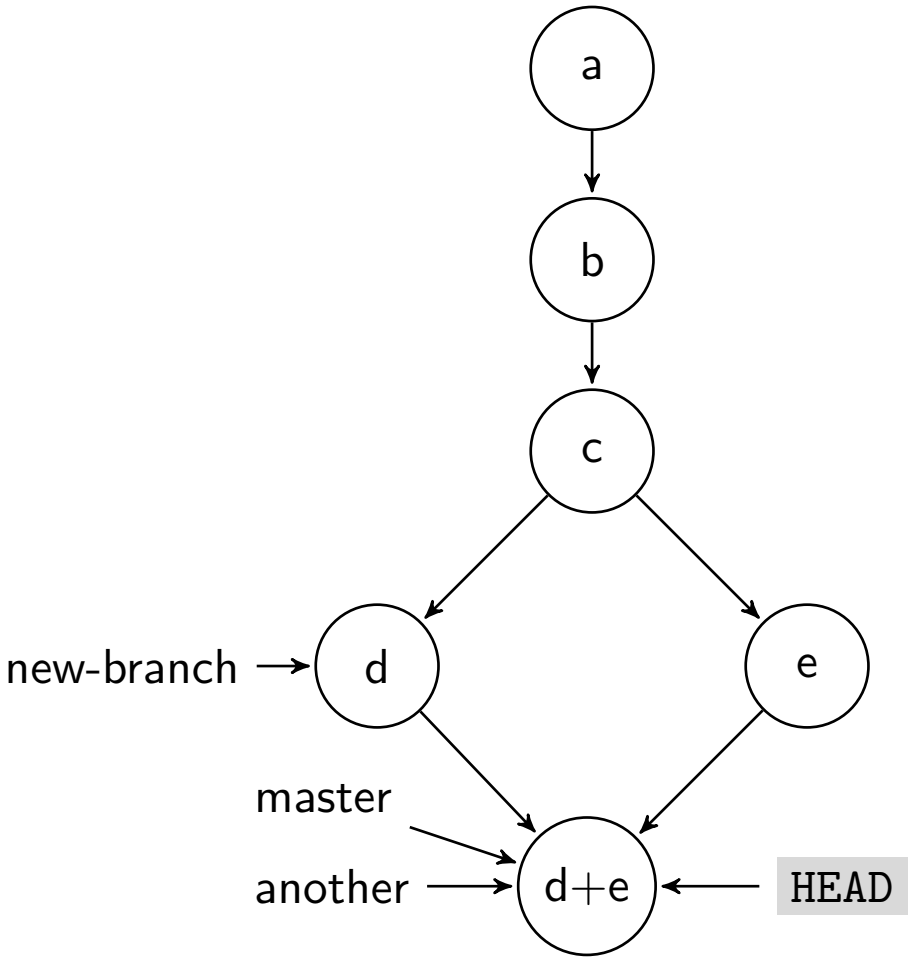
Merging V

```
git checkout master
```



Merging VI

```
git merge another
```



Merging VII

When working with larger repositories, you might not necessarily have rights to arbitrarily merge — you have to create a *merge request*. A merge request allows for someone to review the changes being made to the master branch. *Because merge requests are different per hosting service, this will be shown in the demo.*

Merge Conflicts I

On occasion, you might run into a merge conflict. These arise when you modify two parts of a shared code-base. For instance, an error message could appear like so:

```
Auto-merging the-files(s).txt
CONFLICT (content): Merge conflict in the-file(s).txt
Automatic merge failed; fix conflicts and then commit the result.
```

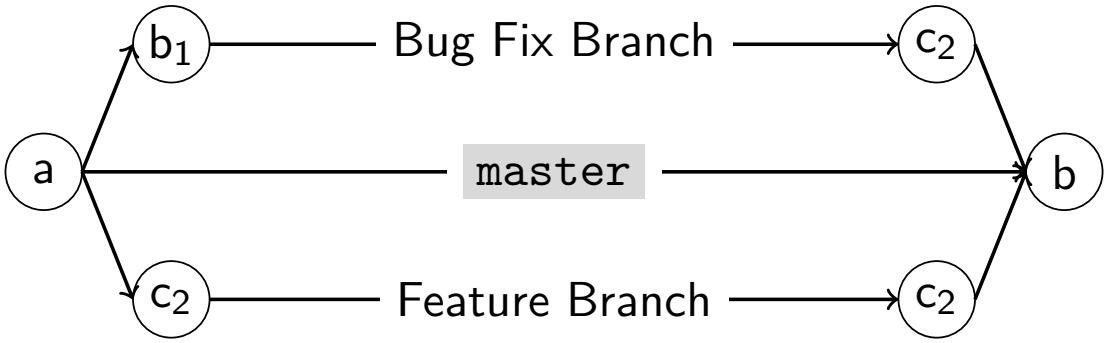
Upon inspection, you should find something along the lines of:

```
<<<<<<< HEAD
The current branch's contents
=======
The branch you're merging's contents
>>>>>>> other-branch
```

Merge Conflicts II

There are many great ways to deal with merge conflicts!

1. Merge early, merge often.



Demo

You Broke It — Finding The Mistake

- `git log` shows all previous commits. It has many useful parameters.
  - `--stat` Shows statistics (deletions, insertions, files changed) for your files.
  - `--pretty=oneline` A one line list of all your changes.
  - `--graph` Shows a graph.
  - `-p` Shows what are the changes to your code base.
- `git diff` show the difference between the working directory and the last commit.

You Broke It — Fixing The Mistake

- `git checkout COMMIT` Just as branches, looks at a previous commit.
- `git revert COMMIT` Generate a new commit that undoes all of the changes introduced in `COMMIT`, then apply it to the current branch.
- `git reset COMMIT` Move `head` to `commit` and reset the staging area to match. *Leaves the working directory alone*.
- `git reset --hard COMMIT` Move `head` to `commit` and reset the staging area to match. *Destroys working direcory*. **Only use if you absolutely know what you're doing.**

Working With Remotes

- `git clone LOCATION` Makes a copy of a repository in the current directory — you can set up SSH keys for Git/Gitlab or just use the url.
- `git push ORIGIN BRANCH` Pushes changes from your current branch to the remote branch it tracks.
- `git pull ORIGIN BRANCH` Pulls changes from the remote branch and merges them into your current branch.

`Git`ting Better-est

If for the first time working on a repo, `git clone` the repo and `cd` into the directory. Then,

1. Pull any new changes with `git pull`.
2. If starting a new feature or bug fix, create new branch with `git branch` (and switch to the branch with `git checkout`).
3. Make your changes.
4. Add your changes with either `git add FILE(s)` or `git add -A`[3].
5. Commit your changes with `git commit`. Provide a **detailed** commit message.
6. When feature/bug fix is finished, merge with either `git merge` or create a merge request. If there are merge conflicts, fix them.
7. `git push` your changes.

In Closing

Special thanks to Nathan Jarus for "lending" me his LaTeX tikz code for the branching, committing and merging section.

@IllyaStarikov

@IllyaStarikov

starikov@mst.com

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.

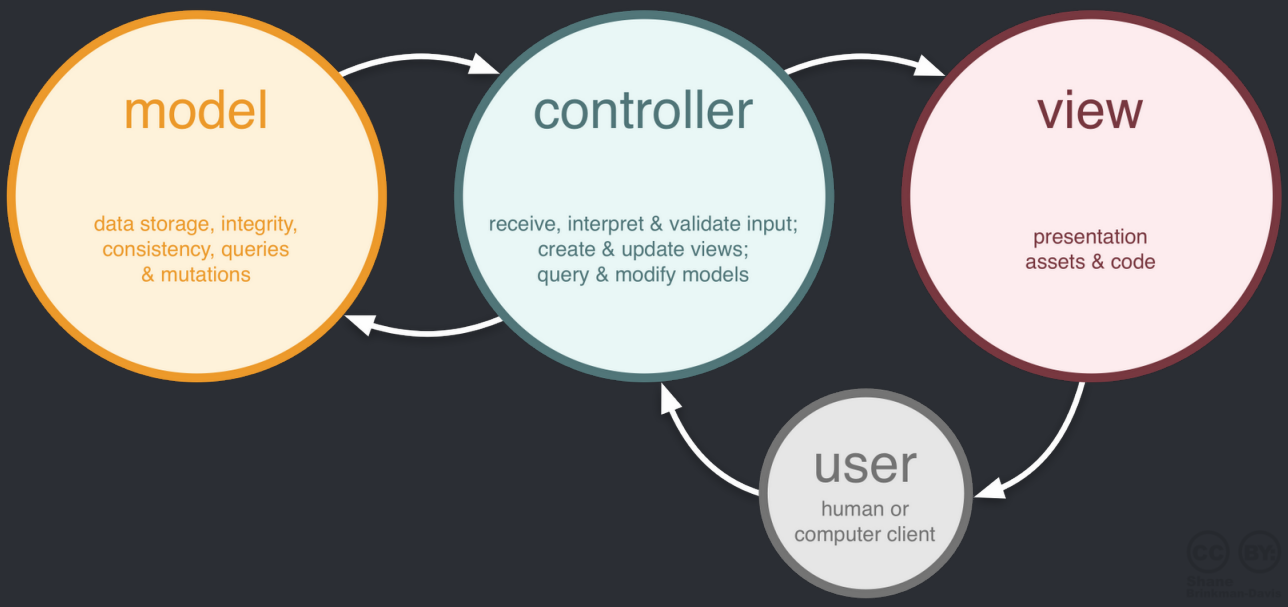
View A visual representation of the model, for user interaction.

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. They 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 private. 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`

GRAD

Graduate Courses

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:

$$\begin{aligned} \text{ageGroup} = \textbf{youngAdult} &\implies \text{football} \\ \text{ageGroup} = \textbf{middleAge} &\implies \text{football} \\ \text{ageGroup} = \textbf{senior} &\implies \text{baseball} \end{aligned}$$

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

$$\text{pet} = \textbf{dog} \text{ and } \text{drink} = \textbf{beer} \implies \text{football}$$

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:

$$\begin{aligned} d = \text{X and } e = 4 &\implies f = T \\ d = \text{S and } e = 4 &\implies f = T \\ d = \text{S and } e = 3 &\implies f = F \\ d = \text{H and } e = 3 &\implies f = F \\ d = \text{M and } e = 4 &\implies f = F \end{aligned}$$

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:

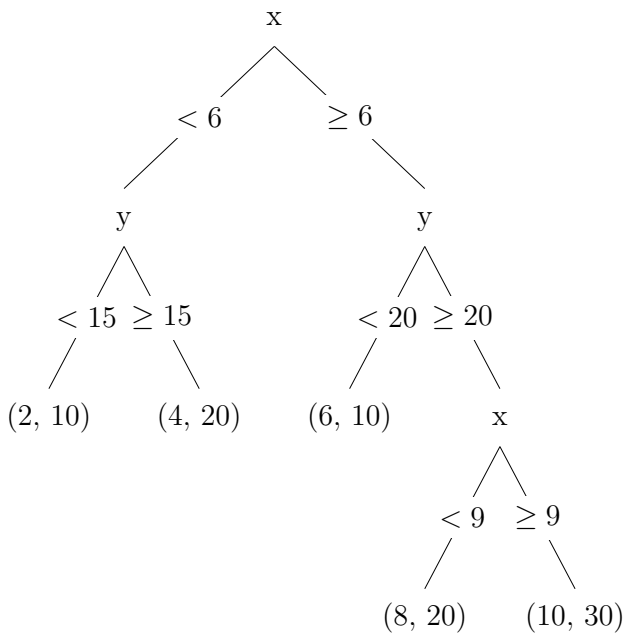
- $y < 15$ group: [(2, 10)]
- $y \geq 15$ group: [(4, 20)]

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) ≈ (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

$$\frac{\text{Non-Random Correct - Random Correct}}{\text{Total}}$$

Which would be as follows:

$$\frac{90 - 39}{100}$$

Exercise #2

Introduction To Artificial Intelligence

Illya Starikov

Due Date: January 24$^{\text{th}}$, 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: $\big(C_L,\, M_L,\, B_{L/R}\big)$, where:

$$\begin{aligned} C_L &= \text{The number of Cannibals on the Left} \\ M_L &= \text{The number of Missionaries on the Left} \\ B_{L/R} &= \text{The location (Left/Right) of the boat} \end{aligned}$$

   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:

$$C_R = 3 - C_L \qquad M_R = 3 - M_L$$

   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 desciption is still $\big(C_L,\, M_L,\, B_{L/R}\big)$, 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:

$$M_L \geq C_L \quad \textbf{and} \quad M_R \geq C_R$$

   Furthermore,

$$B_{L/R} = \text{Left} \iff M_L + B_L = 6 \qquad B_{L/R} = \text{Right} \iff M_R + B_R = 6$$

2. The initial state is as follows:

$$\left(C_L,\, M_L,\, B_{L/R}\right) = (3,\, 3,\, L)$$

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 ¬(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 ← {}                          ▷ A set of sets of moves
12:
13:     for all direction d ∈ {Left, Right} do
14:         if a.M_d > 0 and a.C_d > 0 then
15:             NewAction ← {send Cannibal d → ¬d}
16:             NewAction ← {send Missionary d → ¬d}      ▷ Note this has the form {..., ...}
17:
18:             PossibleActions ← PossibleActions ∪ { NewAction} ▷ Note this has the form {..., {..., ...}, ...}
19:         end if
20:
21:         for i ← 1 to 2 do
22:             if a.M_d − i ≥ 0 then      ▷ Is there even enough missionaries to remove?
23:                 if (a.M_d − i ≥ a.C_d or a.M_d − i = 0) and a.M_¬d + i ≥ a.C_¬d then
24:                     NewAction ← {send i Missionary d → ¬d }[1]
25:                     PossibleActions ← PossibleActions ∪ { NewAction}
26:                 end if
27:             end if
```

[1] Note by saying "send i Missionary", we are just saying "send a missionary" i times. For $i = 2$, we have NewAction ← {send a missionary, send a missionary}.

