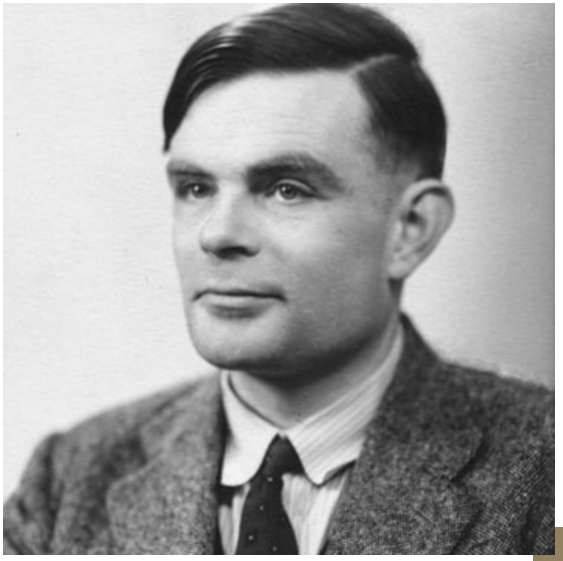
Data Structures and Algorithms 2

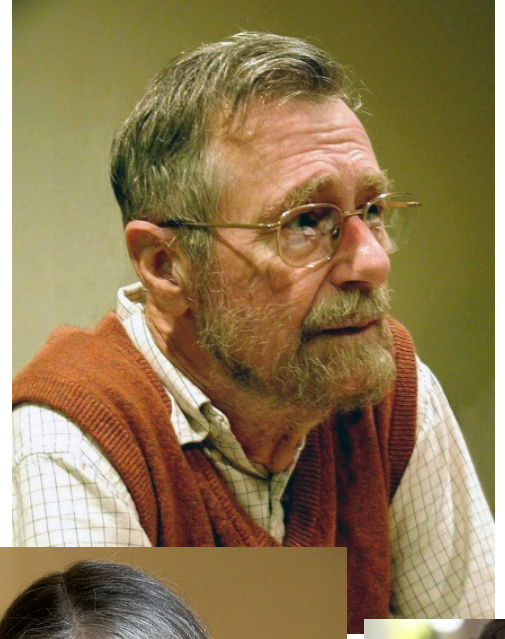
Lecture 1: Introduction and Logistics

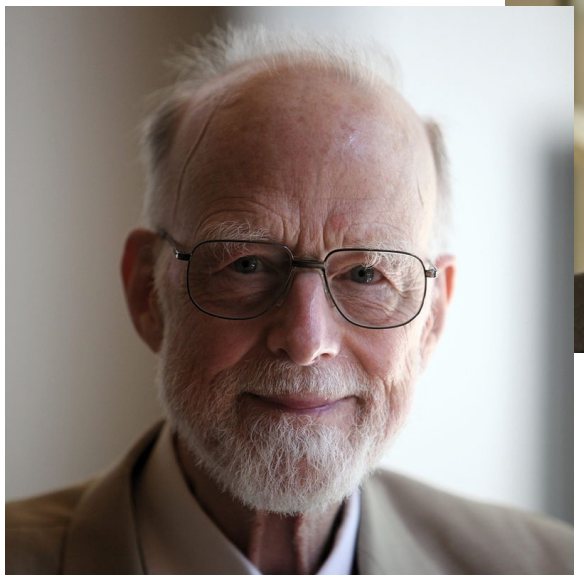
Co-instructors: Robbie Hott and Ray Pettit

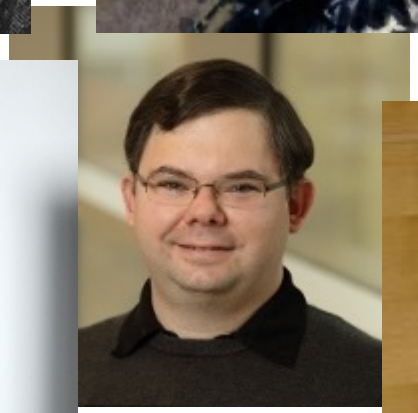
Spring 2024

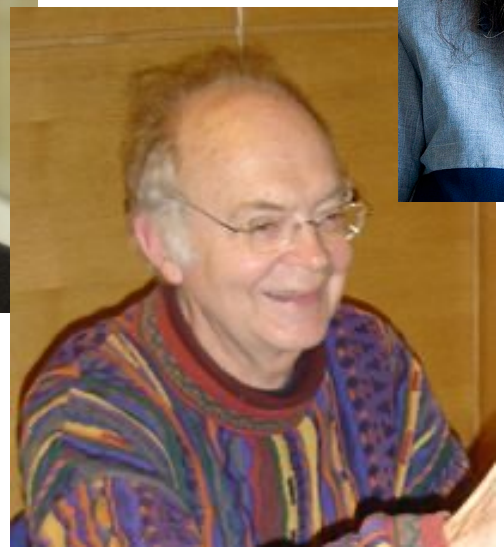
Who's Who in Algorithms?

