# Algorithms Fall 2019

Eric A. Autry

August 27, 2019

# Course People

Professor

- Eric Autry
- eric.autry@duke.edu
- Office: Physics 017 (Physics Building Basement)

TAs

- Jiaran Zhou
- Prathikshaa Rangarajan
- Yukun Yang
- Fengyi Li
- Nan Ni
- Jingru Gao
- Hanyu Xie
- Kai Zhang

# Office Hours and Study Hall

Office Hours:

- Still finalizing schedule and room reservations.
- Will be posted on course website and sent out in an announcement.

Study Hall: Wednesday 4 - 8 pm in TBD

- I will be present to answer questions, along with 2-3 TAs for the entire time.

Homework and projects will be due in class on Thursdays (typically).

# Website

- We will be using the course's Sakai site. Make sure you have access to the site.
- Major announcements will be sent through Sakai. A test announcement was sent this morning before class. Make sure you received it.
- Course notes will be posted on the website in the form of annotated lecture slides within a day or two of the lecture.
- All homework and projects will be posted on the Sakai site.
- All grades will be posted on Sakai.
- We will be using the course Piazza available through Sakai.

# Textbooks

There will be NO required textbooks for this course.

If you are interested, the textbooks I am using:

- Michael Sipser, Introduction to the Theory of Computation
  - Third Edition
  - ISBN-13: 978-1133187790, ISBN-10: 113318779X
- Sanjoy Dasgupta, Algorithms
  - ISBN-13: 978-0073523408, ISBN-10: 0073523402

# Course Grading

Homework: 30%

- Pen-and-paper assignments. Probably 8 of them.

Projects: 30%

- Programming assignments. Probably 4 of them.

Midterm: 15%

- Date: September 26th (or October 1st) in class

Final: 25%

- Date:
  - 11:45 am Section: December 16th, 7 pm - 10 pm
  - 4:40 pm Section: December 11th, 2 pm - 5 pm

# Style Points

Up to 10% of both written homework and coding projects will be ‘style’ points.

Basic idea: homework should be legible and organized, arguments should be complete but concise, and code should be well commented and elegant.

More thorough guidelines will be discussed when the first homework and first project are assigned.

# Late Policy

Each student will receive 3 ‘late tokens’ that can each be used for a 24-hour extension on any homework assignment or project.

Otherwise: -10% per day late

# Collaboration Policy

- All students are expected to follow the Duke Community Standard found at: integrity.duke.edu/standard.html
- Quiz 0 posted on Sakai asks that all students to read this Standard and provide an electronic signature to indicate that they will adhere to this Standard throughout their work in this course.
- Collaboration is allowed and encouraged for both homework and projects.
- Collaboration is not copying.
  - It is not acceptable for one student to simply give another student a solution.
  - Each student is expected to submit their own solution in their own words.

# Collaboration Policy

For projects, pair programming is encouraged with the following guidelines:

- No more than two students can work together.
- Each pair will submit a single version of their solution, indicating the names of the two partners clearly in the submission.
- Each of the partners must be present and working at the same computer any time work is being done on their project.
- 50-50 rule: each of the partners should spend 50% of the time ‘driving’ and 50% of the time ‘navigating.’

# Emergency Classroom Procedures

Guidelines for what to do in an emergency can be found at:

emergency.duke.edu/what-to-do

In most cases you will receive a DukeALERT notification by email and text if there is an emergency situation on campus, and you may also hear the outdoor sirens.

- Tornado
- Fire
- Active Shooter

# Disability Accommodations

The Student Disability Access Office (SDAO) and the Academic Resource Center (ARC) grant ‘accommodations’ or ‘interventions’ to students with documented disabilities.

Please email me at eric.autry@duke.edu if you are one of these students and I will work with you to find proper accommodations.

# Course Overview

So... What is this course about?

The focus of this course is on problem solving:

- what problems we might consider
  - those we can solve VS those that cannot be solved
- what techniques we might use
- what machines might help us

# Question 1

The first major question we will consider:

What is a computation machine, and what sort of problems can it solve?