```
28:
29:             if a.C_d − i ≥ 0 then                ▷ Is there even enough cannibals to remove?
30:                 if a.C_¬d + i ≤ a.M_¬d or a.M_¬d = 0 then
31:                     NewAction ← {send i Cannibal d → ¬d}[2]
32:                     PossibleActions ← PossibleActions ∪ { 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′ ← a
3:     a′.B_L/R = ¬a′.B_L/R
4:
5:     for all action ∈ a do                ▷ We defined an action as a set of steps
6:         if action = send Cannibal Left → Right then
7:             a′.C_L = a′.C_L − 1
8:         end if
9:
10:        if action = send Cannibal Right → Left then
11:            a′.C_L = a′.C_L + 1
12:        end if
13:
14:        if action = send Missionary Left → Right then
15:            a′.M_L = a′.M_L − 1
16:        end if
17:
18:        if action = send Missionary Right → 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)
```

[2]See previous footnote.

2: **return** $s = (0, 0, \text{Right})$
3: **end function**

6. We define $\text{C}(s, a, s')$

1: **function** $\text{C}(s, a, s')$
2: **return** $|\text{a}|$[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.

[3]We define the $|x|$ operator to return the length of x.

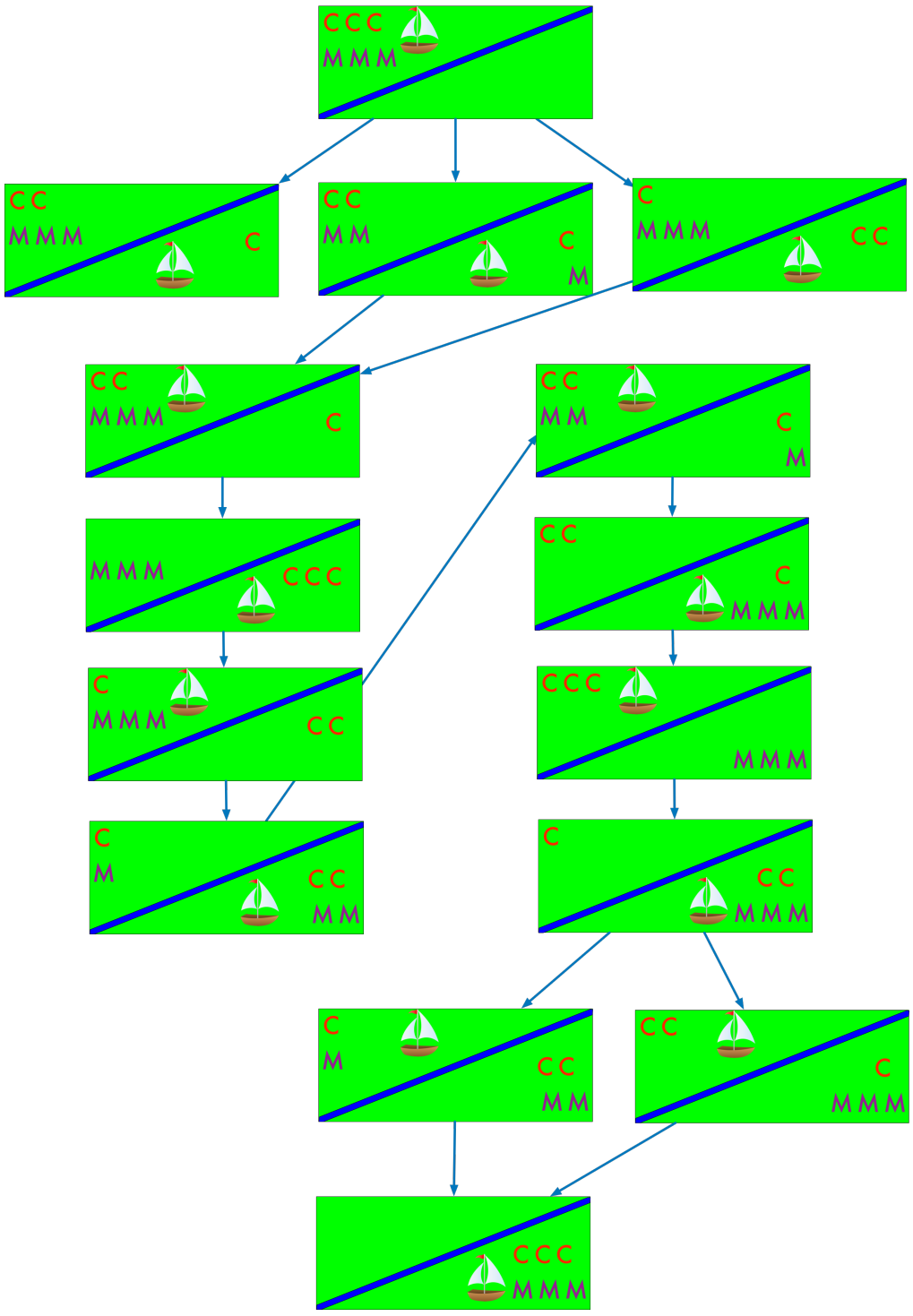


Figure 1: The Complete State Space

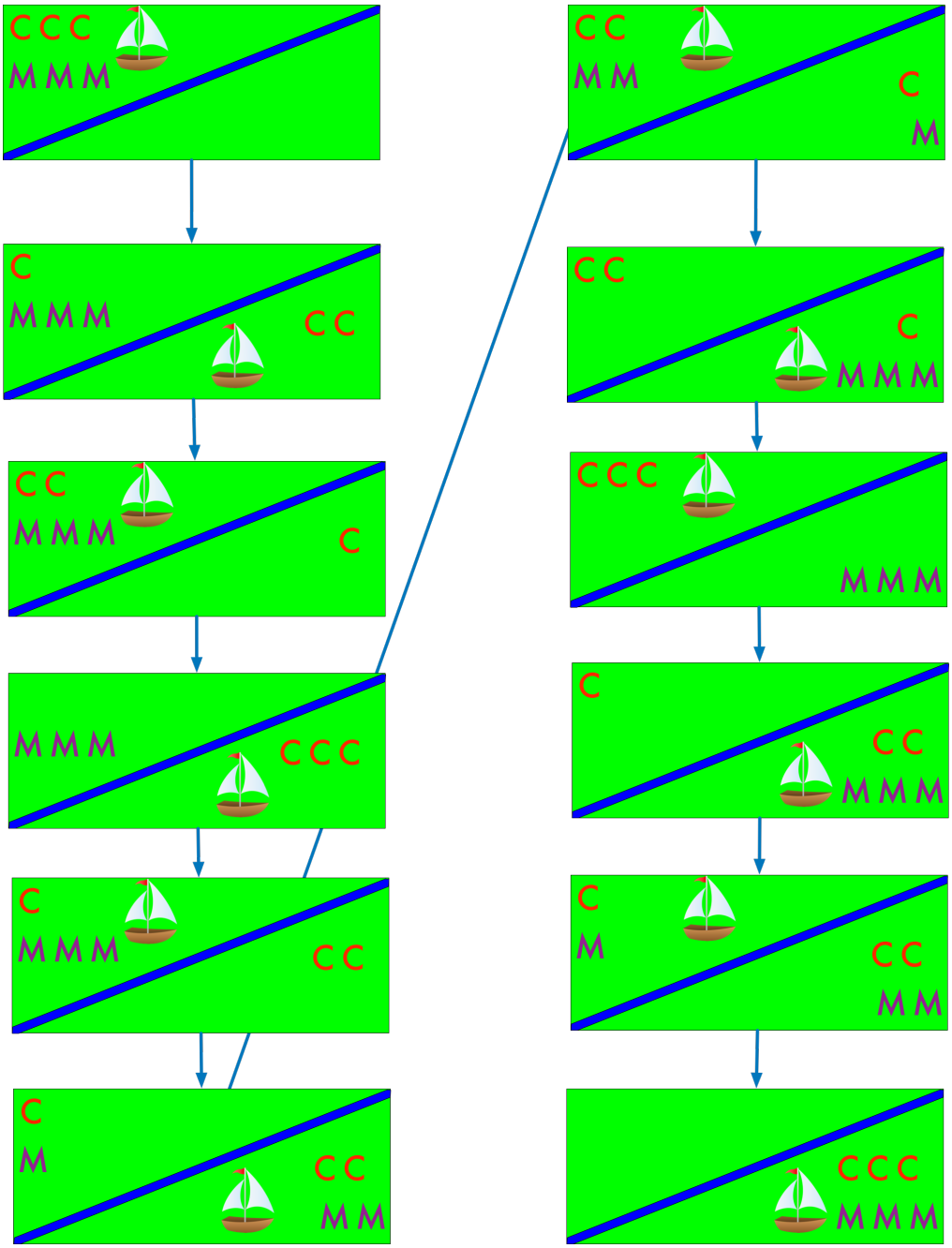


Figure 2: Path To A Solution

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 $\mathcal{O}\left(b^d\right)$.

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 $\mathcal{O}\left(\frac{d}{\epsilon}\right)$.

Homework #8

Analysis of Algorithms

Illya Starikov

Due Date: November 27th, 2017

Contents

Listings

Question #1

Let the samples space $S = \{X_{\text{free}} \implies$ prisoner X is going free, $Y_{\text{free}} \implies$ prisoner Y is going free, Z_{fre} prisoner Z 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 $\mathcal{O}\left({}_{n}um_{1}\right)$ 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.

Question #4

Recall Stirling's Approximation for factorials:

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \tag{1}$$

We skip the $\frac{1}{2}$ and $\frac{1}{n}$ case; for if we can't prove $\frac{1}{2^n} \times n! \in \Omega\left({}_n um_1\right)$, then neither are $\frac{1}{2}$ and $\frac{1}{n}$.

Theorem 1.

$$\forall n \in \mathbb{R}^+, \frac{1}{2^n} \times n! \notin \Omega\left({}_n um_1\right)$$

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

$$\lim_{n\to\infty} \frac{\frac{1}{2^n} \times n!}{c\, n} = \frac{\frac{1}{2^n} \times \sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{c\, n} = \infty \qquad \forall c \in R^+$$

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 nth digit, call it d_1 and d_2 from the two respective lists, it appears something like such:

$\cdots$	$\cdots$	d_1	$\cdots$	$\cdots$
$\cdots$	$\cdots$	d_2	$\cdots$	$\cdots$

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.

$\cdots$	$\cdots$	42	64	$\cdots$
$\cdots$	$\cdots$	42	16	$\cdots$

□

Question #6

We can accomplish $n + \lg n - 2$ runtime as such:

1. Split all elements into pairwise elements ($S \implies (s_1, s_2), (s_3, s_4), \ldots, (s_{n-1}, s_n)$).
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:

$$\begin{aligned} \text{Number of Comparison} &\in \Omega\left({}_n um_1\right) \\ &\leq 10 + \lg 10 - 2 \\ &\leq 12 \\ &= 7 \end{aligned}$$

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 $\mathcal{O}\left({}_{n}um_{1}\right) = \mathcal{O}\left({}_{n}um_{1}\right)$.

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.

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-Cylde check problem is NP. Given a solution S_{Bonnie} and S_{Clyde}, we must verify

$$\sum_{x \in S_{\text{Bonnie}}} x\text{'s worth} = \sum_{x \in S_{\text{Cylde}}} x\text{'s worth} \qquad |S_{\text{Bonnie}}| + |S_{\text{Clyde}}| = n$$

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. □

[1] Cardinality 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

$$\left| \sum_{x \in S_{\text{Bonnie}}} x\text{'s worth} - \sum_{x \in S_{\text{Cylde}}} x\text{'s worth} \right| = 100 \qquad |S_{\text{Bonnie}}| + |S_{\text{Clyde}}| = n$$

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. □

[2]Cardinality 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:

$$\{1, 2, 3, 4, 5\}$$

Homework #8

Analysis of Algorithms

Illya Starikov

Due Date: November 27th, 2017

Contents

Listings

Question #1

Theorem 1. *We take the definition of $\mathcal{O}$ as such.*

For $f(n) \in \mathcal{O}\,({}_n um_1)$, there must exist constants c, where $c > 0$, and there must exist $n_0 \in \mathbb{N}$, where $n_0 \geq 1$, such that

$$|f(n)| \leq c|g(n)| \ \forall n \geq n_0$$

Problem #1.1

Problem Statement. ***Without using limits, but only the definition of*** $\mathcal{O}$ *prove that $81n^3 + 1300n^2 + 300n \in \mathcal{O}\,({}_n um_1)$, but that $n^5 - 15000n^4 - 10n^3 \notin \mathcal{O}\,({}_n um_1)$. Show all work.*

Proof. In this instance, $f(n) = 81n^3 + 1300n^2 + 300n$ and $g(n) \in \mathcal{O}\,({}_n um_1)$. To prove that $f(n) \in \mathcal{O}\,({}_n um_1)$, first we must find constants c and n_0.

Take the following, $c = 1$, $n_0 = 15\,000$. Then, for all $n \geq 15000$,

$$81n^3 + 1300n^2 + 300n \ll n^5 - 15000n^4 - 10n^3$$

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

$$|f(n)| \leq c|g(n)| \ \forall n \geq n_0$$

Therefore, $f(n) \in \mathcal{O}\,({}_n um_1)$, and $81n^3 + 1300n^2 + 300n \in \mathcal{O}\,({}_n um_1)$. □

Proof. Suppose not. That is, suppose that $\exists c, n_0$ such that

$$|f(n)| \leq c|g(n)| \ \forall n \geq n_0$$

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:

$$
n \leq \max \left(n_0, \delta + 1 + 3.6 \times 10^{-7} c \right)
$$

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

$$
n^5 - 15000n^4 - 10n^3 \notin \mathcal{O}\left({}_n u m_1\right)
$$

□

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* $f(x) \in \mathcal{O}\left({}_n um_1\right)$ *or* $g(x) \in \mathcal{O}\left({}_n um_1\right)$. *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).

$$\sum_{n=1}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{1}$$

For $f(x)$ and $g(x)$, we get the following expansions.

$$\sum_{n=1}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = -4x^2 + 2x^3 + \frac{4x^4}{3} - x^5 - \frac{8x^6}{45} + \frac{13x^7}{60} + \frac{4x^8}{315} + \mathcal{O}\left({}_n um_1\right)$$

$$\sum_{n=1}^{\infty} \frac{g^{(n)}(a)}{n!}(x-a)^n = 7 - \frac{13x^2}{2} + \frac{85x^4}{24} - \frac{1093x^6}{720} + \frac{3329x^8}{8064} - \frac{263173x^{10}}{3628800} + \frac{839681x^{12}}{95800320} + \mathcal{O}\left({}_n um_1\right)$$

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 $n_0 = 1$,

$$\begin{aligned} |f(x)| \leq c|g(x)| \ \forall n \geq n_0 &\implies f(x) \in \mathcal{O}\left({}_n um_1\right) \\ &\implies 2\sin^3(x) - 4\sin^2(x) \in \mathcal{O}\left({}_n um_1\right) \end{aligned}$$

□

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:

$$sq(n) = \sum_{i=0}^{n+1} q(n) = q(n+2) - 7 \tag{3}$$

Define The Problem For this problem, we wish to prove that $p \overset{n}{\underset{p}{um_1}} \overset{\propto}{\sim} q$ 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 inducion 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,

$$
\begin{gathered}
\text{S} = (0,\ 0,\ 0),\ (0,\ 0,\ 1),\ (0,\ 0,\ 2),\ (0,\ 0,\ 3),\\
(0,\ 1,\ 0),\ (0,\ 1,\ 1),\ (0,\ 1,\ 2),\ (0,\ 1,\ 3),\ (0,\ 2,\ 0),\\
(0,\ 2,\ 1),\ (0,\ 2,\ 2),\ (0,\ 2,\ 3),\ (0,\ 3,\ 0),\ (0,\ 3,\ 1),\\
(0,\ 3,\ 2),\ (0,\ 3,\ 3),\ (1,\ 0,\ 0),\ (1,\ 0,\ 1),\ (1,\ 0,\ 2),\\
(1,\ 0,\ 3),\ (1,\ 1,\ 0),\ (1,\ 1,\ 1),\ (1,\ 1,\ 2),\ (1,\ 1,\ 3),\\
(1,\ 2,\ 0),\ (1,\ 2,\ 1),\ (1,\ 2,\ 2),\ (1,\ 2,\ 3),\ (1,\ 3,\ 0),\\
(1,\ 3,\ 1),\ (1,\ 3,\ 2),\ (1,\ 3,\ 3),\ (2,\ 0,\ 0),\ (2,\ 0,\ 1),\\
(2,\ 0,\ 2),\ (2,\ 0,\ 3),\ (2,\ 1,\ 0),\ (2,\ 1,\ 1),\ (2,\ 1,\ 2),\\
(2,\ 1,\ 3),\ (2,\ 2,\ 0),\ (2,\ 2,\ 1),\ (2,\ 2,\ 2),\ (2,\ 2,\ 3),\\
(2,\ 3,\ 0),\ (2,\ 3,\ 1),\ (2,\ 3,\ 2),\ (2,\ 3,\ 3),\ (3,\ 0,\ 0),\\
(3,\ 0,\ 1),\ (3,\ 0,\ 2),\ (3,\ 0,\ 3),\ (3,\ 1,\ 0),\ (3,\ 1,\ 1),\\
(3,\ 1,\ 2),\ (3,\ 1,\ 3),\ (3,\ 2,\ 0),\ (3,\ 2,\ 1),\ (3,\ 2,\ 2),\\
(3,\ 2,\ 3),\ (3,\ 3,\ 0),\ (3,\ 3,\ 1),\ (3,\ 3,\ 2),\ (3,\ 3,\ 3)
\end{gathered}
$$

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 $\mathcal{O}\left({}_{n}um_{1}\right)$ solution to $H(n)$, the following was produced:

$$H(n) = \begin{cases} 6 & \iff n \mod 4 = 0 \\ 7 & \iff n \mod 2 \neq 0 \\ 8 & \iff (n-2) \mod 4 = 0 \end{cases}$$

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* $^2/_3$*rds 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 ith operation to be

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} \tag{4}$$

Assuming the ith 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 ith 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 cent, 8 cents, 20 cents}, {1 cent, 6 cents, 18 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 $S = \varnothing$ 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 ($1 \times 20¢ + 4 \times 1¢$), while the optimal solution would take 4 coins ($4 \times 8¢$).

[1] Because 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 simple iterate through all edges, and check:

- Make sure that the graph is colored by ≤ 4 colors.
- Iterate though all edges, ensuring that every edge's neighbor is colored by a different color.

Because this is a $\mathcal{O}\left({}_{n}um_{1}\right)$ 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 verticies 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 reducable from the Three Coloring Problem, we conclude that the Four Coloring Problem is NP-Complete. □

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 simple iterate through all edges, and check:

- Make sure that the graph is colored by $\leq k$ colors.
- Iterate though all edges, ensuring that every edge's neighbor is colored by a different color.

Because this is a $\mathcal{O}\left({}_{n}um_{1}\right)$ 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 reducable from the Three Coloring Problem, we conclude that the k-Coloring Problem is NP-Complete.

□

CS

Computer Science

- 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? 2^{24} 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¿ 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 Richie 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 Wozniac 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.? Xerox/PARC
- 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 -¿ 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.

ForFit

Never Quit Again

Illya Starikov, Jason Young, Claire Trebing

July 27, 2025

Contents

Part I.

The Database Design Manual

1. Problem Statement

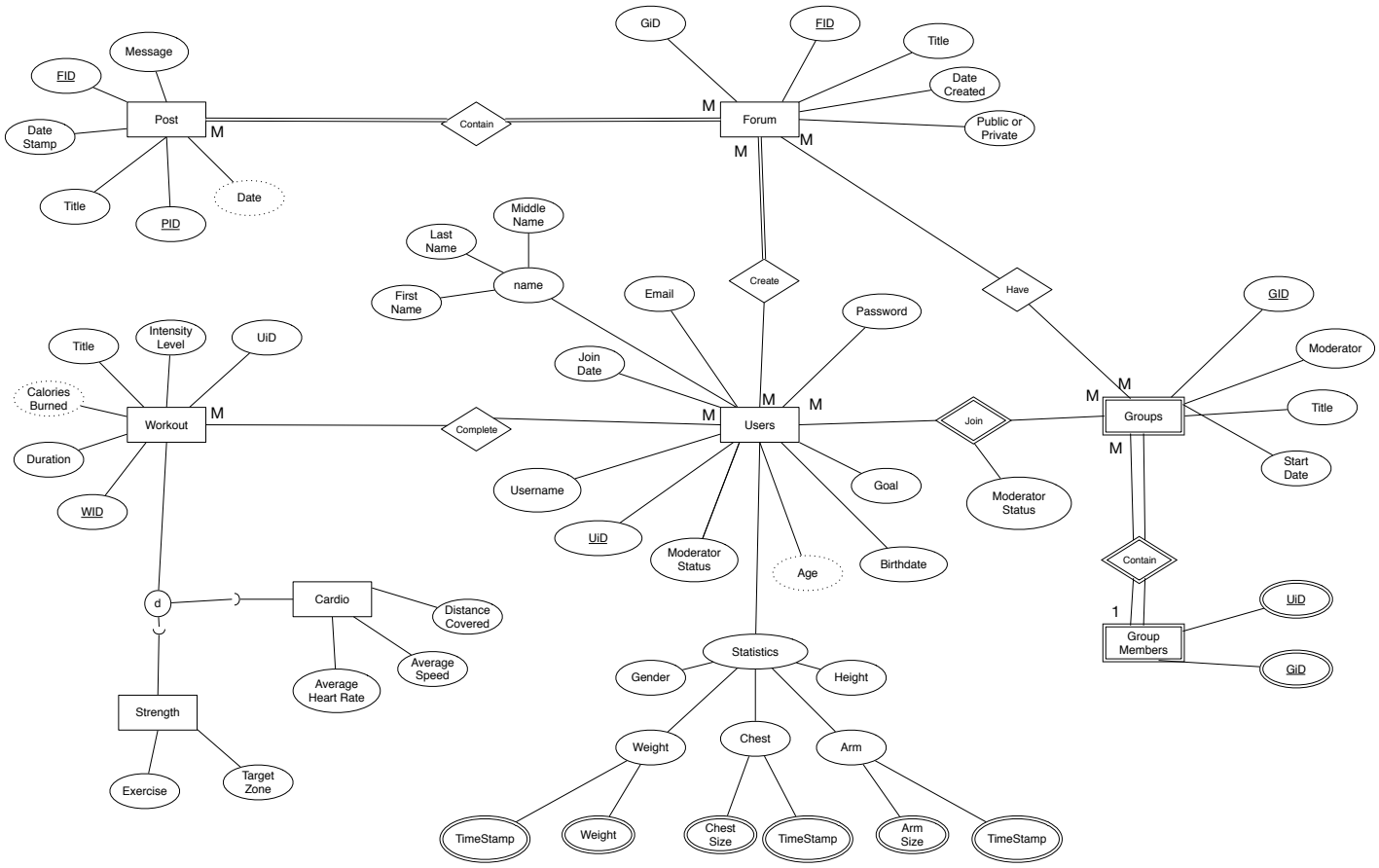
2015 marked a record year for Americans regularly exercising, hitting just over 55%. Keeping that momentum is difficult. As blue-collar jobs continue to decline, it has been more important than ever to keep a weekly regimen of healthy eating and exercising.

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


GiD
FID
Title
Date Created
Public or Private
Forum
Contain
Post
Message
FID
Date Stamp
Title
PID
Date
M
M
M
M
Middle Name
Last Name
First Name
name
Email
Create
Password
Join Date
Have
GID
Moderator
Title
Start Date
Groups
Join
Moderator Status
Users
Complete
Workout
Title
Intensity Level
UiD
Calories Burned
Duration
WID
Username
UiD
Moderator Status
Age
Birthdate
Goal
Contain
1
Group Members
UiD
GiD
d
Cardio
Distance Covered
Average Speed
Average Heart Rate
Strength
Exercise
Target Zone
Statistics
Gender
Height
Weight
Chest
Arm
TimeStamp
Weight
Chest Size
TimeStamp
Arm Size
TimeStamp

3. Logical Database Design

3.1. Relational Set

Please note primary keys are signified by $^{\mathbf{PK}}$ and foreign keys are signified by ^{FK}.

Post

PiD $^{\mathbf{PK}}$	FiD ^{FK}	Message	DateStamp	Title	Date

Forum

FiD $^{\mathbf{PK}}$	Title	DateCreated	PublicOrPrivate	GiD ^{FK}

Groups

GiD $^{\mathbf{PK}}$	Moderator ^{FK}	Title	StartDate

Workouts

WiD $^{\mathbf{PK}}$	Duration	Title	IntensityLevel	CaloriesBurned	UiD ^{FK}

Strength

WiD $^{\mathbf{PK}}$	Duration	Title	IntensityLevel	CaloriesBurned	UiD ^{FK}	Target Zone

Cardio

WiD $^{\mathbf{PK}}$	Duration	Title	IntensityLevel	CaloriesBurned	UiD ^{FK}	AverageHeartRate
AverageSpeed	DistanceCovered					

Users

UiD $^{\mathbf{PK}}$	Username	Height	Birthdate	Goal	Password	JoinDate	Gender
FirstName	MiddleName	LastName					

Weight

UiD $^{\mathbf{PK, FK}}$	TimeStamp $^{\mathbf{PK}}$	Weight

Arm Size

UiD $^{\mathbf{PK, FK}}$	TimeStamp $^{\mathbf{PK}}$	Arm Size

Chest Size

UiD $^{\mathbf{PK, FK}}$	TimeStamp $^{\mathbf{PK}}$	ChestSize

Group Members

UiD **PK**	GiD *FK*	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 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 `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).

4.2. 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.

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.

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.

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 | EtZEdlnfNiBAI | 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.

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.

Parameters

None.

Results

Returns the top 10 users (calculated via count of workouts done) 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 user_leaderboards();
   user_leaderboards
----------------------
 (Gregory,Patty,6)
 (Agueda,Heather,4)
 (Bertram,Gerardoy,4)
 (Gay,Arthury,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 makes sure all of the function are created can tedious. Forunatly, we have a solution — that solution is Shell scripting.

We quickly realized that repetitively running the same commands[1] 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 give 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 artifically 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:

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

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.

Quiz 10

Illya Starikov

August 7, 2025

Problem #1

$$
\begin{aligned}
& S && \\
\implies & AB && \text{via } S \to AB \\
\implies & aXB && \text{via } A \to aX \\
\implies & aXbYd && \text{via } B \to bYd \\
\implies & aXbYd && \text{via } B \to bYd \\
\implies & abXYd && \text{via } Xb \to bX \\
\implies & abYcd && \text{via } XY \to Yc \\
\implies & abcd && \text{via } Y \to \lambda
\end{aligned}
$$

Problem #2

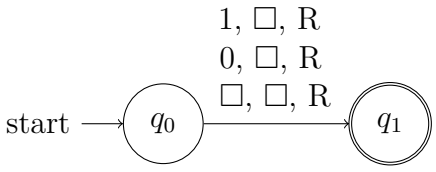
$$
\begin{aligned}
& S & \\
\Longrightarrow\ & TbC & \text{via } S \to TbC \\
\Longrightarrow\ & gC & \text{via } Tb \to g \\
\Longrightarrow\ & Sg & \text{via } gC \to Sg \\
\Longrightarrow\ & TbCg & \text{via } S \to TbC \\
\Longrightarrow\ & gCg & \text{via } Tb \to g \\
\Longrightarrow\ & Sgg & \text{via } gC \to Sg \\
\Longrightarrow\ & TbCgg & \text{via } S \to TbC \\
\Longrightarrow\ & gCgg & \text{via } Tb \to g \\
\Longrightarrow\ & egg & \text{via } gC \to e
\end{aligned}
$$

Problem #3

The language is derived by the grammar is $L = \{aaa^n b^k \mid 0 \leq n \leq k, n + k = 2c\}$, for some arbitrary constant c (i.e. $n + k$ is even).

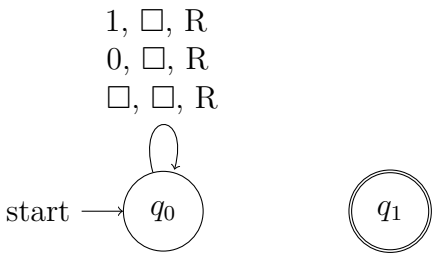
Problem #4

The following always halts, regardless of the input from $\Sigma = \{0, 1\}$.



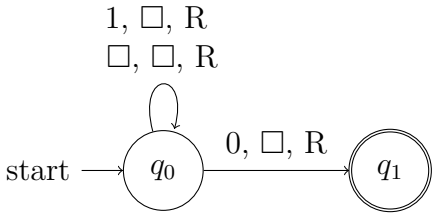
Problem #5

The following never halts, regardless of the input from $\Sigma = \{0, 1\}$. Note it also never halts even if the tape is blank (λ).



Problem #6

The following halts for some, but not all, input from $\Sigma = \{0, 1\}$. In this case, the only input that would cause the machine to halt would be 0.



Problem #7

No, we will prove so as follows.

Theorem 1. *Suppose A to be an arbitrary program — that is, $A \in$ all programs. Then, $\forall A \in$ all program, there **does not** exists a program P such that P can determine if A will halt **regardless of input**.*

Proof. Suppose not. That is, suppose $\exists P \in$ all programs such that P can determine if $\forall A \in$ all programs, A will halt. Suppose the following to be such program P, with an additional source code at the end:

```
bool willHalt(program P) { ··· }

int recursiveNightmare() {
    if (willHalt(P)) {
        return recursiveNightmare();
    }

    return 42;
}
```

If we were to run this program through P, that is $P(P)$, we have the following cases:

Case 1: The program P loops forever. Assuming the program to be correct, this will return false; hence program P halts on input P, which is a contradiction (by our `recursive nightmare`).

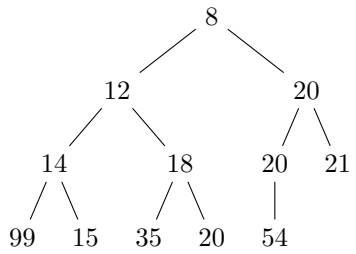
Case 2: The program halts on input P. Assuming the program to be correct, this will return true; hence program P runs indefinitely on input P; this is a contradiction.

We see this to be a paradox; P only halts when `willHalt` will not halt, and runs indecently when it does halt. Either way, we see this to be a contradiction. Therefore, there **does not** exists a program P such that P can determine if A will halt **regardless of input**.

□

Heap

- Binary Search Tree
- All but the bottom level is complete.
- From every node x, x is less than it's two children.
- Member functions.
  - top(T) = *8*
  - insert(T, x)
  - remove(T) = *T = 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.
    }
};
```

Test II Study Guide

Prepared By: Illya Starikov

July 27, 2025

5 Sequences, Mathematical Induction, and Recursion

5.1 Sequences

If $a_m, a_{m+1}, a_{m+1} \ldots$ and $b_m, b_{m+1}, b_{m+1} \ldots$ 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. $(\Pi_{k=m}^{n} a_k) \times (\Pi_{k=m}^{n} b_k) = \Pi_{k=m}^{n} a_k \times b_k$

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 $n \geq 1$,

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

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 $\mathbf{A = B}$, 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:

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

Partition

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

1. A is the union of all the A_i
2. The sets $A_1, A_2, A_3, \ldots$ 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

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

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

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 $f : x \to Y$, 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,

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

This can be read as: A function $F : X \to Y$ 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:

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

In other words, $F : X \to Y$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:

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

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

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

This is equivelent 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 $A_1, A_2, \ldots, A_n$, an ***n*-ary relation** R on $A_1 \times A_2 \times \ldots \times A_n$.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* $(x, y) \notin R$].

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* $(x, y) \notin R$].

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

$$m \equiv n (\text{mod } d)$$

if, and only if,

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

Symbolically:

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

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 $i = 1, 2, \ldots, m$, 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 $i = 1, 2, \ldots, k-1$ 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 $\epsilon \preceq s$.

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.

CPE

Computer Engineering

Homework 8

Illya Starikov

Due Date: December 1st, 2016

This homework will be **extra credit.**

Problem 1

```
#include <reg51.h>

typedef enum { false, true } bool;
unsigned char R0;

void main() {
    TMOD = 0x06; // Counter 2, Mode 2
    TH0 = -0x60; // Initial value given

    R0 = 0; // Just in case
    TR0 = true; // Start the count

    do {
        if (TF0) {
            R0++;
            TF0 = false; // continue the count
        }
    } while (R0 != 60);

    TR0 = false; // Stop the count
}
```

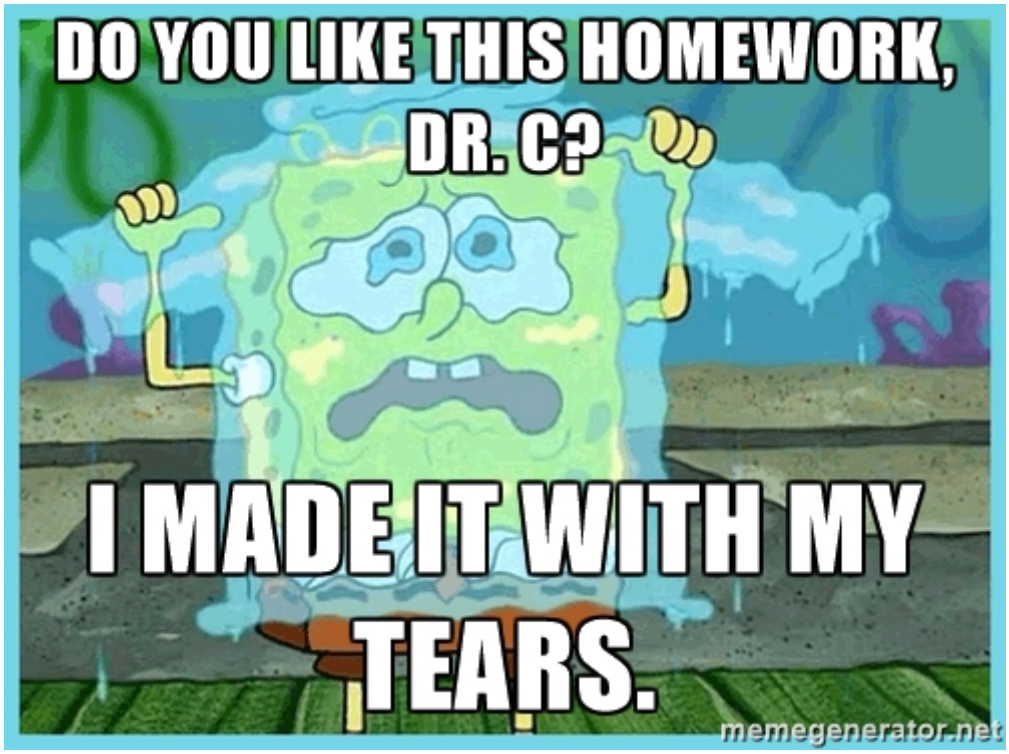
Problem 2

```
        ORG 0H

        LJMP MAIN

        /* Timer 1 Interrupt */
        ORG 001BH
```

```
7           CPL P1.1
8           RETI
9
10          /* Main */
11          ORG 0030H
12  MAIN:   MOV P2, _num_00H          ; P2 = output port
13          MOV TMOD, _num_20H        ; Timer 1, mode 2
14          MOV TH1, #-37D        ; 5 kHz => 25000 Hz => 4^-5 s
15                                ; => 40 us => 40/1.085 ticks => 37 ticks
16          MOV IE, _num_88H
17          SETB TR1
18
19  LOOP:   INC R0
20          MOV P2, R0            ; P2 = R0
21          ACALL DELAY
22          SJMP LOOP
23
24  ; so we need to generate a 10.85us delay
25  ; 10.85/1.085 = 10MC. Easier to just do a brute force
26  DELAY:  MOV R1, _num_4D           ; 2 MC
27  HERE:   DJNZ R1, HERE         ; 4 * 2 = 8 MC
28          RET                   ; 8 + 2 = 10 MC
29
30
31          END
```

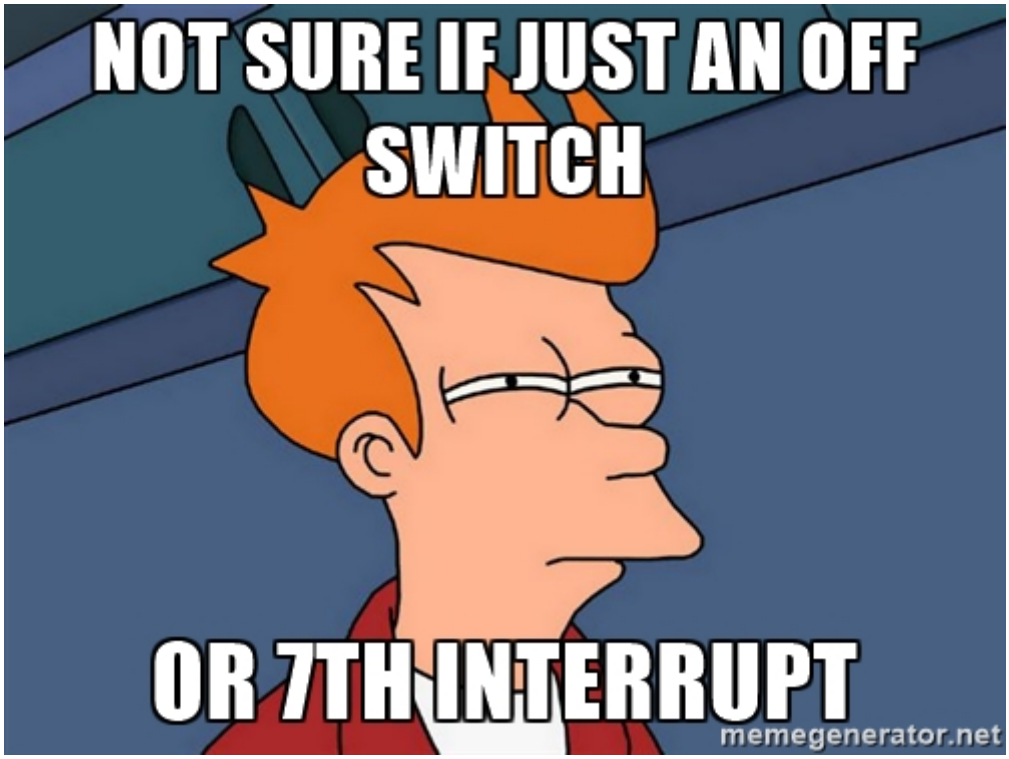


Problem 3

```
1           ORG 0
2           LJMP MAIN
3
4           /* Timer 0 Interrupt */
5           ORG 000B
6           CPL P1.1
7           RETI
8
9           /* Timer 1 Interrupt */
10          ORG 001B
11          CPL P1.2
12          RETI
13
14          /* Main */
15          ORG 30H
16  MAIN:   MOV TMOD, _num_12H          ; T0M2 & T1M1
17          MOV P1, _num_0              ; P1 = output port
18          ACALL WAVE0
19          ACALL WAVE1
20
21          MOV IE, _num_8AH
22  REPEAT: SJMP REPEAT
23
24  ; 5kHz => 1/5000 s = .0002 s => 200 microseconds
25  ; 200 us => 184 ticks PER PERIOD. Assuming a 50%
26  ; duty cycle. So actual value is 184/2 = 92 ticks
27  WAVE0:  MOV TH0, #-92
28          SETB TR0
29          RET
30
31  ; 500 Hz => .002 s => 2000 us
32  ; (FFFF - x + 1)(1.085) = 2000
33  ; x = FFFF - 1842 => E7 BD
34  WAVE1:  MOV TH1, _num_0E7H
35          MOV TL1, _num_0BDH
36          SETB TR1
37          RET
38
39
40          END
```