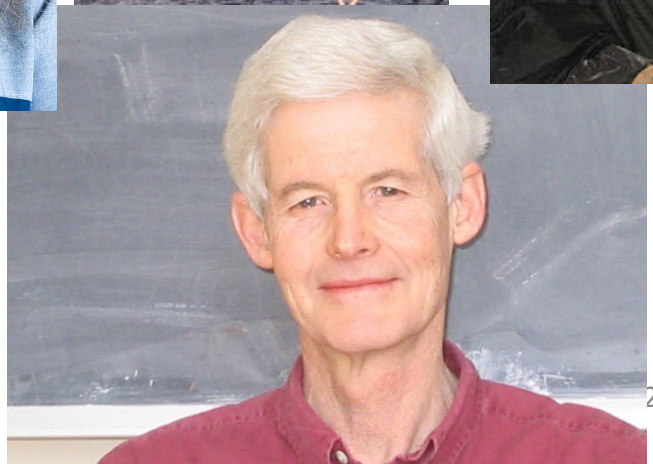




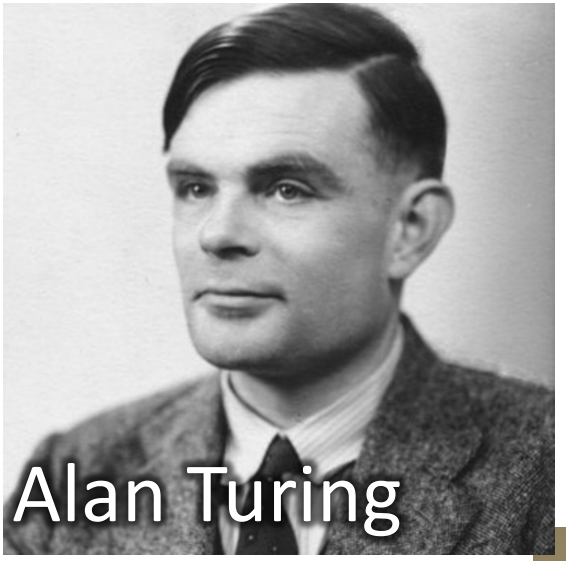




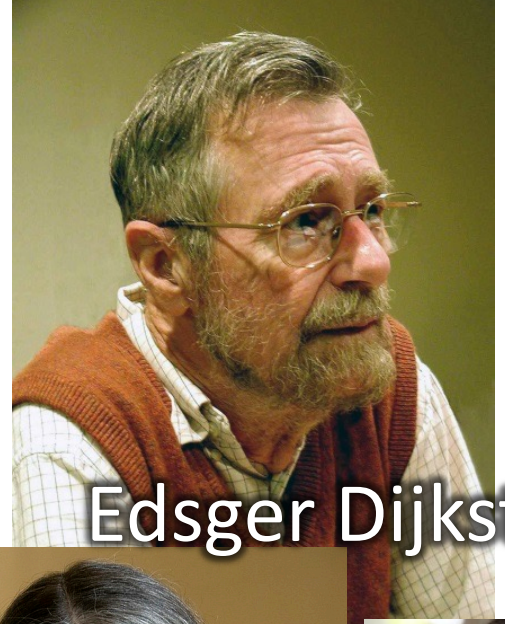




Who's Who in Algorithms?


Alan Turing


Ada Lovelace

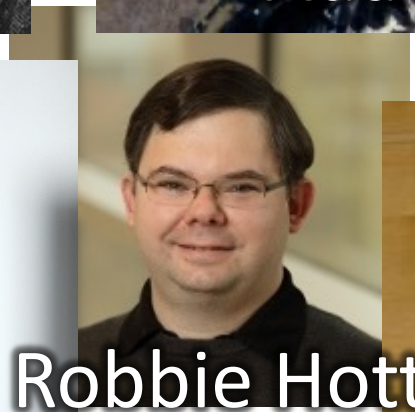

Edsger Dijkstra

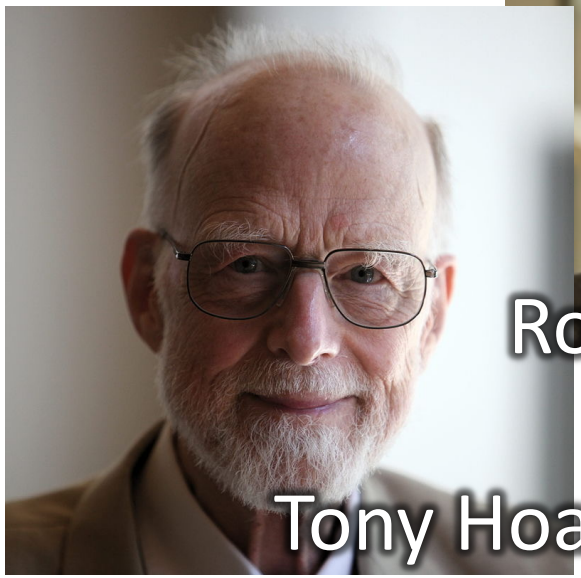

Al-Khwarizmi


Radia Perlman

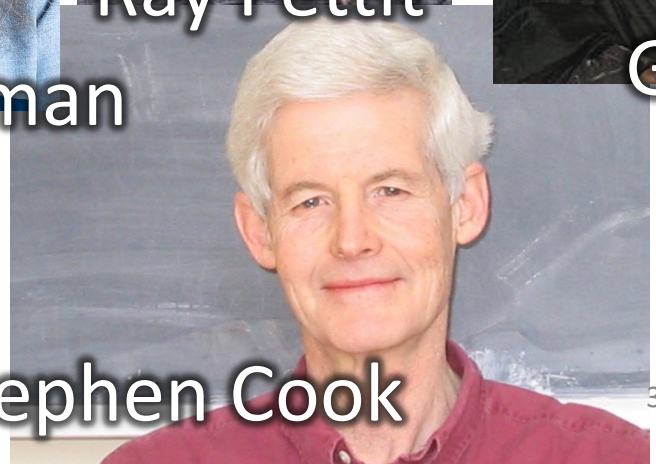

Ray Pettit


Gauss


Robbie Hott


Tony Hoare


Donald Knuth


Stephen Cook

Who's Who in Algorithms?