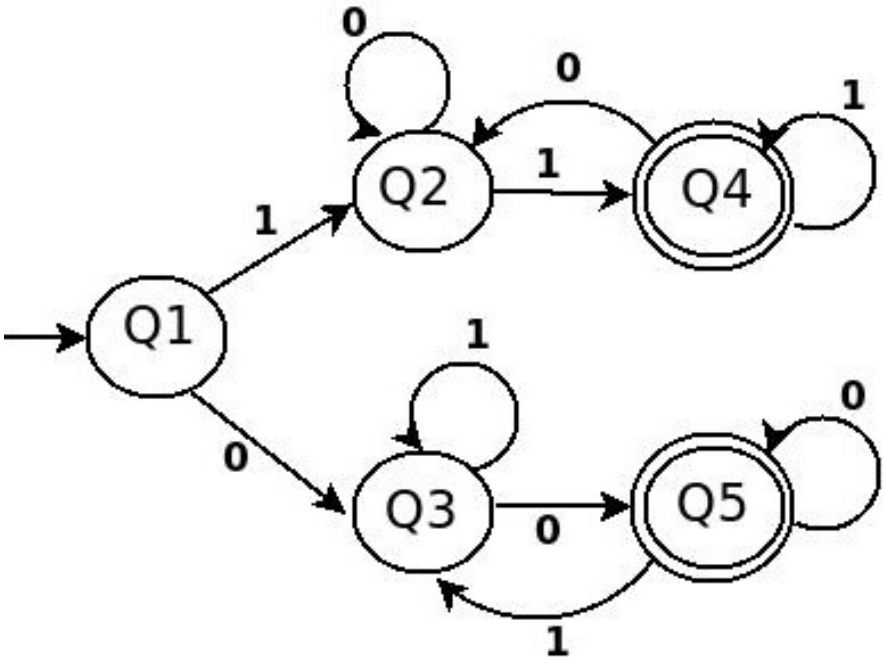
# Unsolvable Problems

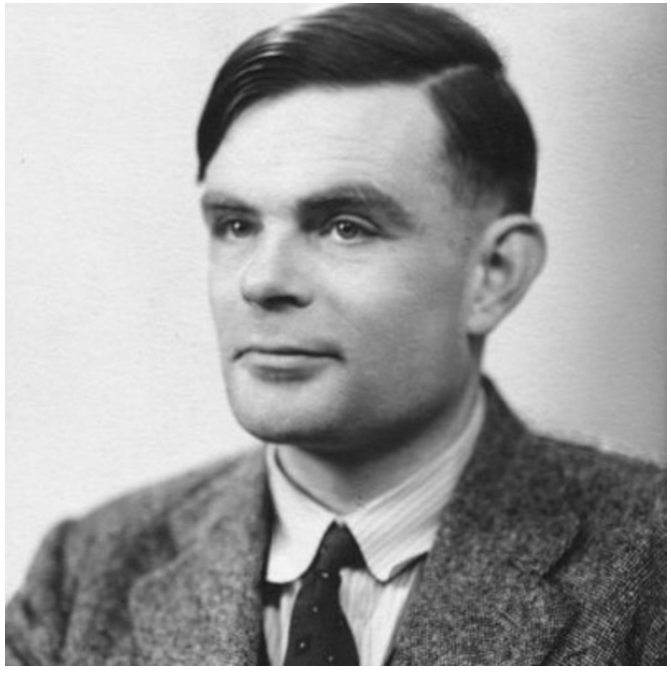
There do exist problems that are provably unsolvable.

For example, these basic questions about the performance of provided code:

- Are two pieces of code equivalent to each other?
- Will this code enter an infinite loop?
- Will two different variables in this code ever have the same value?

# Automata Theory



# Automata Theory

We will build a theoretical framework of computation and create a mathematical description of the machines used.

This framework will allow us to classify problems based on their 'difficulty'.

- computational trade-off: power vs analyzability
- goal: use the least powerful machine that gets the job done

Our discussion of Automata Theory will culminate with a description of the powerhouse of computing: Turing Machines.

# Question 2

Now that we have a sense of what problem can actually be solved, our next natural question is:

How?

Or, more specifically:

How can we solve problems efficiently and optimally?

# Basic Algorithmic Tactics

We will discuss a number of basic approaches that many algorithms rely upon.

These are the ‘building blocks’ of more complicated algorithms we will see:

- Greed
- Recursion
- Divide-and-Conquer
- Use it or Lose it
- Dynamic Programming

# Complexity

Once we have a sense for what methods we can use, we can start to discuss their complexity.

Algorithmic complexity is a way to represent how efficient an algorithm is.

Usually presented: the number of actions performed as a function of the size of the input (for large inputs). This is called the asymptotic runtime of the algorithm.

# Derivation of Well Known Algorithms

At this point in the course, we will have a sense of the algorithmic tactics we can use, and an understanding of the methods to measure an algorithm's complexity.

This knowledge will allow us to begin examining some of the classical problems, and to derive the famous algorithms used to solve them:

- Sorting a List
- Shortest Path in a Graph (aka 'Network')
- Minimum Spanning Trees
- Network Flow
- and more...

# Question 3

We have now begun to answer:

- What problems are solvable.
- How problems can be solved efficiently.

It now becomes natural to ask:

Can all problems be solved efficiently, and if not, is there a way to efficiently approximate the solutions?

# Question 3

Can all problems be solved efficiently?

$1,000,000 Question:

Does $P = NP$?

# NP Complete Problems

The last major topic we will examine in this course is:

NP Completeness

We will see a number of NP Complete Problems and how they relate to each other:

- 3SAT
- Vertex Cover
- Linear Programming
- Hamiltonian Paths and Cycles
- Traveling Salesman
- and many others...

# NP Problems and Turing Machines

As it turns out, NP stands for nondeterministic polynomial time problems.

These are the set of all problems that can be solved in polynomial time (i.e. 'can be solved efficiently') by a nondeterministic Turing Machine.

It's a good thing we start the course with Turing Machines!