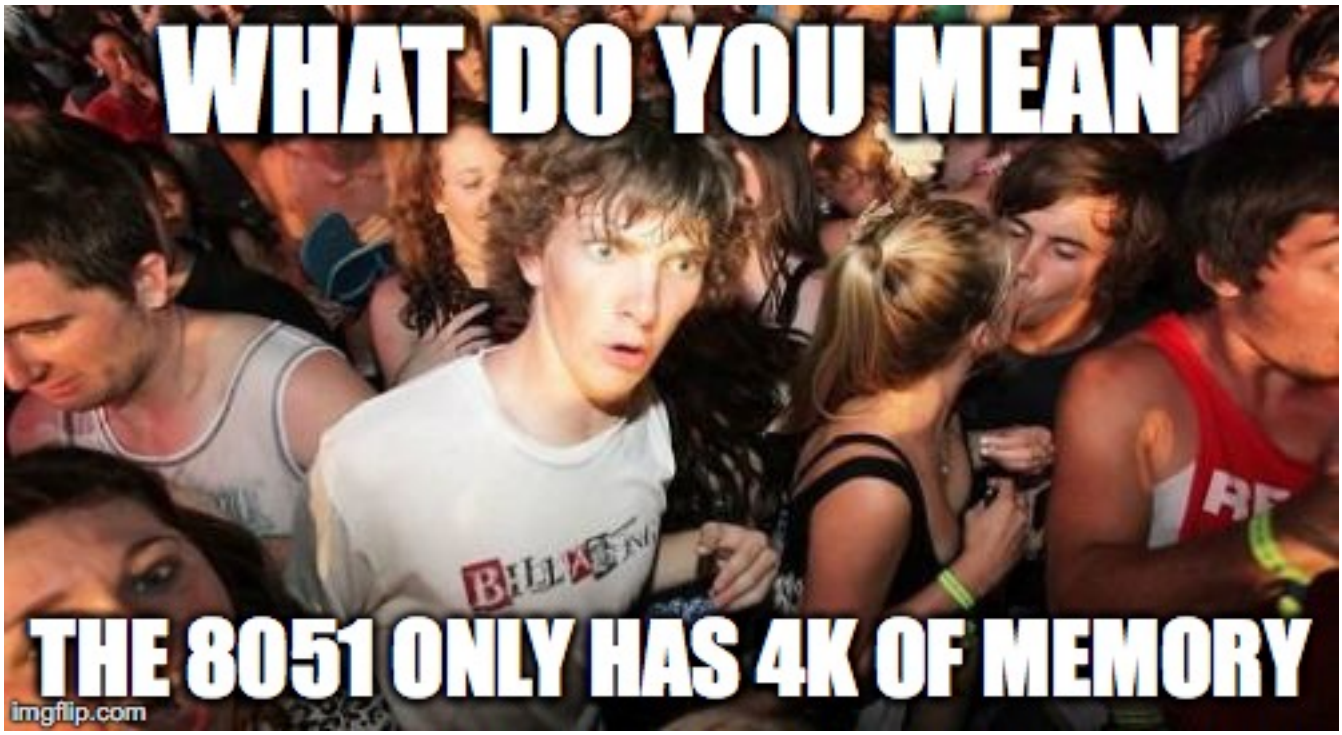
Problem 4

```
1           ORG 0H
2           LJMP MAIN
3
4           /* Interrupts */
```




```
5           ORG 001BH
6           LJMP SWAVE
7           RETI
8
9           /* Main */
10          ORG 30H
11  MAIN:   MOV R7, _num_1D
12          MOV P1, _num_0              ; Make P1 an output port
13          MOV IE, _num_88H            ; Enable interrupt on T1
14
15  LOOP:   MOV P2, R7              ; R7 holds the value of the div series
16          ACALL INFSER
17          SJMP LOOP
18
19  ; code for the square wave interrupt
20  ; determines via R6 what the last wave was
21  SWAVE:  MOV TMOD, _num_20H  ; Timer 2, mode 2
22
23          CJNE R6, _num_70, SKIP2LOW
24          ACALL HWAVE
25          SJMP NEXT
26
27  SKIP2LOW:
28          ACALL LWAVE
29
30  NEXT:   SETB TR1
```

```
31          RET
32
33  ; the wave has a duty cycle of 70%
34  ; so the high wave needs a high wave of 7kHz
35  ; & kHz => 7000 Hz => 1.42^-4 s => 142 us => 130 ticks
36  ; 130.8 *.7 = 92 ticks for 70% duty cycle of a 7kHz wave
37  HWAVE:
38          SETB P1.1
39          MOV TH1, #-92
40          MOV R6, _num_70             ; Just to keep track of what our last
        value was
41          RET
42
43  ; so, exact same reasoning as before, we need 130 ticks for
44  ; the period of the wave, but now we just subtract 92 from 130
45  ; so, 130 - 92 = 38
46  LWAVE:
47          CLR P1.1
48          MOV TH1, #-38
49          MOV R6, _num_30
50          RET
51
52  ; computes infinite divergent series incrementally
53  ; uses R7
54  INFSER:
55          MOV A, R7
56          ACALL TWOCOMP
57          ACALL ISNEGATIVE
58
59          JC RETURN
60          INC A
61
62  RETURN: MOV R7, A
63          RET
64
65
66  ; returns two's compliment of A
67  TWOCOMP:
68          CPL A
69          ADD A, _num_1D
70          RET
71
72
73  ; returns 1 in C if B is negative
74  ISNEGATIVE:
75          CJNE A, _num_10000000B, NEXT2
76  NEXT2:  CPL C
77          RET
78
79
80          END
```


WHAT DO YOU MEAN
THE 8051 ONLY HAS 4K OF MEMORY
imgflip.com

Paradigms and Applications of the Arduino Uno

Computer Engineering 3150 Research Paper

Illya Starikov

1 Introduction

The Arduino is an open-sourced family of microcontrollers aimed at not just the specialist, but also the hobbyist or enthusiast. With a multi-platform[1] IDE (Integrated Development Environment) provided and maintained by Arduino, code can be written and deployed within seconds, making it easy for anyone to get started. A "hello world"-esque program (cause an led light to flash on and off) can be implemented in only 12 lines of code[2] [**?**].

```
int ledPin = 10;

void setup() {
    pinMode(ledPin, OUTPUT);
}

void loop() {
    digitalWrite(ledPin, HIGH);
    delay(1000);
    digitalWrite(ledPin, LOW);
    delay(1000);
}
```

Anyone familiar with C or C++ can get started by briefly skimming documentation. However, the flexibility of and power of Arduino's microcontrollers is why professionals use it.

What makes Arduino so powerful is its ease of expandability, a process known as shielding. Such shields can be provide additional functionality, such as: [**?**]

- Bluetooth
- Internet connectivity via WiFi or Ethernet
- Sound and speech synthesizer
- Allowing a stack of Arduino boards to communicate

This readily available power of any other board makes Arduino ideal for any project. And if the functionality is not there, producing one is fairly simple — design a PCB (print circuit board) and mount it.

As mentioned prior, Arduino open sources everything; by everything, this covers to code, schematics, and designs. Because of the open source license (Creative Commons Attribution Share-Alike), it is possible to entirely replicate the board[3]. This is actually a common occurrence, giving rise to Arduino "clones", such as the Aliexpress and Teensy — of course, the Arduino Uno (the topic of this paper) is also a "clone"[4] of ATmega328.

1.1 The Arduino Uno

The Arduino Uno is an 8-bit microcontroller with 32 kB of flash storage and 2 kB of RAM (Random Access Memory). Because it operates at 5 V, it can be charged through the provided power jack or a typical 5 V charging block (e.g. a typical phone charger), or a 7 V - 12 V input pin.[5]. As for as analog controls and peripherals, the Uno has "14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button" [**?**]. Another great feature is the price: $29.95.

Because of its open-sourcing and given the 6 years to mature [**?**], the Uno has gather quite a community on the internet. The forums, hosted on the Arduino website, are active with some threads reaching up to 100 000 posts. Topics include [**?**]:

- Robotics ("Platforms, motors, algorithms and sensors to conform robotics projects")

[1] Supported on MAC OS X, Linux, and Windows.

[2] Excluding `int main()` and preprocessor directives.

[3] Excluding the name Arduino or Uno

[4] Based on the technology of the ATmega328.

[5] However, the pin "bypasses the regulator, and can damage your board." It is not advised charging anywhere out of the recommended range

Figure 1: The Arduino Uno Microcontroller [**?**].

- Home Automation and Networked Objects (Internet of Things)
- E-Textile and Craft ("flexible conductive materials, electroluminiscent materials")
- Device Hacking ("reappropiation of devices, tricks of the trade")

2 Applications of The Arduino Uno

The Arduino Uno's flexibilty makes it pragmatic to be used in a wide array of industries; three of these industries are Education, Healthcare, and Home Automation.

2.1 Education

In the education market, there are many factors that are needed to accounted for in adopting new technology, and the Arduino Uno usually accommodates for these factors, typically making it an ideal candidate. The two most important factors are affordability and ease of use.

When considering purchasing equipment, price is the most important element. Students need to have the ability to experiment with technology without fear of breaking it; not only that, but there is maintenance, replacement and installation costs associate with it. Fortunately, the Uno fits this niche. As aforementioned, the Uno is competitively at $29.95. The Arduino Starter Kit, which not only contains the Uno, but many sample projects that gives a hand-on experience with it, costs $99.99 Lastly, the multiple shields that can be purchased range form $1.00 to little over $100, but most usually lower than $40.

For such a competitive price, the Uno still manages to have a lot of thought and consideration with regards to the user. The cross-platform IDE ensures frictionless setup; programming the microcontroller can begin immediately after installation. The online documentation and forums are house enough information to get started with any kind of project. This makes the Arduino Uno the ideal microcontroller to get started with.

This was the mentality behind a team of researches from Federal University of Rio Grande do Sul Porto Alegre-Rs and Criciúma-Sc from Brazil. In a recent attempt to further engage electrical engineering students, the group built a "remote educational experiment of a current motor (DC motor) control speed, which allow the students to interact with the control systems P, PI, PID applied to the rotation control of the remote mode engine" [**?**].

Instead of being constrained to the labs, where traditional equipment is kept, the researches moved the lab online where it is accessible from anywhere at anytime. The user can, through an online Java application, access the hardware in the lab. The user interaction is still natural, because the application is tethered to a camera that sends a continuous live video that will be streamed to the user's browser. The user can send live commands to the device, and get instantaneous feedback. As the user is interacting with the application, the commands are being sent to the Arduino microcontroller, which is not only controlling the engine, but is acting as a PID control machine.

Figure 2: The Java applet, with the livestream on the left and PID controls on the right [**?**].

A PID controller (proportional-integral-derivative) is a self-correcting feedback mechanism that asymptotically approaches a stable point, also referred to as a setpoint. The error correcting equation, as a function of time, is a integro-differential equation as such:

$$u(t) = k_p e(t) + K_i \int_0^t e(\tau)\ d\tau + K_d \frac{d\ e(t)}{dt}. \quad (1)$$

It reads "as the error tolerance at a set time $u(t)$ is calculated by the current error $(k_p e(t))$ with the potential error that needs to be considered $(K_i \int e(\tau)\ d\tau)$, while taking away the excess error $(K_d \frac{d\ e(t)}{dt})$". As this function changes with respect to time, the system (in this case, the DC motor) calculates the error and asymptotically reaches a setpoint. In this case, the PID controller tries to regulate the voltage and bring the engine to a steady state. And the Uno does all of the calculations in real time and lets the user see at a glance what the each state (P, I, or D) is at, and what the setpoint is.

And at the end of the session, a majority of the participants enjoyed the project. The original goal of further engaging students outside of the classroom was achieved; the project was an overwhelming success.

Experiment contributed to better understanding the concepts developed in the discipline of control systems?			
Excellent	Good	Regular	Bad
43%	36%	11%	10%
What is your impression about developing experiments remote related to control systems?			
Excellent	Good	Regular	Bad
40%	38%	14%	8%
In your opinion greater learning control systems in the development of activities linked to experiments remote or simulations?			
Simulation	Real experiments	Both	Combination
5%	27%	44%	24%

Table 1: Satisfaction of the PID controller project [**?**].

However, this is only one of the many interesting projects — all of which don't have to be a strict application to Electrical Engineering. A group of researches from IBM decided to target a younger age-group: 9th–12th grade students. Seeing as this was the age group that had a rapid loss of interest in STEM (Science, Technology, Engineering, and Mathematics) based fields, the IBM group realized this was a delicate experiment. The experiments could not be too technical, because of the lack of knowledge and experience in high schools, but could not be mundane; all of this had to be considered while trying to teach students the basics of electronics and programming.

The students received hands-on experience by doing a simplified experiment: programming a blinking LED light; and then further expanding this concept by placing the LED lights to a bag and making text light up. The researches said the students "were very engaged to and excited about open source, which is affordable and accessible. The students conducted the experiments with no major issues, except the lack of time" [**?**].

After the experiment, there were presentations of the more advanced things that could be done. Some of these were: a Twitter client that projects negative or positive feelings in regards to an debated topic, a toy car that could be controlled with a PlayStation controller, and even a quadcopter. All of these demonstrations were powered with the Arduino Uno and a couple of shields.

The IBM group is continuing to expand this project in the coming years and hoping to educate more high school students about the power of programming, electronics, and open source.

2.2 Healthcare

As previously mentioned, the Arduino Uno is based on the ATmega328P, the high performance microcontroller from Atmel. However, performance is not exclusively what the team at Arduino focused on; they weighed performance with reliability. This blend of high performance and reliability makes the Arduino Uno a suitable microcontroller to be used in healthcare. Two graduate students at the Institute of Biophysics and Biomedical Engineering in Portugal have found an interesting use for it: hand rehabilitation therapy.

Because modern, technological approaches to hand rehabilitation therapy have been proven to be quite expensive and needed heavy technical experience and support, the team decided to make an affordable and simple system that can used at the comfort of your own home. It's called a "hand

therapist".

The team started with Unity3D, a popular game engine, to simulate an environment on a personal computer. The environment is essentially a canvas with a working hand that be controlled with a series of commands in the xyz-plane, making it easy to wire up a working hand to a microcontroller to interpret said commands; and that's exactly what the team did.

The Uno interprets the movement and synchronously plots the movement on the computer. Then the user can "grab, hold, transport and drop a cube in several increasingly difficult puzzle levels". Not only does this make hand therapy much easier, but more affordable (less than $600) [**?**].

Figure 3: The rehabilitation system [**?**].

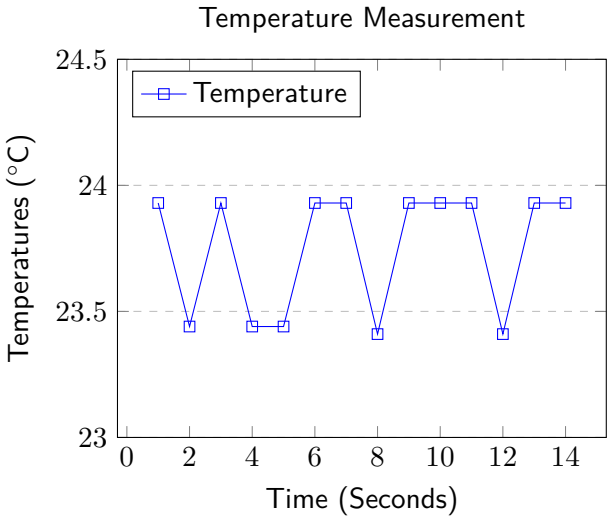


Figure 4: The system usually keeps the room within a 1°C.