Euclid – father of geometry, logic, number theory
Al-Khwarizmi – algebra, solution to linear and quadratic equations
Gauss – fundamental theorem of algebra, numeric analysis
Ada Lovelace – first computer program, more than number crunching
Alan Turing – breaking Engima, Turing Machine (abstract model of computation)
Edsger Dijkstra – structured programming languages, graph algorithms
Tony Hoare – Algol, Hoare logic, null reference, quicksort
Stephen Cook – polynomial time reduction, NP-completeness, proof complexity
Donald Knuth – analysis of algorithms, TAOCP, TeX, literate programming
Radia Perlman – spanning tree algorithm and protocol (STP), LOGO-Tortis language

A Historic Perspective

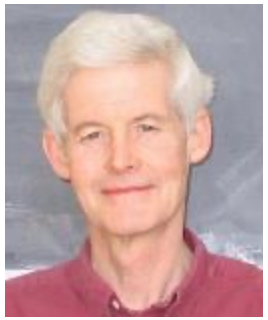
Euclid

Al-Khwarizmi

Gauss

Ada Lovelace

Stephen Cook



Radia Perlman

300 BC

800

1600

1800

1900

Alan Turing

Edsger Dijkstra

Don Knuth

What Is an Algorithm?

- In mathematics and computer science, an algorithm is a finite sequence of well-defined, computer-implementable instructions, typically to solve a class of problems or to perform a computation. Algorithms are unambiguous specifications for performing calculation, data processing, automated reasoning, and other tasks. [Wikipedia Jan 2020]
- An algorithm is a step by step procedure to solve logical and mathematical problems. [Simple English Wikipedia Aug 2019]

Being unambiguous is not always easy!

An example

Goals

Create an awesome learning experience

Instill enthusiasm for problem solving

Give broad perspective on computer science

Have fun!

Co-Instructors

Prof. Hott

Rice 210

jrhott@virginia.edu

Email or Piazza
(no DMs in Discord)

Prof. Pettit

Rice 212

rp6zr@virginia.edu

Email or Piazza
(no DMs in Discord)

Office Hours TBD! Wait for announcement!

See course website for TA office hours

cs3100@cshelpdesk.atlassian.net

Overview of Platforms

Canvas

- Mostly for linking to other things

Course Webpage

- https://uva-cs.github.io/cs3100-s24
- Contains syllabus, coursework instructions, lecture slides, lecture recordings

Gradescope (linked through Canvas)

- For submitting all assignments, viewing grades, requesting regrades

Discord

- Link will be announced soon!
- For communication, including: announcements, (some) office hours, memes, collaboration, finding teams, etc.

Piazza (available through Canvas)

- For Q&A about content and assignments
- Instructors, TAs and other students read and answer, so ask here rather than email instructors

Requirements

- Discrete Math and Theory 1 (CS 2120 or CS 2102) with C- or higher
- Data Structures and Algorithms 1 (CS 2100) with C- or higher
- Logarithms, exponents, summations, derivatives, limits & series (Calc I)
- Tenacity
- Inquisitiveness
- Creativity

Note: CS2100 pre-req taken seriously. Don’t meet it? Need approval or you will be dropped (after add deadline). ☹

Warning

This can be a very challenging class

- Hard material that combines problem-solving, logic, math and programming
- “Holy grail” of computer science
- Useful in practice
- Job interviews

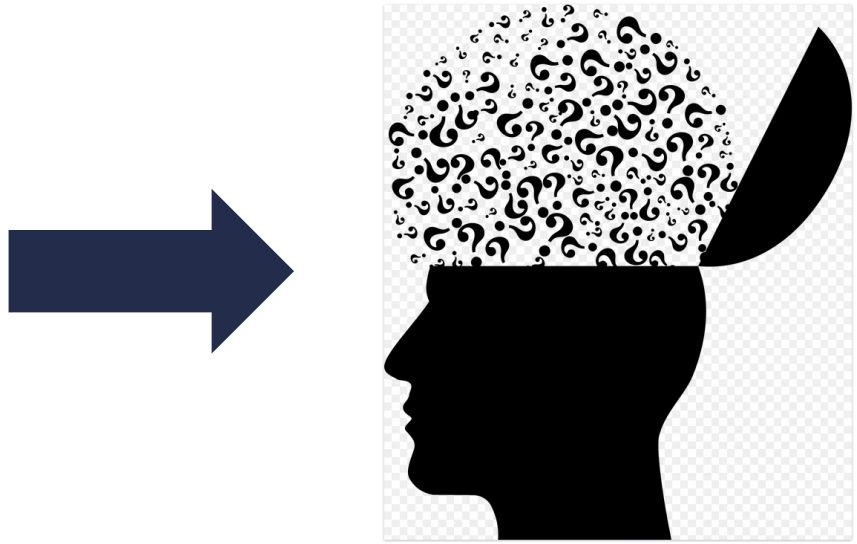
Lots of opportunities to succeed!

Hopefully not you…

“Learning Sources”

From what sources will you learn?

What I say in Lectures
What you get from the slides
Explanations you read in CLRS
Activities you do in/out of class
Assignments



- All of these are important.
- Realistically, IMHO it’s impossible to get all the “book knowledge” from lectures and slides!

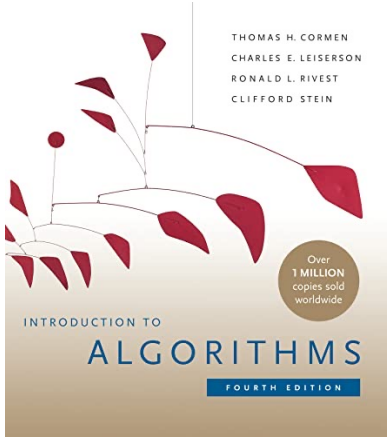
Textbook

You really need to read and study material other than the slides.

There are options, but a textbook is the easiest option.

I'll post readings from CLRS, urge you to read them.

Note: We will also have some resources posted on Collab site.



Freely accessible online via the UVA library

Follow this link!

Cormen et al. (CLRS) *Introduction to Algorithms* 4th Edition. (3rd edition OK, but…)

We may also ask you to read other online materials!

Units and Assignments

Five Units:

1. Graph Algorithms
2. Divide and Conquer
3. Greedy Algorithms
4. Dynamic Programming
5. Network Flow, Reductions, Machine Learning

Course work for each unit will include:

- Quiz (in-class, closed book)
- 1 Programming Assignment (PA)
- 2-3 Weekly Problem Sets (PS)

Additional coursework:

- “PS0”, a warm-up exercise on using LaTeX typesetting
- Several surveys, activities along the way

Course Learning Objectives

By the end of this course, students will:

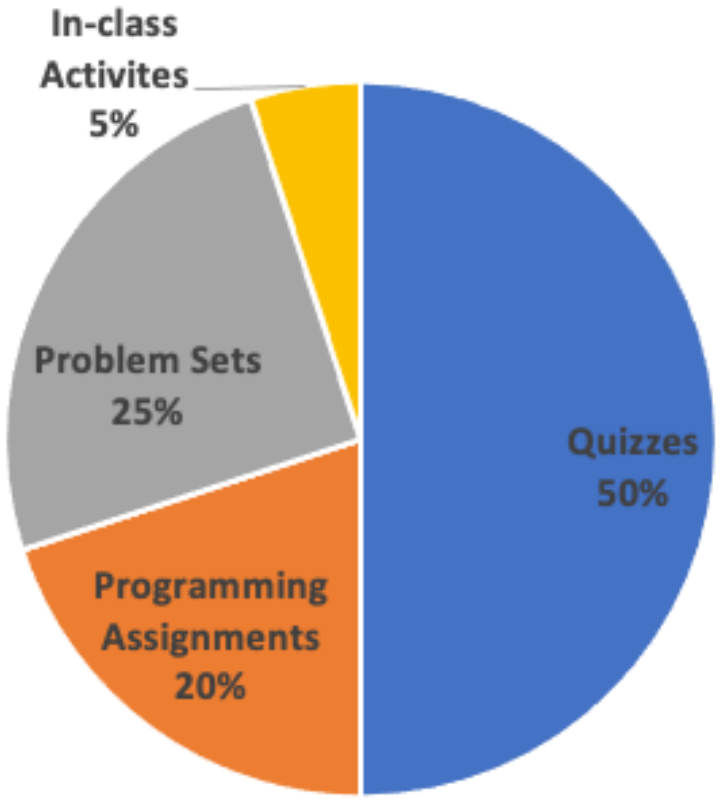
- **Understand and implement more advanced data structures,** including (but not limited to) graphs and sets (e.g., find-union). Be able to analyze the asymptotic complexity of the operations of these structures and use them to solve problems.
- **Understand a variety of problem solving paradigms such as dynamic programming, divide-and-conquer, and greedy algorithms.** Be able to implement a variety of algorithms, including Breadth-First (and Depth-First) Search, Dijkstra's algorithm, Ford-Fulkerson, and others.
- **Analyze a new problem and select an appropriate problem-solving technique.** Be able to construct an algorithm to solve the problem, prove the correctness of their algorithm, and analyze the asymptotic behavior of the algorithm.
- **Analyze behavior of algorithms with recursive, and other more advanced properties.** Solve recurrence relations for their closed form either manually or via the Master Theorem. Prove both upper and lower bounds on the behavior of such algorithms.

Note: department course specification also includes an introduction to machine learning and topics on ethics and machine learning.

Grading Breakdown

Note: unlike some courses/semesters, each assignment gets a numeric score, and they're combined to get an overall course score from 0-100

- Quizzes: 50%
- Programming Assignments (PAs): 20%
  - Reminder: one per unit
- Problem Sets (PSs): 25%
- Activities and Course Feedback (5%)



Quizzes

In-person, in your assigned section's classroom

- Feb. 29: Quiz 1 (Graphs), Quiz 2 (Divide and Conquer)
- Apr. 11: Quiz 3 (Greedy), Quiz 4 (Dynamic Programming)
- **May 2 (7-10 pm)**: Quiz 5 (Network Flow, Machine Learning), Re-take Questions

"Re-take" Opportunity for Quizzes 1-4

- During final exam period (along with Quiz 5)
- Available only if you made a reasonable first attempt on quiz
- Answer 2-3 questions on that unit
- Grade on those questions will replace up to 10 missing points on score from first attempt of that unit's quiz
  - Example: First attempt score is a 75/100. You get 5 of 10 on re-take. Final score: 80/100.

Problem Sets (PSs)

Goal: Think carefully and practice problem solving with techniques discussed during the unit

Grading: We will assign points for each question based on a rubric

What are they like? A set of approximately 2-3 questions

- Concept-level questions, details of what we've studied
- Design an algorithm for a given problem (pseudocode, words)
- Analyze an algorithm (mathematically), provide a proof of correctness

Collaborate? Yes, in groups of up to 5

- But everyone must prepare and submit an individual write-up
- Submissions must be formatted in LaTeX! (more later)

Programming Assignment (PAs)

Goal: Explore one or more topics from a unit by applying and implementing it

Grading: Primarily based on passing test-cases on Gradescope

- Include comments: any sources, collaborators
- We may ask you to answer short discussion questions about your solution

Languages accepted: Python (3.10.6), Java (19.0.2)

Collaborate? Not in writing or debugging the code, but…

- Can discuss **the problem** and **the overall strategy** with other students
- Must list these people in your submission
- Cannot share code, look at each other's code! (Read syllabus carefully!)

PS0 – LaTeX Warmup

PS0 is out this week, due next week!

- Learning LaTeX
  - LaTeX (pronounced “LAH-tech”) is a markup language where you mix text and commands in a .tex file, run process the file to get a publication-quality PDF output file
  - Created to format mathematical symbols and texts
  - You can create .tex files and run LaTeX in the cloud using a site called Overleaf
- PS0 is due next week

Academic Integrity

Collaboration Encouraged!