But complicated hand therapy is not the only application for the Uno in healthcare. Another research team in India have managed to hook up a microcontroller to monitor patient heartrate through the course of the day with an Uno and a bright LED light and transmit the information to the doctor at a fixed schedule, making it easy to regularly check up on patients [**?**].

Thanks to the Uno, not only is healthcare becoming more reliable, but less expensive too.

2.3 Home Automation

Today's technology is significantly more advanced then ever; devices are getting smarter, better at communicating, and even more efficient. Something that came out of this "technological revolution" is the idea of a smarthome — powered by the IoT (Internet of Things). A smarthome is nothing more than a home "that constitutes advanced automation systems to provide the occupants with sophisticated monitoring and controlling over the different functions of the building" [**?**].

Home automation is a very broad topic, ranging from things such as temperature monitoring to motion tracking to light automation; fortunately, the Arduino Uno can do all of the above. Researchers from MIT managed to mount input/output blocks consisting of "PIR (Passive Infra-Red) motion sensor, LDR (Light Dependent Resistor) and an LM35 temperature sensor as inputs and some lamps" and get the blocks communicating.

The PIR motion sense was to detect motion; not only could it turn lights on/off (via the LDR) if anyone was present, but it could act as a security system as well. If there was any suspicious movement during irregular hours, an alarm is buzzed. This buzzing acts as a warning system to any intruders or potential wild animals. The LM35 is not only to monitor temperature, but to enable and Air Conditioning or fans when a certain temperature threshold has been reached.

What makes these sensor different than switches on the wall or a regular AC controller is these can be monitored, synchronously, virtually. The current front-end is a MATLAB GUI (Graphical User Interface). Instead of having to traverse the typical household, the entire system is a few clicks away. This makes the entire household not only systematic and tidy, but more cost and energy efficient; and because it runs on a 5 volt Arduino Uno, it is using significantly less power than the AC unit, lighting, and security systems[6].

[6] However, the energy efficiency should be traded for one's safety.

3 Conclusion

As laid out previously, the Arduino Uno has incredible uses in three major fields: Education, Healthcare, and Home Automation. The Uno's ease of use makes it an ideal electronic to get interest in Engineering-based fields, while it's price makes a candidate to be used in the classroom without fear of breaking hardware. The reliabilty and processing power makes it a great aid in healthcare, such as "hand therapy" or monitoring a patient status. Also, the Uno is great at home automation, where it can simultaneously monitor movement, temperature, and lighting based on room occupancy. Looking forward, one can surely expect many more great microcontrollers from Arduino that will get be in many, interesting ways.

References

[1] M. McRoberts, *Beginning Arduino*.

[2] "Arduino - ArduinoShields", *Arduino.cc*, 2016. [Online]. Available: https://www.arduino.cc/en/Main/arduinoShields. [Accessed: 11- Sep- 2016].

[3] "Arduino - ArduinoBoardUno", *Arduino.cc*, 2016. [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno. [Accessed: 11- Sep- 2016].

[4] "Arduino Uno", *Wikimedia*, 2016. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/9/90/Barbone_Arduino_Uno.jpg. [Accessed: 11- Sep- 2016].

[5] M. Banzi, "Arduino Blog – Dinner is Ready", *Blog.arduino.cc*, 2010. [Online]. Available: https://blog.arduino.cc/2010/09/24/dinner-is-ready/. [Accessed: 11- Sep- 2016].

[6] "Arduino Forum - Index", *Forum.arduino.cc*, 2016. [Online]. Available: http://forum.arduino.cc. [Accessed: 11- Sep- 2016].

[7] J. M. Neto, S. Paladini, C. E. Pereira and R. Marcelino, "Remote educational experiment applied to electrical engineering," *Remote Engineering and Virtual Instrumentation (REV)*, 2012 9th International Conference on, Bilbao, 2012, pp. 1-5.

[8] L. M. Herger and M. Bodarky, "Engaging students with open source technologies and Arduino," *Integrated STEM Education Conference (ISEC)*, 2015 IEEE, Princeton, NJ, 2015, pp. 27-32.

[9] R. Lipovský and H. A. Ferreira, "Hand therapist: A rehabilitation approach based on wearable technology and video gaming," *Bioengineering (ENBENG)*, 2015 IEEE 4th Portuguese Meeting on, Porto, 2015, pp. 1-2.

[10] P. A. Pawar, "Heart rate monitoring system using IR base sensor & Arduino Uno," *IT in Business, Industry and Government (CSIBIG)*, 2014 Conference on, Indore, 2014, pp. 1-3.

[11] A. R. Behera, J. Devi and D. S. Mishra, "A comparative study and implementation of real time home automation system," *2015 International Conference on Energy Systems and Applications*, Pune, 2015, pp. 28-33.

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; we[1] decided to apply this new-found excitement to implement a retro game from the 70s: Space Invaders.

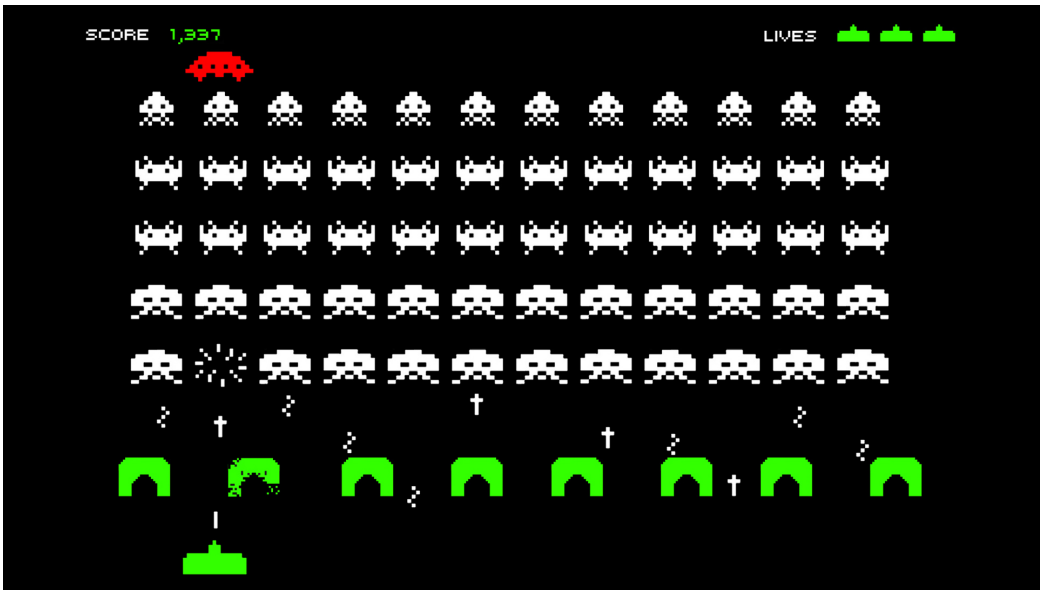


Figure 1 – The original space invaders.

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).

[1] Illya.

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 possible[2]. 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.
- q Quits the game
- Space 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 $C4$,

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

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

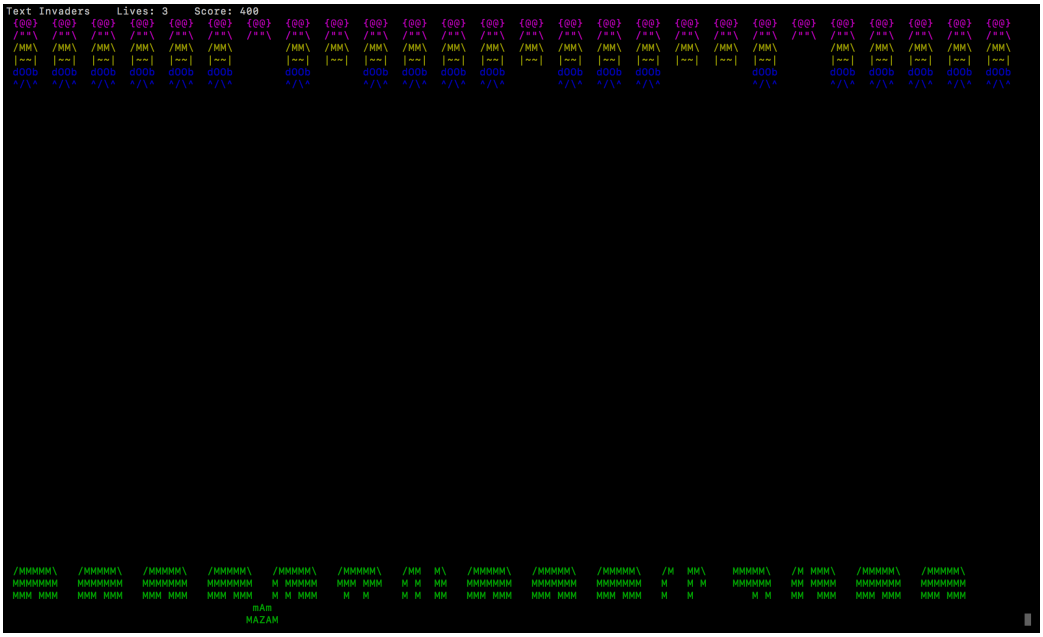


Figure 2 – Text Invaders, our version of space invaders..

$$\frac{1/4 \,\text{Oscillator Frequency}}{\text{Note Frequency}} = \frac{1/4 \cdot 7.373\,\text{Hz}}{261.63\,\text{Hz}} = 7.045\,\text{ms} \tag{1}$$

To produce the music, we had a function that took in which note and the type of note (16^{th}, 8^{th}, 4^{th}, half or whole note). The function multiplied the base length of a note by a constant depending on the type of note. A 16^{th} note would be multiplied by 1, an 8^{th} 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

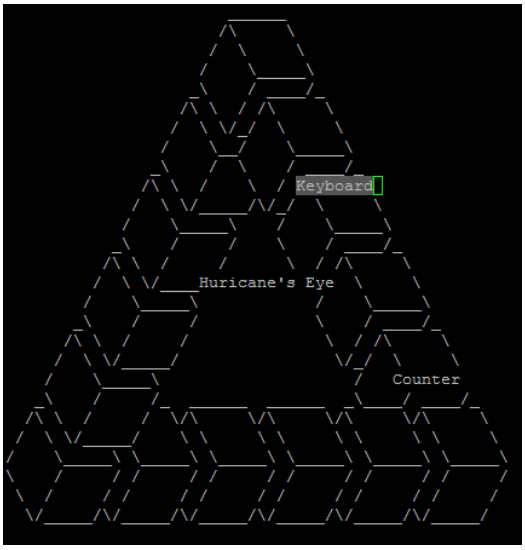


Figure 3 – The pen-rose style menu.

1. Counter
2. Hurricane's Eye
3. Keyboard

respectively. From there the user has the option to 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

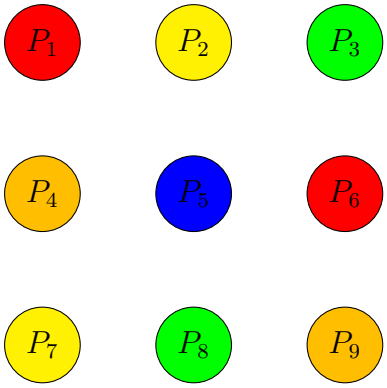


Figure 4 – The Main Menu.

function that lets you display any number. We used the data sheet for the 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 $P_1 \cdots P_8$ 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

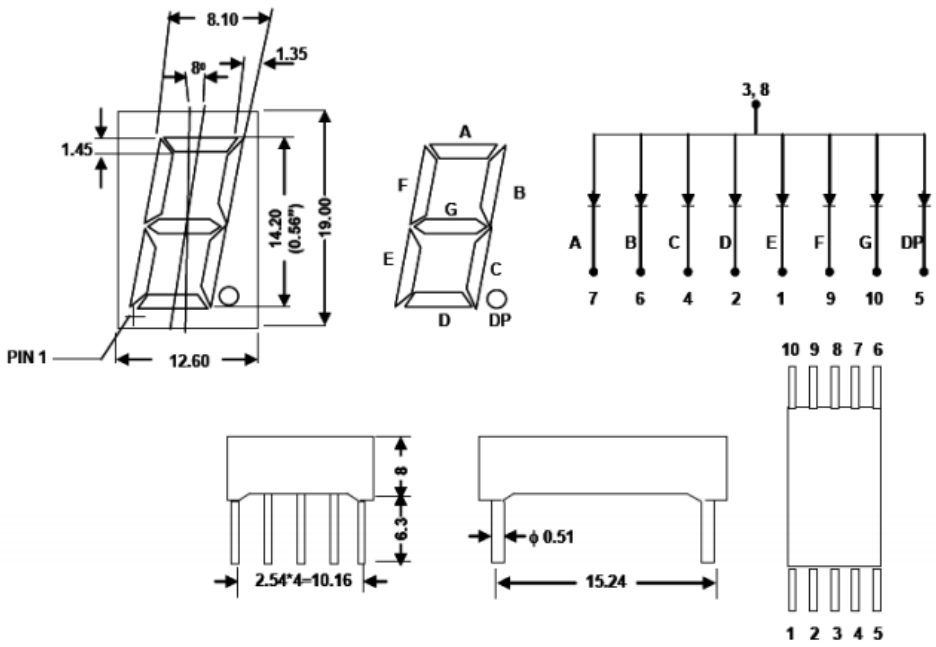


Figure 5 – The schematic for the seven segment display.

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`s[4], 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 game[5] 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, `slsnif`[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.

[4]Since `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.

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

[6]Can be found at `https://sourceforge.net/projects/slsnif/`.

[7]Can 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      TL0=-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 $(\mathrm{FFFF}_{16} - \mathrm{0E66}_{16} + 1) \times 1.085\,\mu s \approx 1\,ms$. 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!

MATH

Mathematics

Calculus I: Single-Variable Calculus

Illya Starikov

August 6, 2025

Contents

num_1!

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 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** $a_0, a_1, \ldots, a_n$ 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 nth-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.

Algebriac 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 $b \neq q$. 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 $e \approx 2.71828\ldots$ 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 $\cos x$; 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 $e = 2.718281828459\ldots$.

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 is $y(x_1) \neq f(x_2)$ 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

$$f^{-1}(f(x)) = x \qquad \text{and} \qquad f(f^{-1}(y)) = y,$$

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

$$b^x = e^{x \ln b}, \text{for all } x \qquad \text{and} \qquad \log_b x = \frac{\ln x}{\ln b}, \text{for } x > 0$$

More generally, if c is a positive real number with $c \neq 1$, then

$$b^x = c^{x \log_c b}, \text{for all } x \qquad \text{and} \qquad \log_b x = \frac{\log_c x}{\log_c b}, \text{for } x > 0$$

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

$$\sin\theta = \frac{y}{r} \qquad \cos\theta = \frac{x}{r} \qquad \tan\theta = \frac{y}{x} \tag{1}$$

$$\cot\theta = \frac{x}{y} \qquad \sec\theta = \frac{r}{x} \qquad \csc\theta = \frac{r}{y} \tag{2}$$

$$\tag{3}$$

Trigonometric Identities

Reciprocal Identities

$$\tan\theta = \frac{\sin\theta}{\cos\theta} \qquad \cot\theta = \frac{1}{\tan\theta} = \frac{\cos\theta}{\sin\theta} \tag{4}$$

$$\csc\theta = \frac{1}{\sin\theta} \qquad \sec\theta = \frac{1}{\cos\theta} \tag{5}$$

Pythagorean

$$\sin^2\theta + \cos^2\theta = 1 \qquad 1 + \cot^2\theta = \csc^2\theta \qquad \tan^2\theta + 1 = \sec^2\theta \tag{6}$$

Double- and Half-Angle Formulas

$$\sin 2\theta = 2\sin\theta\cos\theta \qquad \cos 2\theta = \sin^2\theta - \sin^2\theta \tag{7}$$

$$\cos^2\theta = \frac{1+\cos 2\theta}{2} \qquad \sin^2\theta = \frac{1-\cos 2\theta}{2} \tag{8}$$

Period of Trigonometric Function

The function $\sin\theta$, $\cos\theta$, $\sec\theta$, and $\csc\theta$ have a period of 2π

$$\sin(\theta + 2\pi) = \sin\theta \qquad \cos(\theta + 2\pi) = \cos\theta \tag{9}$$

$$\sec(\theta + 2\pi) = \sec\theta \qquad \csc(\theta + 2\pi) = \csc\theta \tag{10}$$

for all θ in the domain.
The functions $\tan\theta$ and $\cot\theta$ have a period of π:

$$\tan(\theta + \pi) = \tan\theta \qquad \cot(\theta + \pi) = \cot\theta \tag{11}$$

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

$$\lim_{x \to a} f(x) = L$$

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

$$\lim_{x \to a^+} f(x) = L \tag{12}$$

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

$$\lim_{x \to a^-} f(x) = L \tag{13}$$

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 $\lim_{n um_1} f(x) = L$ if and only if $\lim_{n um_1} = \lim_{n um_1} = L$.

1.3 Techniques For Computing Limits

Limits of Linear Functions

Let a, b, and m be real numbers. For Linear functions $f(x) = mx + b$,

$$\lim_{x \to a} f(x) = f(a) = ma + b \tag{14}$$

Limit Laws

Assume $\lim_{n um_1} f(x)$ and $\lim_{n um_1} g(x)$ exist. The following properties hold, where c is a real number, and $m > 0$ and $n > 0$ are integers.