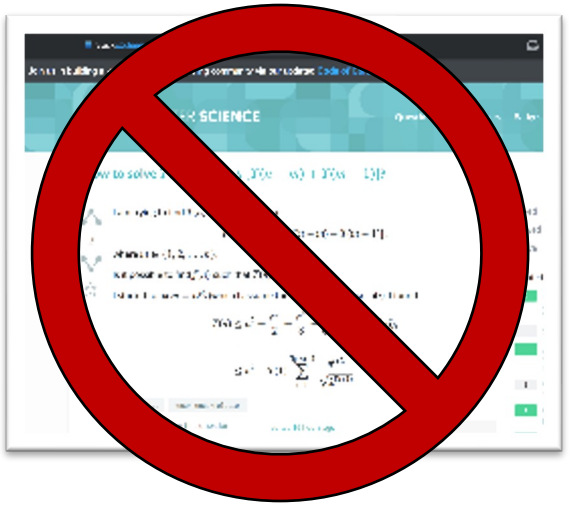
For PSs: groups of up to 5 per assignment (you + 4)

- List your collaborators (by name and UVA computing ID)
- OK to discuss problem, approach to solution, even details about solution, but… collaboration is "whiteboard only"
- Write-ups (.tex files) must be created **independently**
- **DO NOT** share written notes / pictures / code / etc
- **DO NOT** share documents (ex: Overleaf)

Be able to explain any solution you submit!

DO NOT seek published solutions online

See syllabus about online code examples



Academic Integrity

Collaboration Encouraged!

For PAs: you can talk, strategize, design,… together, but creating and debugging code must be done **alone**

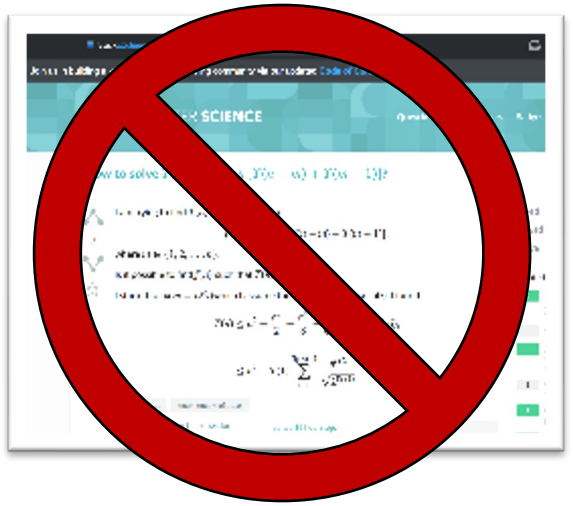
- List your collaborators (by name and UVA computing ID)
- **DO NOT** share pieces of code or programs or files
- **DO NOT** share debugging of code

Be able to explain any solution you submit!

DO NOT seek published solutions online

See syllabus about online code examples

See syllabus about generative AI



Late (and Early) Policy

PAs and PSs can be submitted early for a **2% bonus**

- *Final submission* to Gradescope must be **48 hours** before the deadline
- Start early, meet with your groups, come to office hours as needed

Late (and Early) Policy

But, sometimes things happen in life, and we can't meet the deadline

- PAs and PSs may be submitted 48 hours after the deadline
- Requests must be made using our online form (see course website)
  - Must include a valid reason
  - Must be made before the deadline
  - Must acknowledge getting an extension for the assignment
  - Must include significant work (uploaded to the form)
- No penalty for late submissions within this window

Generative AI

For each assignment, we'll say if and how generative AI tools can be used

- When you use a tool, you must properly document and credit the tool (including the prompt)
  - Failure to do this is serious violation of academic integrity!
- Keep in mind tools may produce incorrect results, or may result in plagiarism or copyright violations
- It is your responsibility—not the tool's—to assure the quality, integrity, and accuracy of work you submit
- **Having said all this, in CS3100 we will examine how such tools can be useful in the study and practice of algorithm design and analysis**!

Feedback

We professors are not course dictators, more like civil servants.

We're open to any suggestion to help you learn.

Let us know!

- In person
- Piazza
- Course staff email: cs3100@cshelpdesk.atlassian.net
- Email: rp6zr@virginia.edu or jrhott@virginia.edu
  - PLEASE: put CS3100 in subject line of all emails
- (No Discord DMs, please)