1 Sum $\lim_{n um_1} [f(x) + g(x)] = \lim_{n um_1} f(x) + \lim_{n um_1} g(x)$

2 Difference $\lim_{n um_1} [f(x) - g(x)] = \lim_{n um_1} f(x) - \lim_{n um_1} g(x)$

3 Constant Multiple $\lim_{n um_1} [c\, f(x)] = c \lim_{n um_1}$

4 Product $\lim_{n um_1} [f(x)g(x)] = \left[\lim_{n um_1} f(x)\right] \left[\lim_{n um_1} g(x)\right]$

5 Quotient $\lim_{n um_1} \left[\frac{f(x)}{g(x)}\right] = \frac{\lim_{n um_1} f(x)}{\lim_{n um_1} g(x)}$, provided $\lim_{n um_1} g(x) \neq 0$

6 Power $\lim_{n um_1} [f(x)]^n = \left[\lim_{n um_1} f(x)\right]^n$

7 Fractional Power $\lim_{n\!um_1} [f(x)]^{n/m} = \left[\lim_{n\!um_1} f(x)\right]^{n/m}$, 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: $\lim_{n\!um_1} p(x) = p(a)$
- Rational functions: $\lim_{n\!um_1} \frac{p(x)}{q(x)} = \frac{p(a)}{q(a)}$, provided $q(a) \neq 0$

Limit Laws For One-Sided Limits

Laws 1–6 hold with $\lim_{n\!um_1}$ replaced by $\lim_{n\!um_1}$ or $\lim_{n\!um_1}$. Law 7 is modified as follows, assume $m > 0$ and $n > 0$ are integers.

7 Fractional Power

- $\lim_{n\!um_1} [f(x)]^{n/m}$, provided $f(x) \geq 0$, for x near a with $x > a$, if m is even and n/m is reduced to lowest terms
- $\lim_{n\!um_1} [f(x)]^{n/m}$, 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 $\lim_{n\!um_1} f(x) = \lim_{n\!um_1} h(x) = L$, then $\lim_{n\!um_1} g(x) = L$.

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

$$\lim_{x \to a} f(x) = \infty \tag{15}$$

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

$$\lim_{x \to a} f(x) = -\infty \tag{16}$$

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 $\lim_{n um_1} f(x) = \infty$. The one-sided infinite limit $\lim_{n um_1} = -\infty$, $\lim_{n um_1} f(x) = \infty$, and $\lim_{n um_1} = -\infty$ are defined analogously.

Vertical Asymptote

If $\lim_{n um_1} f(x) = \pm\infty$, $\lim_{n um_1} = \pm\infty$ or $\lim_{n um_1} f(x) = \pm\infty$, 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

$$\lim_{x \to \infty} f(x) = L \tag{17}$$

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, $\lim_{n um_1} f(x) = M$, 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

$$\lim_{x \to \infty} f(x) = \infty \tag{18}$$

The limits $\lim_{n um_1} = -\infty$, $\lim_{n um_1} = \infty$, and $\lim_{n um_1} = -\infty$ are defined similarly.

Limit of Infinity at Powers and Polynomials

Let n be a positive integer and let p be the polynomial $p(x) = a_n x^m + a_{n-1}x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$, where $a_n \neq 0$.

1. $\lim_{n um_1} x^n = \infty$ when n is even.
2. $\lim_{n um_1} x^n = \infty$ and $\lim_{n um_1} x^n = -\infty$ when n is odd.
3. $\lim_{n um_1} \frac{1}{x^n} = \lim_{n um_1} x^{-n} = 0$
4. $\lim_{n um_1} p(x) = \infty or -\infty$, 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

$$p(x) = a_m x^m + a_{m-1} + \cdots + a_2 x^2 + a_1 x + a_0 \tag{19}$$

and

$$q(x) = b_m x^m + b_{m-1} + \cdots + b_2 x^2 + b_1 x + b_0 \tag{20}$$

with $a_m \neq 0$ and $b_n \neq 0$.

1. If $m < n$ then $\lim_{n um_1} f(x) = 0$, and $y = 0$ is a horizontal asymptote of f.
2. If $m = n$, then $\lim_{n um_1} f(x) = a_m/b_n$, and $y = a_m/b_n$ is a horizontal asymptote.
3. If $m > n$, then $\lim_{n um_1} f(x) = \infty$ or $-\infty$, and f has no horizontal asymptote.
4. If $m = n+1$, then $\lim_{n um_1} f(x) = \infty$ 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:

$$\lim_{x\to\infty} e^x = \infty \qquad \text{and} \qquad \lim_{x\to-\infty} e^x = 0 \tag{21}$$

$$\lim_{x\to\infty} e^{-x} = \infty \qquad \text{and} \qquad \lim_{x\to-\infty} e^x = 0 \tag{22}$$

$$\lim_{x\to0^+} \ln x = -\infty \qquad \text{and} \qquad \lim_{x\to\infty} \ln x = \infty \tag{23}$$

7.7 Hyperbolic Functions

Calculus II

0.1 Hyperbolic Functions

0.1.1 Hyperbolic Functions

Hyperbolic Cosine

$$\cosh x = \frac{e^x + e^{-x}}{2} \tag{1}$$

Hyperbolic Sine

$$\sinh x = \frac{e^x - e^{-x}}{2} \tag{2}$$

Hyperbolic Tangent

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3}$$

Hyperbolic Cotangent

$$\coth x = \frac{\cosh x}{\sinh x} = \frac{e^x + e^{-x}}{e^x - e^{-x}} \tag{4}$$

Hyperbolic Secant

$$x = \frac{1}{\cosh x} = \frac{2}{e^x + e^{-x}} \tag{5}$$

Hyperbolic Cosecant

$$x = \frac{1}{\sinh x} = \frac{2}{e^x - e^{-x}} \tag{6}$$

0.1.2 Hyperbolic Identities

$\cosh^2 x - \sinh^2 x = 1$	$\cosh(-x) = \cosh x$
$1 - \tanh^2 x =^2 x$	$\sinh(-x) = -\sinh x$
$\coth^2 x - 1 =^2 x$	$\tanh(-x) = -\tanh x$
$\coth(x+y) = \cosh x \cosh y + \sinh x \sinh y$	
$\sinh(x+y) = \sinh x \cosh y + \cosh x \sinh y$	
$\cosh 2x = \cosh^x + \sinh^2$	$\sinh 2x = 2 \sinh x \cosh x$
$\cosh^2 x = \frac{\cosh 2x+1}{2}$	$\sinh^2 x = \frac{\cosh 2x-1}{2}$

0.1.3 Derivatives and Integral Formulas

1. $\frac{d}{dx}(\cosh x) = \sinh x \quad \Rightarrow \quad \int \sinh x \; dx = \cosh x + C$
2. $\frac{d}{dx}(\sinh x) = \cosh x \quad \Rightarrow \quad \int \cosh x \; dx = \sinh x + C$
3. $\frac{d}{dx}(\tanh x) =^2 x \quad \Rightarrow \quad \int^2 x \; dx = \tanh x + C$
4. $\frac{d}{dx}(\coth x) = -^2 x \quad \Rightarrow \quad \int^2 x \; dx = -\coth x + C$
5. $\frac{d}{dx}(x) = -x \tanh x \quad \Rightarrow \quad \int x \tanh x \; dx = -x + C$
6. $\frac{d}{dx}(x) = -x \coth x \quad \Rightarrow \quad \int x \coth x \; dx = -x + C$

0.1.4 Integrals of Hyperbolic Functions

1. $\int \tanh x\,dx = \ln\cosh x + C$
2. $\int \coth x\,dx = \ln|\sinh x| + C$
3. $\int x\,dx = \tan^{-1}|\sinh x| + C$
4. $\int x\,dx = \ln|\tanh(x/2)| + C$

0.1.5 Inverses of the Hyperbolic Functions Expressed as Logarithms

$\cosh^{-1} x = \ln(x + \sqrt{x^2-1})$ $(x \geq 1)$ $\quad {}^{-1}x = \cosh^{-1}\frac{1}{x}$ $(0 < x \leq 1)$
$\sinh^{-1} x = \ln(x + \sqrt{x^2+1})$ $\quad {}^{-1}x = \sinh^{-1}\frac{1}{x}$ $(x \neq 0)$
$\tanh^{-1} x = \frac{1}{2}\ln(\frac{1+x}{1-x})$ $(|x| < 1)$ $\quad \coth^{-1} x = \tanh^{-1}\frac{1}{x}$ $(|x| > 1)$

0.1.6 Derivatives of the Inverse Hyperbolic Functions

$\frac{d}{dx}(\cosh^{-1} x) = \frac{1}{\sqrt{x^2-1}}$ $(x > 1)$ $\quad \frac{d}{dx}(\sinh^{-1} x) = \frac{1}{\sqrt{x^2+1}}$
$\frac{d}{dx}(\tanh^{-1} x) = \frac{1}{1-x^2}$ $(|x| < 1)$ $\quad \frac{d}{dx}(\coth^{-1} x) = \frac{1}{1-x^2}$ $(|x| > 1)$
$\frac{d}{dx}({}^{-1}x) = -\frac{1}{x\sqrt{1-x^2}}$ $(0 < x < 1)$ $\quad \frac{d}{dx}({}^{-1}x) = -\frac{1}{|x|\sqrt{1+x^2}}$ $(x \neq 0)$

0.1.7 Integral Formulas

1. $\int \frac{dx}{\sqrt{x^2-a^2}} = \cosh^{-1}\frac{x}{a} + C,$ for $x > a$
2. $\int \frac{dx}{x^2+a^2} = \sinh^{-1}\frac{x}{a} + C,$ 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. $\int \frac{dx}{x\sqrt{a^2-x^2}} = -\frac{1}{a}{}^{-1}\frac{x}{a} + C,$ for $0 < x < a$
5. $\int \frac{dx}{x\sqrt{a^2+x^2}} = -\frac{1}{a}{}^{-1}\frac{|x|}{a} + C,$ for $x \neq 0$

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.

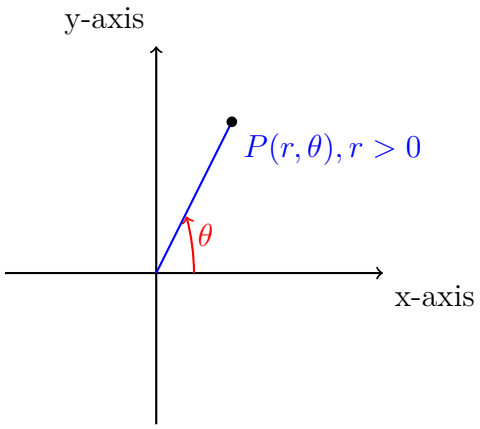
Polar Coordinates A polar coordinates P has the form (r, θ).

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,

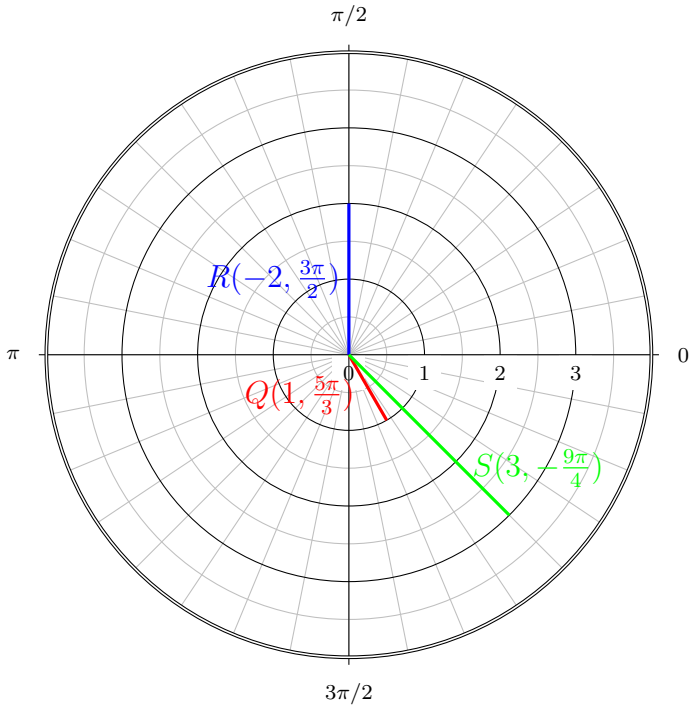
- $(r, \theta + 2\pi)$ represents the same point as (r, θ)
- $P(r, \theta)$ and $P'(-r, \theta)$ are reflections through the origin.

Examples

Graph the following points in polar coordinates:

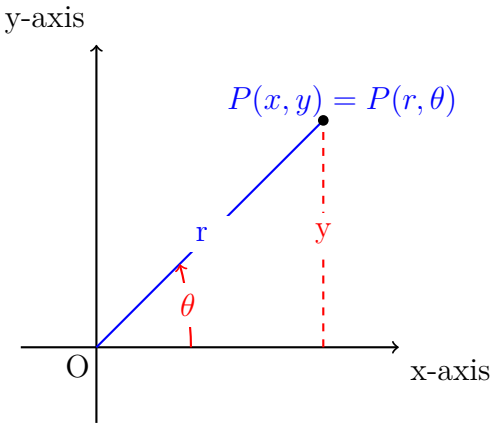
- $Q(1, \frac{5\pi}{3})$
- $R(-2, \frac{3\pi}{2})$
- $S(3, -\frac{9\pi}{4})$
  - Now give two alternative representations.

- $S'(3, \frac{1\pi}{4})$
- $S''(-3, -\frac{5\pi}{4})$



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

$$x = r\cos\theta \quad \text{and} \quad y = r\sin\theta \tag{1}$$

A point with Cartesian coordinates (x, y) has polar coordinates (r, θ), where

$$r^2 = x^2 + y^2 \quad \text{and} \quad \tan\theta = \frac{y}{x} \tag{2}$$

Examples

BE SURE TO GRAPH POINTS IN CARTESIAN FIRST.

Polar to Cartesian

Express the point with the following polar coordinates in Cartesian coordinates: $P(3, \frac{2\pi}{3})$

$$\begin{aligned} x &= r\cos\theta \\ &= 3\cos({}^{2\pi}/_{3}) \\ &= -3({}^{1}/_{2}) \\ &= -{}^{3}/_{2} \end{aligned} \qquad \begin{aligned} y &= r\sin\theta \\ &= 3\sin({}^{2\pi}/_{3}) \\ &= 3({}^{\sqrt{3}}/_{2}) \\ &= {}^{3\sqrt{3}}/_{2} \end{aligned}$$

Polar to Cartesian

Express the point with the following polar coordinates in Cartesian coordinates: $Q(e, -\frac{\pi}{4})$

$$\begin{aligned} x &= r\cos\theta \\ &= e\cos(-\pi/4) \\ &= e(\sqrt{2}/2) \\ &= e\sqrt{2}/2 \end{aligned} \qquad \begin{aligned} y &= r\sin\theta \\ &= e\sin(-\pi/4) \\ &= -e(\sqrt{2}/2) \\ &= -e\sqrt{2}/2 \end{aligned}$$

Cartesian To Polar

Express the point with the following Polar coordinates to Cartesian coordinates: $R(1, -1)$

$$\begin{aligned} r &= \sqrt{x^2+y^2} \\ &= \sqrt{1^2+(-1)^2} \\ &= \sqrt{2} \end{aligned} \qquad \begin{aligned} \tan\theta &= y/x \\ &= -1/1 \\ &= -1 \\ \theta &= -\pi/4 \text{ or } 7\pi/4. \end{aligned}$$

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: $S(1, \sqrt{3})$

$$\begin{aligned} r &= \sqrt{x^2+y^2} \\ &= \sqrt{(\sqrt{3})^2+(1)^2} \\ &= 2 \end{aligned} \qquad \begin{aligned} \tan\theta &= y/x \\ &= \sqrt{(3)}/1 \\ \theta &= \pi/3 \text{ or } 4\pi/3. \end{aligned}$$

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

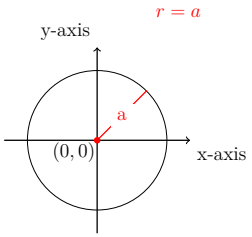
Convert the polar equation $r = 6\sin\theta$ to Cartesian coordinates and describe the corresponding graph.

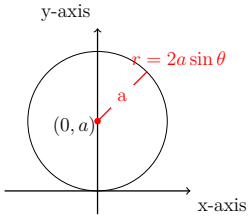
$$\begin{aligned} r^2 &= 6r\sin\theta && (3)\\ x^2 + y^2 &= 6y && (4)\\ 0 &= x^2 + y^2 - 6y && (5)\\ &= x^2 + (y^2 - 6y + 9) - 9 && (6)\\ &= x^2 + (y-3)^2 - 9 && (7) \end{aligned}$$

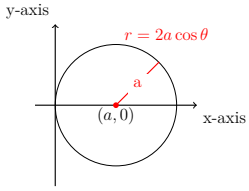
We recognize this to be the equation of a circle, centered at $(0, 3)$ at 3. We can also generalize this.

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 $(a, 0)$.



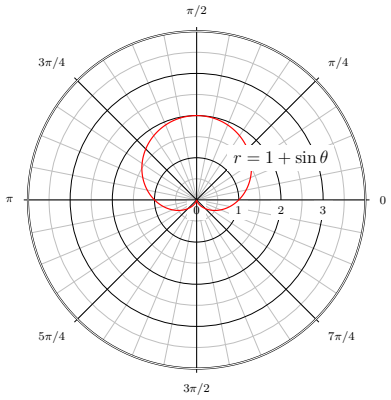




Graphing In Polar Coordinates

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 $r = 1 + \sin\theta$.

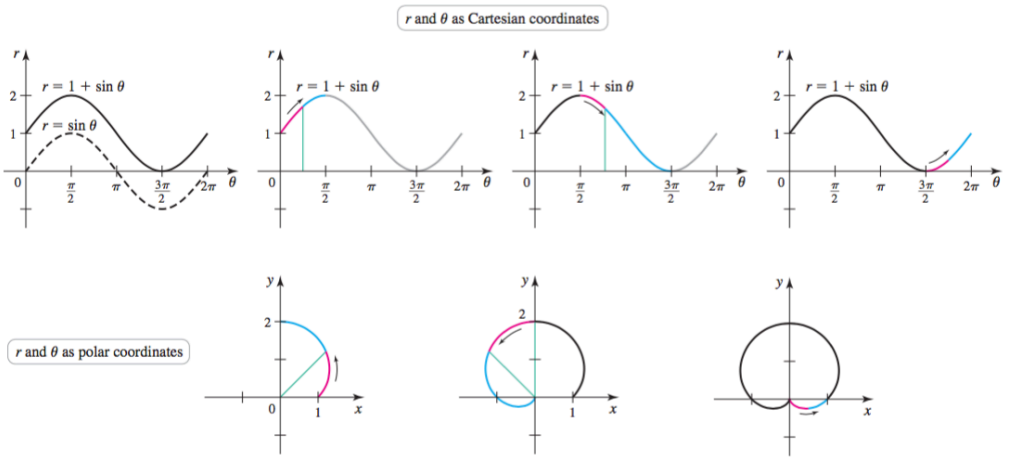


FIGURE 11.26

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 $(r, \theta$ is on the graph whenever $(-r, \theta) = (r, \theta + \pi)$ is on the graph.

MATH 1215 Practice Final

This is a closed-book, closed-notes exam. The only items you are allowed to use are writing implements. If you are caught cheating, you will receive a zero grade for this exam. The max number of points per question is indicated in square brackets after each question. The sum of the max points for all the questions is 61, but note that the max exam score will be capped at 60 (i.e., there is 1 bonus point, but you can't score more than 100%). You have exactly 60 minutes to complete this exam. Keep your answers clear and concise while complete. Best of luck.

1. Calculate the arc length of $\ln(\cos x)$ on $[0, \frac{\pi}{4}]$. Simplify. [7]

2. Calculate the following derivatives:

   (a) $\frac{d^2}{dx^2}(2\log_7(x-3))$ [2]

   (b) $\frac{d}{dx}(x^{\sec x})$ [4]

3. Classify which function has a faster rate of growth via limit methods **or** growth classes. If the limits are comparable, identify the growth factor M.

   (a) 1.00001^x and x^{40}. [1]

   (b) $\ln x^{18}$ and $\ln x$. [1]

   (c) $x \ln x$, $\ln^3 x$, and e^x. [2]

4. Evaluate the following indefinite integrals, simplification is optional (i.e. $\ln(1), \frac{2}{4}, 5^4$ are all acceptable forms).

   (a) $\int 18x^2 \ln x \, dx$ [4]

   (b) $\int \frac{2}{y^4\sqrt{y^2-25}} \, dy$. Assume θ to be in the first quadrant *at all times*[1]. [8]

   (c) $\int \frac{7}{y^2-9y-112} dy$ [5]

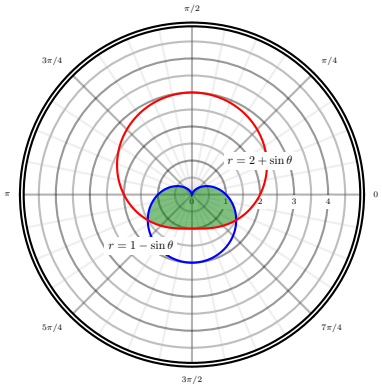
5. State the convergence or divergence of the following sequences and series. State all preconditions. Unless specified, use any method.

   (a) $\{-\frac{\sin n}{6n}\}$ [3]

   (b) $\sum_{n=2}^{\infty} \frac{1}{1+n\ln n}$ [5]

   (c) $\sum_{k=1}^{\infty} k \sin(\frac{1}{k})$ [6]

6. Set up the integral to find the area common to both $r = 2 + \sin\theta$ and $r = 1 - \sin\theta$ as shaded to the right. Do not simplify. **You do not need to evaluate the integral**. [5]



7. Set up the integral to find the pressure experienced by a circular plate with a radius of 2 that is submerged 6 meters below the water. This implies that the center of the plate is 8 meter below the water. Use ρ and g for pressure and acceleration. **You do not need to evaluate the integral**. [5]

8. Specify your answer as *True* or *False*. No counterexample is need.

   (a) Given $\{a_n\}_{n=1}^{\infty}$, if a_n converges, then it it is bounded. Likewise, if a_n is bounded, then it is convergent. [1]

   (b) If $\{a_n\}$ and $\{a_n\}$ are both divergent, then $\{a_n + b_n\}$ is divergent. [1]

   (c) $\int_0^{\pi} \sec\theta \, d\theta$ is an improper integral. [1]

[1]Reasoning: this ensures all trigonometric functions will always be positive.

12.4 Cross Products

Calculus III

0.1 Cross Product

Given two nonzero vectors and in $\mathbb{R}^3$, the **cross product** $\times$ is a vector with magnitude

$$| \times | = |||| \sin\theta \tag{1}$$

where $0 \leq \theta \leq \pi$ is the angle between and . The direction of $\times$ 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 to the direction of $\times$ is the direction of your thumb, orthogonal to both and . When $\times = 0$, the direction of $\times$ undefined.

0.1.1 Geometry of the Cross Product

Let and be two nonzero vectors in $\mathbb{R}^3$.

1. The vectors and are parallel ($\theta = 0$ or $\theta = \pi$) if and only if $\times = 0$.
2. If and are two sides of a parallelogram, then the area of the parallelogram is

$$| \times | = |||| \sin\theta \tag{2}$$

0.1.2 Properties of the Cross Product

Let , , and be nonzero vectors in $\mathbb{R}^3$, and let a and b be scalars.

$$\times = -(\times) \qquad \text{Anticommutative property} \tag{3}$$
$$(a) \times (b) = ab(\times) \qquad \text{Associative property} \tag{4}$$
$$\times(+) = (\times) + (\times) \qquad \text{Distributive property} \tag{5}$$
$$(+)\times = (\times) + (\times) \qquad \text{Distributive property} \tag{6}$$

0.1.3 Cross Products of Coordinate Unit Vectors

60125 [tdplot$_m ain_c oords, cube/.style = verythick, black, grid/.style = verythin, gray, axis/.style = -{>}, blue, thick$]
in -1.5,0,...,3.5 in -1.5,0,...,3.5 [grid] (,-1.5) – (,3.5); [grid] (-1.5,) – (3.5,);
[axis] (0,0,0) – (5,0,0) node[anchor=west]x =; [axis] (0,0,0) – (0,5,0) node[anchor=west]y =; [axis] (0,0,0) – (0,0,5) node[anchor=west]z =;

$$\times = -(\times) = \tag{7}$$
$$\times = -(\times) = \tag{8}$$
$$\times = -(\times) = \tag{9}$$
$$\times = \times = \times = 0 \tag{10}$$

0.1.4 Evaluating the Cross Product

Let $= u_1 + u_2 + u_3$ and $= v_1 + v_2 + v_3$. Then

$$\times = \begin{vmatrix} & & \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{vmatrix} = \begin{vmatrix} u_2 & u_3 \\ v_2 & v_3 \end{vmatrix} - \begin{vmatrix} u_1 & u_3 \\ v_1 & v_3 \end{vmatrix} + \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix} \tag{11}$$

15.5 Divergence and Curl

Calculus III

0.1 Divergence and Curl

0.1.1 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

$$\operatorname{div}\mathbf{F} = \nabla\cdot\mathbf{F} = \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} + \frac{\partial h}{\partial z} \tag{1}$$

If $\nabla\cdot\mathbf{F} = 0$, the vector field is **source free.**

0.1.2 Divergence of Radial Vector Fields

For a real number p, the divergence of the radial vector field

$$\mathbf{F} = \frac{\mathbf{r}}{|r|^p} = \frac{\langle x,\, y,\, z\rangle}{(x^2+y^2+z^2)^{\frac{p}{2}}} \text{ is } \nabla\cdot\mathbf{F} = \frac{3-p}{|\mathbf{r}|^p} \tag{2}$$

0.1.3 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

$$\nabla\times\mathbf{F} = \operatorname{curl}\mathbf{F} = \left(\frac{\partial h}{\partial y} - \frac{\partial g}{\partial z}\right) + \left(\frac{\partial f}{\partial z} - \frac{\partial h}{\partial x}\right) + \left(\frac{\partial g}{\partial x} - \frac{\partial f}{\partial y}\right) \tag{3}$$

If $\nabla\times\mathbf{F} = \mathbf{0}$, the vector field is **irrotational**.

0.1.4 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 ration 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

$$\omega = |\mathbf{a}| = \frac{1}{2}|\nabla\times\mathbf{F}| \tag{4}$$

0.1.5 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.

0.1.6 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.

0.1.7 Product Rule for the Divergence

Let u be a scalar-valued functino that is differentiable on a region D and let $\mathbf{F}$ be a vector field that is differentiable on D. Then

$$\nabla \cdot (u\mathbf{F}) = \nabla u \cdot \mathbf{F} + u(\nabla \cdot \mathbf{F}) \tag{5}$$

0.1.8 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.

Confidence Intervals for a Population Mean

Let $X_1, \ldots, X_n$ 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**,

$$\bar{X} \sim N\left(\mu, \frac{\sigma^2}{n}\right)$$

and

$$S_n \sim N\left(n\mu, n\sigma^2\right)$$

Let $X_1, \ldots, X_n$ 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

$$\bar{X} \pm z_{\frac{\alpha}{2}}\sigma_{\bar{X}}$$

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 $X_1, \ldots, X_n$ be a small $(n < 30)$ sample from a *normal* population with mean μ. Then the quantity

$$\frac{\bar{X} - \mu}{\frac{s}{\sqrt{n}}}$$

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 $X_1, \ldots, X_n$ 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:

$$z = \frac{\bar{X} - \mu_0}{\frac{\sigma}{\sqrt{n}}}$$

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 $X_1, \ldots, X_n$ 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

$$t^* = \frac{\bar{X} - \mu_0}{\frac{s}{\sqrt{n}}}$$

- 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 $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_n$ be **large** $(n_x > 30$ and $n_y > 30)$ independent random samples from populations with mean u_x and u_y and standard deviation σ_x and σ_y, respectively.
The test statistic is as follows:

$$z^* = \frac{(\bar{X} - \bar{Y}) - \delta_0}{\sqrt{\frac{\sigma_X^2}{n_x} + \frac{\sigma_Y^2}{n_y}}}$$

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**
$H_0 : \mu_x - \mu_y \leq \delta_0$	$H_1 : \mu_x - \mu_y > \delta_0$	$P(Z \geq z^*)$
$H_0 : \mu_x - \mu_y \geq \delta_0$	$H_1 : \mu_x - \mu_y < \delta_0$	$P(Z \leq z^*)$
$H_0 : \mu_x - \mu_y = \delta_0$	$H_1 : \mu_x - \mu_y \neq \delta_0$	$2 \times P(Z \geq \lvert z^* \rvert)$

Small-Sample Tests for the Difference Between Two Means

Population Variances Are Not Equal

Let $X_1, \ldots, X_{n,x}$ and $Y_1, \ldots, Y_{n,y}$ 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

$$\nu = \frac{\left[\frac{s_X^2}{n_X} + \frac{s_Y^2}{n_Y}\right]^2}{\frac{\left(\frac{s_X^2}{n_X}\right)^2}{n_X - 1} + \frac{\left(\frac{s_Y^2}{n_Y}\right)^2}{n_Y - 1}}$$

- Compute the test statistic

$$t = \frac{(\bar{X} - \bar{Y}) - \Delta_0}{\sqrt{S_X^2/n_x + S_Y^2/n_Y}}$$

- Compute the P-value. The P-value is an area under the Student's t curve with v 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 $X_1, \ldots, X_{n,x}$ and $Y_1, \ldots, Y_{n,y}$ 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 qual, then, to test a null hypothesis of the fortm $H_0 : \mu_X - \mu_Y \leq \Delta_0$, $H_0 : \mu_X - \mu_Y \geq \Delta_0$, or $H_0 : \mu_x - \mu_y = \Delta_0$:

- Compute

$$s_p = \sqrt{\frac{(n_X - 1)s_X^2 + (n_Y - 1)s_Y^2}{n_X + n_Y - 2}}$$

- Compute the test statistic

$$t = \frac{(\bar{X} - \bar{Y}) - \Delta_0}{s_p\sqrt{\frac{1}{n_X} + \frac{1}{n_Y}}}$$

- 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) deasures the strength and direction of a linear relationship between two variables. Let $(x_1, y_1), \ldots, (x_n, y_n)$ represent bivariate data, then the correlation coefficient is

$$\begin{aligned} r &= \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{s_x}\right)\left(\frac{x_i - \bar{x}}{s_x}\right) \\ &= \frac{\sum_{i=1}^n x_i y_i - n\bar{x}\bar{y}}{\sqrt{\sum_{i=1}^n x_i^2 - n\bar{x}^2}\sqrt{\sum_{i=1}^n y_i^2 - n\bar{y}^2}} \qquad = \frac{SSR}{SST} \end{aligned}$$

The Least-Squares Line

For an equation of the form

$$y_1 = \beta_0 + \beta_1 x_i + \epsilon_i$$

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**.

$$\begin{aligned} e_1 &= y_i - \widehat{y}_i = y_i - \widehat{\beta}_0 - \widehat{B}_1 x_i \\ \widehat{\beta}_1 &= \widehat{\beta}_1 = \frac{\sum_i^n y_i - n\bar{x}\bar{y}}{\sum_i^n x_i^2 - n\bar{x}^2} \\ \widehat{\beta}_0 &= \bar{y} - \widehat{\beta}_1\bar{x} \end{aligned}$$

Uncertainties in the Least-Squares Coefficients

Using some assumptions,

- The quantity $\widehat{\beta}$ is *normally distributed* random variables.
- The means of $\widehat{\beta}$ is the true values of $\widehat{\beta}$.
- The *standard deviations* of β is estimated with

$$\begin{aligned} s_{\beta_0} &= s\sqrt{\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^n (x_i - \bar{x})^2}} \\ s_{\beta_1} &= \frac{s}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2}} \end{aligned}$$

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 $\frac{\widehat{\beta_1}-\beta_1}{s_{\beta_1}}$ and $\frac{\widehat{\beta_1}-\beta_1}{s_{\beta_1}}$ 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

$$\bar{\beta_0} \pm t_{n-2} \times s_{\bar{\beta_0}} \qquad \bar{\beta_1} \pm t_{n-2} \times s_{\bar{\beta_1}}$$

Level $100(1-\alpha)\%$ confidence intervals for the quantity $\beta_0 + \beta_1 x$ is given by

$$\widehat{\beta_0} + \widehat{\beta_1} x \pm t_{n-2,\alpha/2} \times s_{\bar{y}}$$

where

$$s_{\bar{y}} = s\sqrt{\frac{1}{n} + \frac{(x-\bar{x})^2}{\sum_{i=1}^{n}(x_i-\bar{x})^2}}$$

Checking Assumptions

If the plot of residuals versus fitted values

- Shjows no substantial trend or curve, and
- Is **homoscedastic**, that is, the vertical spread does not varu 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 standad 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 correleation (ρ).
- Some properties of the correlation coeffciient (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 whicht he 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 correleation 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 = $\sum_{i=1}^{n}(y_i-\widehat{y})^2$
- SSR = regression sum of squares = $SST - SSE$
- The following assumptions are satisfied.
  - The errors $\epsilon_1, \ldots, \epsilon_0$ 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 $\epsilon_1, \ldots, \epsilon_0$ all have mean 0.
  - The errors $\epsilon_1, \ldots, \epsilon_0$ all have the same variance, which we denote by σ^2.
  - The errors $\epsilon_1, \ldots, \epsilon_0$ 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**
Pan Size	153.94 cm^2	254.47 cm^2	314.16 cm^2
Salt Weight	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.

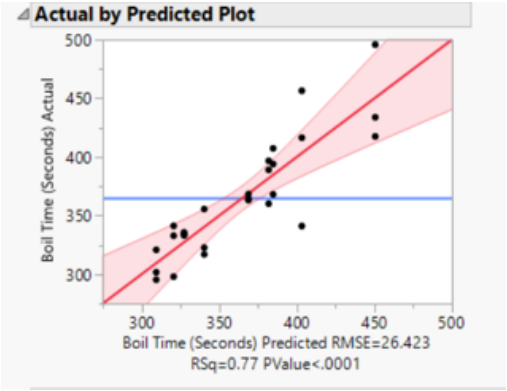
To analyze the data we first needed to test for the presence of interaction effect.

$$H_0 : \gamma_{11} = \gamma_{12} = \ldots = \gamma_{IJ} = 0$$
$$H_1 : \text{at least one of the } \gamma_{ij} \text{ is nonzero}$$

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.