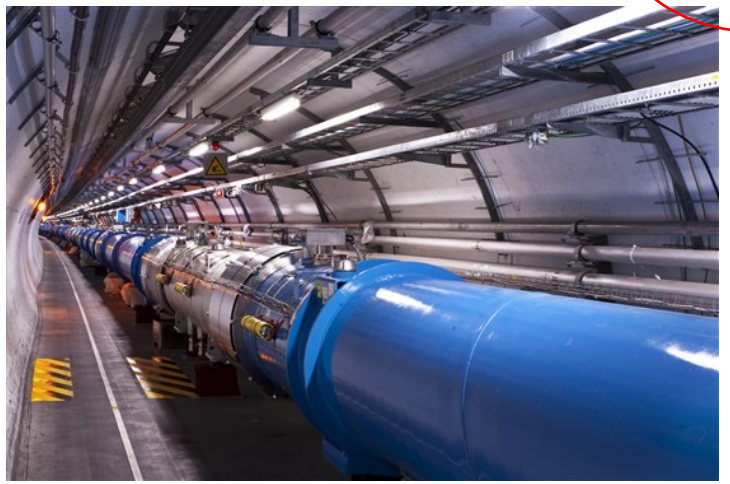
Pre-Course Survey

Let’s Get Acquainted

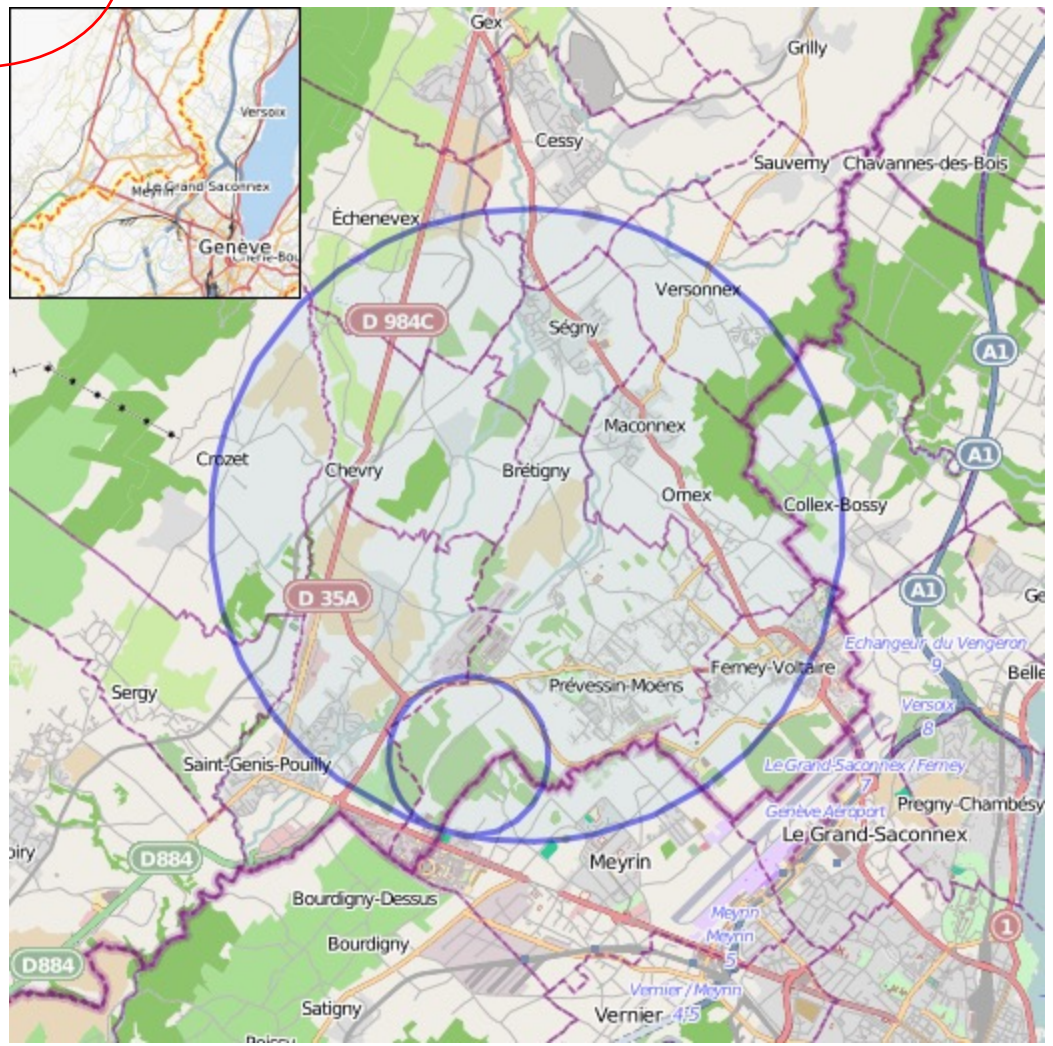
Would you like to know a bit about me? ☺

A motivating problem

Need an accurate approximation → π



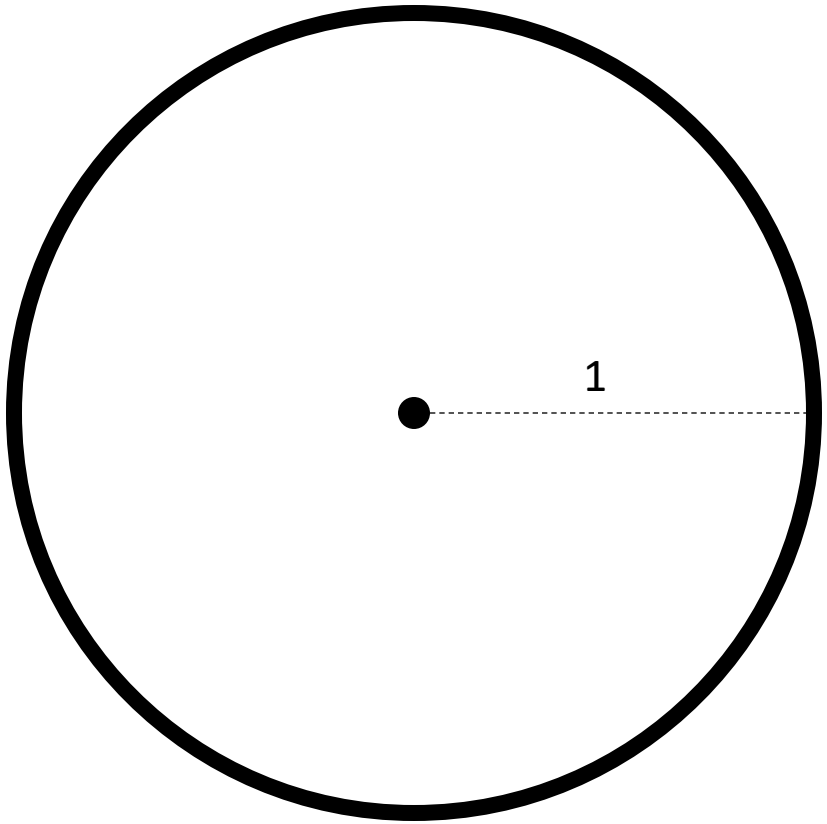
How much concrete do I need?



4.3km (2.7mi) diameter

π Approximation Algorithm

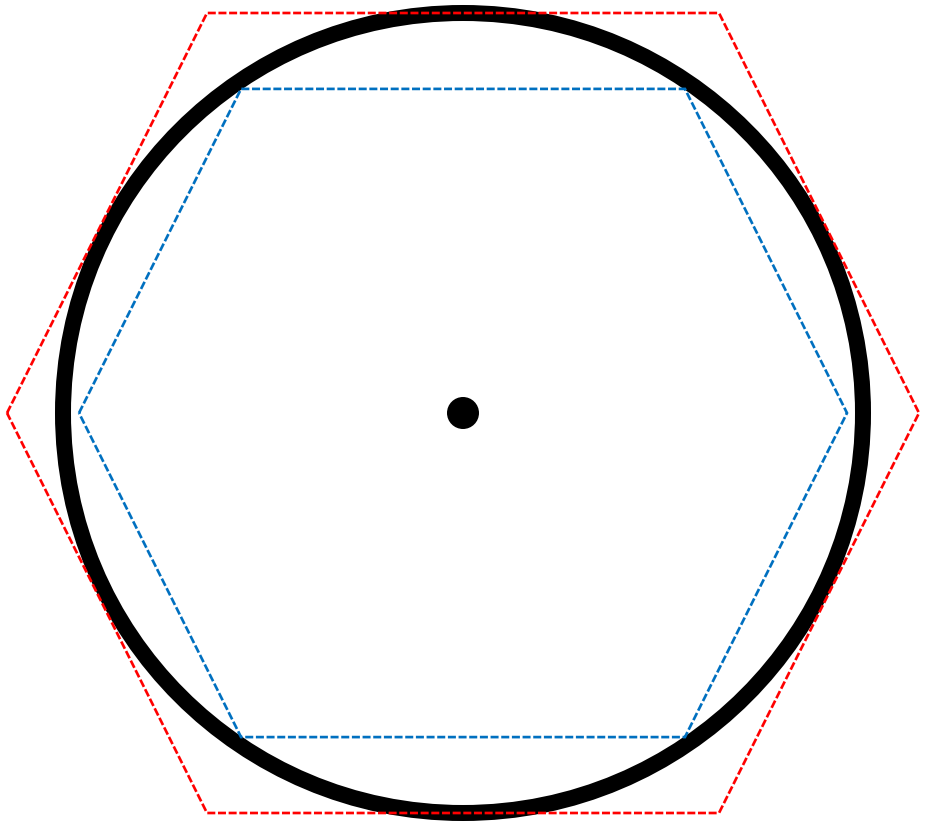
$\pi = 3.14159265359\ldots$



Circumference = 2π

π Approximation Algorithm

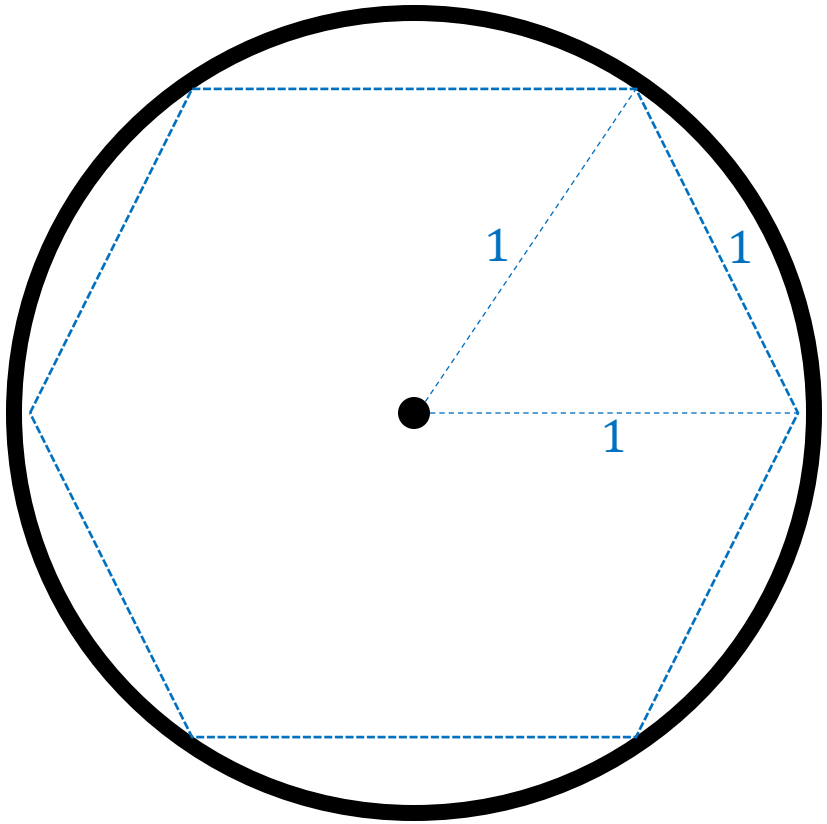
$\pi = 3.14159265359\ldots$



Perimeter > 2π > Perimeter

π Approximation Algorithm

$\pi = 3.14159265359\ldots$



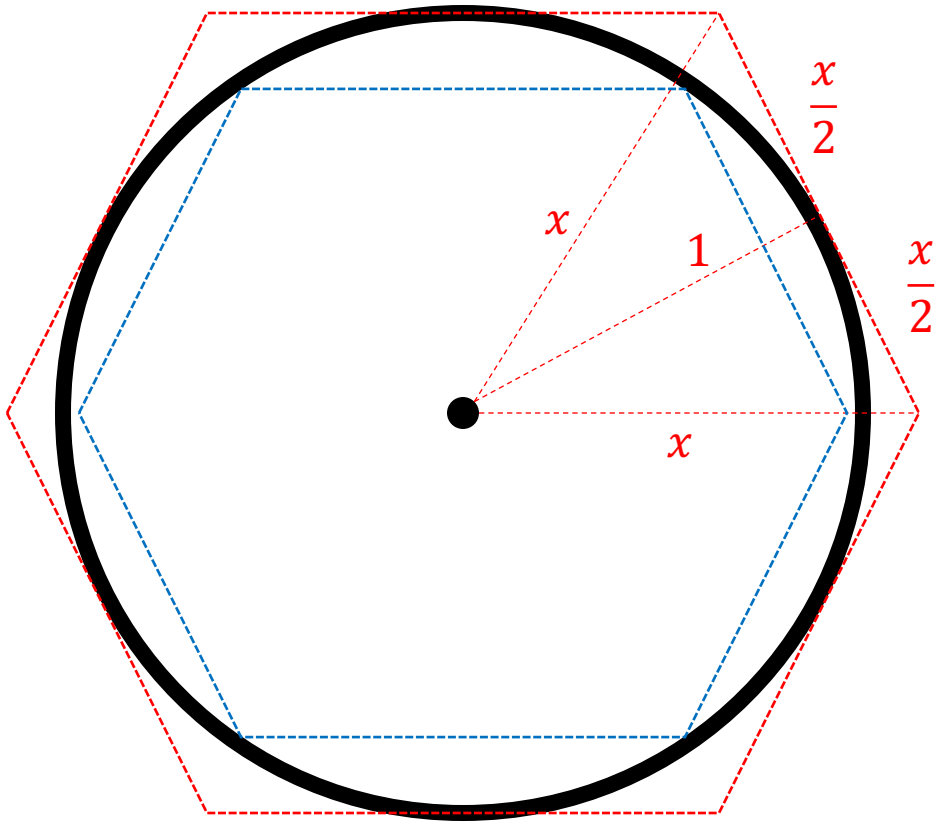
$2\pi >$ Perimeter $= 6$

π Approximation Algorithm

$\pi = 3.14159265359\ldots$ 1 digit correct

Solve for x

$$x = \frac{2}{\sqrt{3}}$$

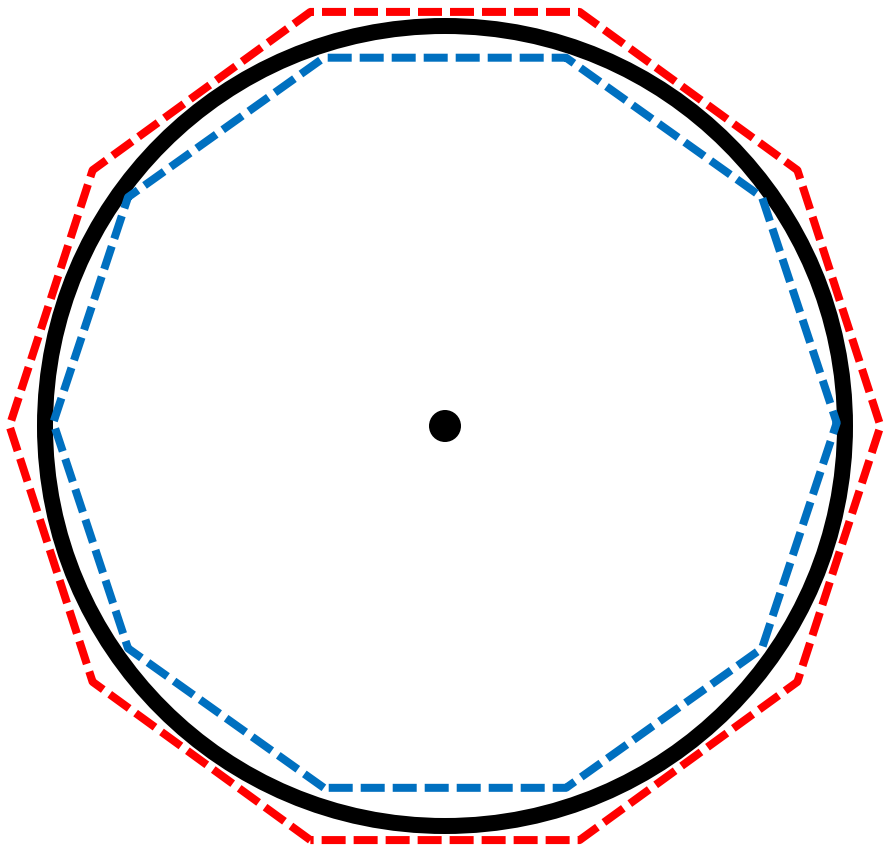


$$\frac{12}{\sqrt{3}} = \text{Perimeter} > 2\pi > \text{Perimeter} = 6$$

$$3.46 > \pi > 3$$

π Approximation Algorithm

$\pi = 3.14159265359\ldots$ 3 digits correct



$$6 + \frac{20}{70} = \text{Perimeter} > 2\pi > \text{Perimeter} = 6 + \frac{20}{71}$$

$$3.14285 > \pi > 3.14084$$

How to analyze this approach?

How fast do we “converge?”

How much work is needed to do better?

Better π Approximation (Ramanujan)

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \sum_{k=0}^{\infty} \frac{(4k)!\,(1103 + 26390k)}{(k!)^4\, 396^{4k}}$$

$\pi = 3.14159265358979323846264338327950288419716939937510582097494459$

$k = 0$

$\pi \approx 3.1415927$

8 digits per iteration!

$k = 1$

$\pi \approx 3.141592653589793 8$

Next Steps

Review

Next class we will:

- Review how we analyze algorithms, including
  - Asymptotic notation, order classes, worst-case, etc.
- Remind you how all this works by applying/reviewing it with sorting and searching you did in previous course
  - Sequential search, binary search
  - Sorting: insertion, quicksort, maybe mergesort
- Review and/or introduce proofs from CS2120
  - Proof by induction
  - Introduce you to how this can be used for proving algorithm correctness