(a) The Actual by Predicted Plot

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

(b) The Main Effects Table

Figure 3: The Results

$$H_0 : \alpha_1 = \alpha_2 = \ldots = \alpha_I = 0$$
$$H_1 : \text{at least one of the } \alpha_i \text{ is nonzero}$$

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.

$$H_0 : \beta_1 = \beta_2 = \ldots = \beta_J = 0$$
$$H_1 : \text{at least one of the } \beta_j \text{ is nonzero}$$

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 than 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.

PHYS

Physics

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, dA 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 $\vec{\mathbf{d}}$ 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.

$$\phi_E = \oint \vec{E} \cdot d\vec{A} \tag{1}$$

$$= \oint E \; da \cos\theta \qquad \text{Cross product expansions} \tag{2}$$

$$= E \oint dA \qquad \text{If E is constant, pull it out} \tag{3}$$

$$= E \times \text{Surface Area} \tag{4}$$

$$\tag{5}$$

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

$$t = \gamma t_0 \qquad L = \frac{L_0}{\gamma}$$

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

$$E = \gamma m_0 c^2 \qquad \vec{p} = \gamma m_0 \vec{v}$$

From the this, we can derive the more fundamental equation:

$$E^2 = m_0^2 c^4 + p^2 c^2 \tag{1}$$

For kinetic energy, we can eloquently describe it as $(\gamma - 1)m_0 c^2$, this is simply the rest mass energy (mc^2) subtracted from the total energy (γmc^2). At low speeds (i.e. $v \ll c$), we can simply use a Taylor Series expansion to get $E \approx {}^1\!/_2 m_0 v^2 + \frac{3m_0 v^4}{8c^2} + \cdots$. Ignoring the other terms, we get $E \approx {}^1\!/_2 m v^2$, 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 is 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

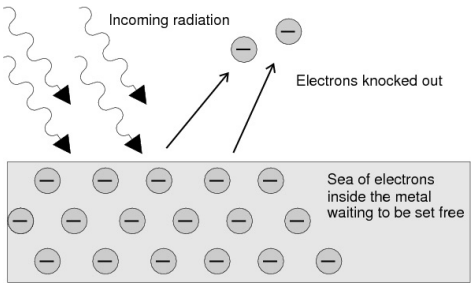


Figure 1 – The Photoelectric Effect.

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 $\lambda_2 - \lambda_1 = \frac{h}{m_0 c}(1 - \cos\theta)$, 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.

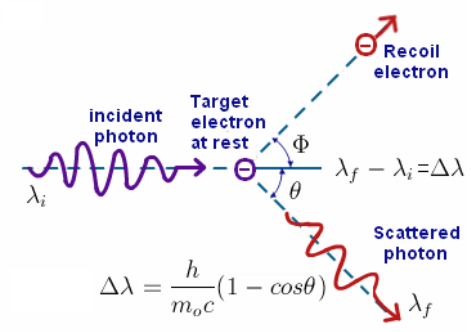


Figure 2 – Compton scattering.

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

$$|\psi(x)|^2 dx = \text{the probability of finding the particle in the range of } x \text{ to } x + dx$$

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,

$$\Delta t \Delta E \geq \frac{\hbar}{2} \qquad \hbar = \frac{h}{2\pi}$$

Or, more famously,

$$\Delta x \Delta p \geq \frac{\hbar}{2} \qquad \hbar = \frac{h}{2\pi}$$

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,

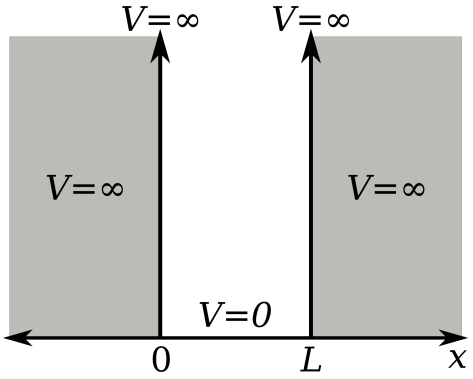


Figure 3 – Particle in a box (or infinite potential well).

$$E = \frac{p^2}{2m} = \frac{n^2h^2}{8mL^2}$$

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 $n = 1, 2, 3, \ldots, \infty$, where ∞ is ionization).

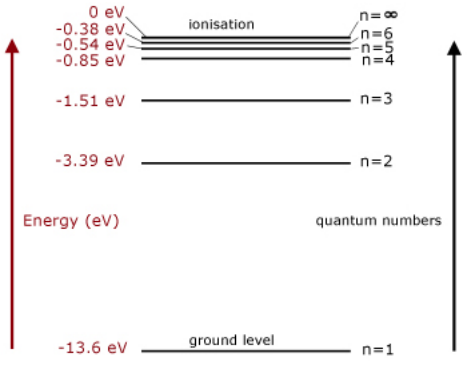


Figure 4 – Energy levels of hydrogen atom.

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,

$$\begin{aligned} h\nu = E_f - E_i &= \frac{E_1}{n_f^2} - \frac{E_1}{n_i^2} \\ &= E\left(\tfrac{1}{n_f^2} - \tfrac{1}{n_i^2}\right) \end{aligned}$$

Similarly, for emission,

$$h\nu = E_i - E_f = \frac{E_1}{n_i^2} - \frac{E_1}{n_f^2}$$
$$= E\left(\frac{1}{n_i^2} - \frac{1}{n_f^2}\right)$$

PHIL

Philosophy

PHILOS 3235: Test 2

Due on Wednesday, May 10th, 2017

Dittmer 15:00

Illya Starikov

Monday, April 3rd, 2017

Problem I

Utilitarianism states that actions are right just in case they produce the most overall (available) goodness. Discuss some ways in which according to Kantian ethics, utilitarianism is false.

Utilitarianism and Kantian ethics seem to have have a large overlap. After all, utilitarianism is the doctrine that actions are right if they benefit the majority and Kantian Ethics reflect much of this thinking. However, when analyzing the two frameworks of ethics, there are areas where they differ. The biggest area of difference is in the first Categorical Imperative.

The first Categorical Imperative states that

> Act only according to that maxim by which you can at the same time will that it should become a universal law.

This may seem like a perfect utilitarian definition; until one realizes that "universal" is not the same as "maximum benefit". To show how this juxtaposition is true, let us take two examples.

One of the most widely known examples of a wrong in Kantian Ethics is suicide. If everyone committed suicide (making it universal), the human-centric ethics would fall apart. However, we *know* there to be situation where self-sacrifice is the correct in a utilitarian framework. Consider the transplant problem.

> Suppose you are an outstanding surgeon working in the top medical facility in your area. During a shift, you have five subsequent patients needing transplants within the hour (with their life in jeopardy). In a stroke of luck, a healthy patient walks is in for a checkup that has all the organs necessary. Do you save five at the expense of one?

The one patient is **obligated** to sacrifice himself to save five other patients. This is not the case in a Kantian framework.

Take another example: a situation where you are forced to lie. Imagining back to Nazi Germany in WWII, no sane human being would admit to harboring a fugitive Jew. In a utilitarian framework, not getting the innocent person killed would definitely be considered the maximum benefit. However, in pure Kantian Ethics, lying is very much breaking the first Categorical Imperative.

Admittedly, there is a lot of overlap in Utilitarian and Kantian; most of which is at the intersection of maximum benefit and universal benefit. However, from the point of maximum (whether it be at the 51% mark or on) to the point of universal (which has to lie at the 100% mark), there is a tangible difference.

Problem II

> Utilitarianism states that actions are right just in case they produce the most overall (available) goodness. Discuss some ways in which according to Kantian ethics, utilitarianism is false.

Upon making decisions, the goal should always be to set create a policy that everyone can agree to. This goal cannot scale, of course, but it should give a general heuristic for how to gauge the conversation. For every situation, the can be a *hypothetical* common ground that everyone can agree to.

This may seem impossible, seeing how everyday situations do not reflect this thinking. However, when considering everyday situations, we are often blind sided by a different way of thinking: the popular vote. When trying to agree on something, we fight for the maximum benefit instead of the most benefit for the group. To achieve a common policy, one must take a more utilitarian point of view.

Take a common, everyday example of a deciding where and what to eat. This monumental decision seemingly never is able to get settled. Instead of the group all blurting what they would like, set up a list of food, and have everyone vote on what they *would be willing* to eat. While the popular vote would leave people unhappy, this system allows for everyone to be satisfied.

Of course, scaling this is very difficult. Having a common ground for the entirety of the United States on foreign policy is borderline impossible because of sheer scale. But on a smaller scale, the framework is feasible.

The main limitation of this framework the "the weakest link" principle. Almost always, it always appears to be a Pareto principle, where 80% of the people can agree on most things while 20% always disagree. Being at the mercy of the 20% or less often makes it very difficult to come to a common policy.

In summary, it is technically feasible to come to a common policy. By taking a more utilitarian approach, and not being selfish and trying to meet your own goals, a common ground can be established quite easily. However, the difficulty lies in the small subset that prohibit progress by disagreeing with the established policy.

Problem III

> Visit `moralmachine.mit.edu`. Take "the test". Characterize two of the (thirteen) scenarios presented to you, and what you chose in those scenarios. Then explain why you made those choices.

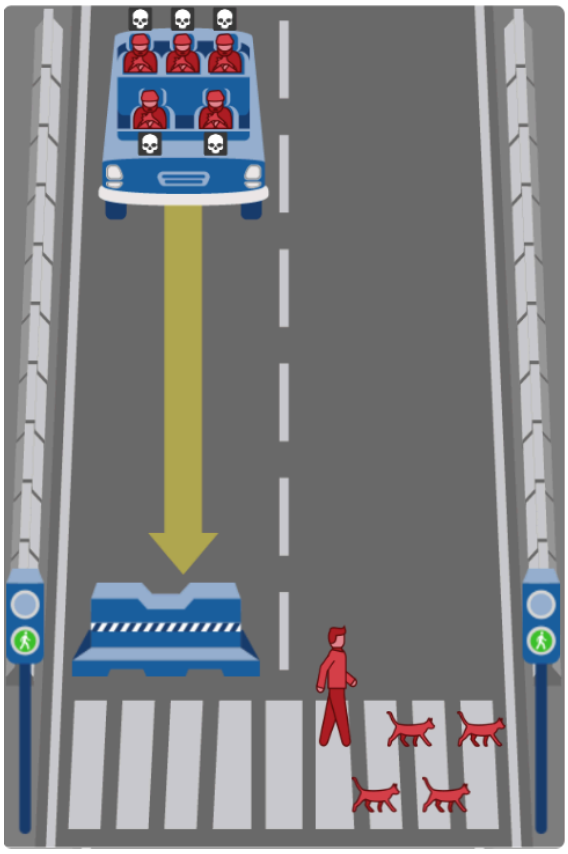
When considering the moral issues of the self driving car, it should always be made clear: *The car has no "mind" of its own.* Many people fall into the trap of empathizing with the self driving car, not realizing it's a two ton metal machine with algorithmic decision making. When considering the trolley-esque problem of selfing driving cars deciding what/who to impact, it is not the car deciding — the programmer is deciding. So it always better to empathize with the programmer, instead of the car. Unfortunately, this is not always the case. For the arguments presented here, the approach will be completely from the perspective of the programmer.

For the first scenario, described in Figure **??**, I believe the scenario that should be chosen is the straight into the concrete barrier. This may seem counterintuitive, favoring five human lives over 1 human life and 4 dead animals. However, remember the empathy is with the programmer. The best heuristic is often the simplest one; in this case, the simpler heuristic is going straight into a barricade.

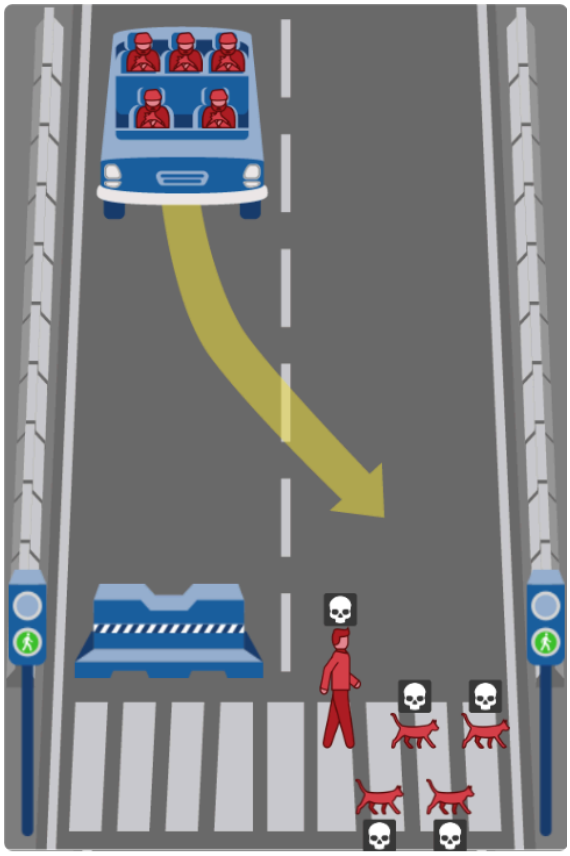
To further expand, this heuristic *should not* always be used. This heuristic should be used in situations where there are a *comparable* number of passengers and pedestrians.

For the second scenario, described in Figure **??**, I believe the scenario that should be chosen is to swerve — contradicting my first decision! This is simply because this is no longer a pedestrian vs. passenger situation. It is a pedestrian vs. pedestrian situation. This alternative situation calls for a different heuristic, because it a completely different situation.

From the perspective of the programmer, both of these situations are a good trade off between simplicity and favorable for the passengers/pedestrians — and this is something the consumer should want; even though it is not always what the consumer *does* want.

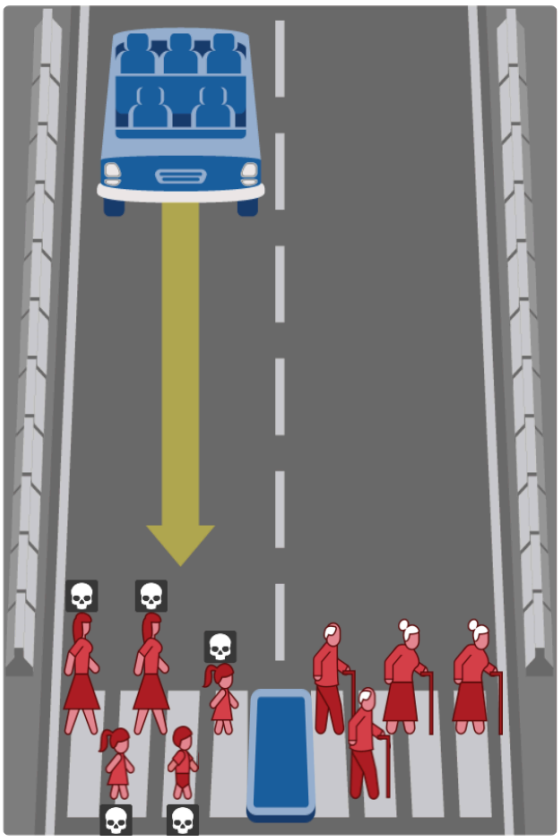


(a) In this case, the self-driving car with sudden brake failure will continue ahead and crash into a concrete barrier. *This will result in 5 dead homeless people.*

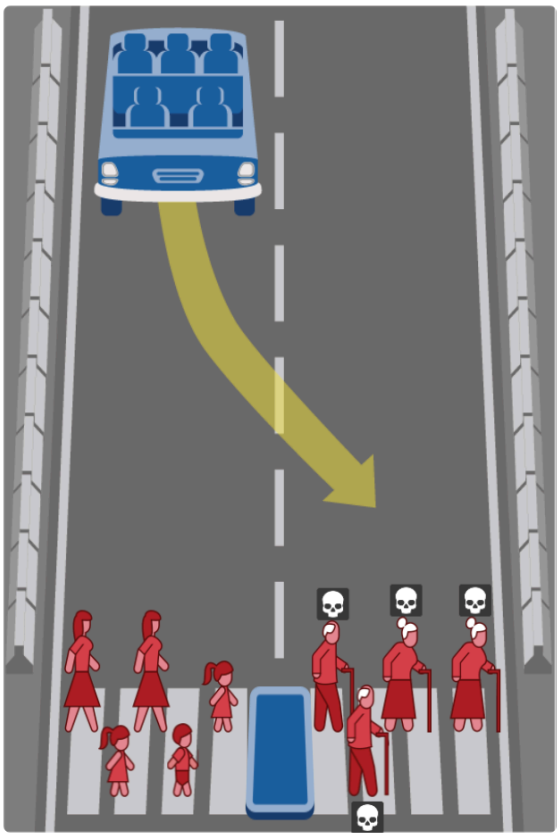


(b) In this case, the self-driving car with sudden brake failure will swerve and drive through a pedestrian crossing in the other lane. *This will result in 1 dead man, 4 dead cats.*

Figure 1: First scenario.



(a) In this case, the self-driving car with sudden brake failure will continue ahead and drive through a pedestrian crossing ahead. *This will result in 2 women, 2 girls, 1 boy dead.*



(b) In this case, the self-driving car with sudden brake failure will swerve and drive through a pedestrian crossing in the other lane. *This will result in 2 elderly men, 2 elderly women.*

Figure 2: Second scenarios