CS 374 Syllabus

Outline

- Course goals
- Logistics
- Teams
- What's this class about? – Expanding your toolbox
- Methods of evaluation
- Types of assignments
- What does success in CS 374 look like?

Course Goals - Theoretical

- Be able to write proofs about programs.
- Be able to compare the asymptotics (Big-O, Big-Theta, Big-Omega, dominance relations) of standard functions, including polynomials, logarithms, exponentials, and the factorial function.
- Be able to justify algorithm correctness and analysis formally by relating algorithms to their RAM models.
- Be able to reason about concepts of tractability and intractability, like polynomial-time, polynomial-space, and NP-completeness.

Course Goals - Practical

- Be able to implement standard data structures and understand how they affect the efficiency of algorithms.
- Be able to implement standard sorting algorithms and compare them for performance.
- Be able to use graphs to solve algorithmic problems.
- Know the use-cases, non-use-cases, and analysis techniques of greedy algorithm & "Divide and conquer" algorithm design.
- Be able to solve algorithmic problems and optimize algorithms using dynamic programming.
- Be able to compare algorithm runtimes empirically.

The basics: where, when, how

Section -01:

- **Days**: Tuesdays & Thursdays
- **Time**: 1:30-3:10 pm
- **Location**: Galileo 105

Section -97:

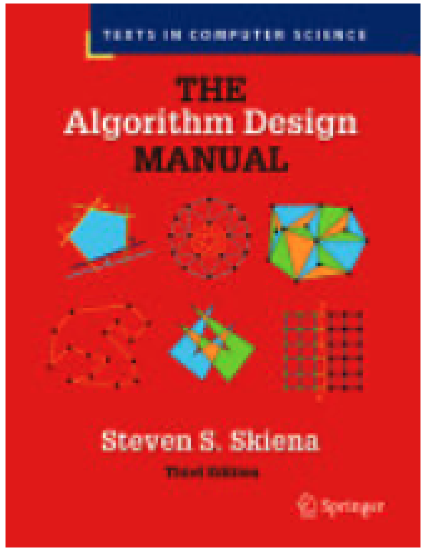
- Online, asynchronous with synchronous check-ins.

Office Hours

- I will be **in my office** ready to **answer questions** or **talk about anything**.

- **Tuesdays**: 10-11:30am
- **Wednesdays**: 11-12:30pm
- **Location:** Galileo 103A
- **I am available many other times**:
  - Find a time that works for you.
  - Zoom is an option, just let me know.
  - Show where the link is on Canvas.

The textbook



The Algorithm Design Manual

Authors: Steven S. Skiena

Edition: 3

ISBN: 9783030542559

Where to Purchase

Bookstore, Amazon, AbeBooks (recommended), etc. Previous editions ok and encouraged if it's cheaper!

Accessibility

- I am willing and happy to accommodate for accessibility needs.
- Both officially from SDS and other reasonable accommodations.

- Let me know as we go if anything comes up that needs adjusting.
- If it seems like I've forgotten something, please remind me ASAP.

- Lectures are recorded. [Show where the link is.]
- I try to post things ahead of time, as best I can.

Teamwork Makes the Dream Work

Teams

- For the first ~3 weeks of class, you will be paired up randomly.
  - For in-class activities, algorithm history presentations, etc.
  - Gives you a chance to meet people
- At the start of Unit 2, you will be assigned to a team.

Teams

- What a team does:
  - Sits together in class
  - Works on in-class activities together
  - Meets outside of class to discuss (hopefully!)
  - Works on projects together

- A consistent group for you to work with.
- Your preferences will be considered.
  - Via Google form, assigned end of week 2.

Go through the syllabus document

What's CS 374 about?

Outline

- Expanding your toolbox
- Subjectivity
- What's a "good" answer?
- Communication

Expanding your tool chest

- To solve an algorithmic problem, you need to apply both:

- Understanding
- Language

Example: Too much to do, too little time

- Dr. Roscoe's Tuesday todo list:
  - Prep CS 374 Tuesday lecture
  - Write a sentence for research paper
  - Decide Math 104 homework problems
  - Prep Math 104 Tuesday lecture
  - Organize Canvas course
  - Knit hat
  - Read a research paper

Example: Too much to do, too little time

- One way to solve: enumerate all possibilities
  - How many ways can I order my todo list?

- Not useful for getting stuff done.
- I need a **scheduling algorithm**.

What's an algorithm?

- An **algorithm** is a **process to accomplish a specific task.**

Structure is needed

- To get anything done, I need a ***data structure***.

- If you work with data, how are you storing it?
  - List, graph, queue, etc.

- I choose **priority queue.**

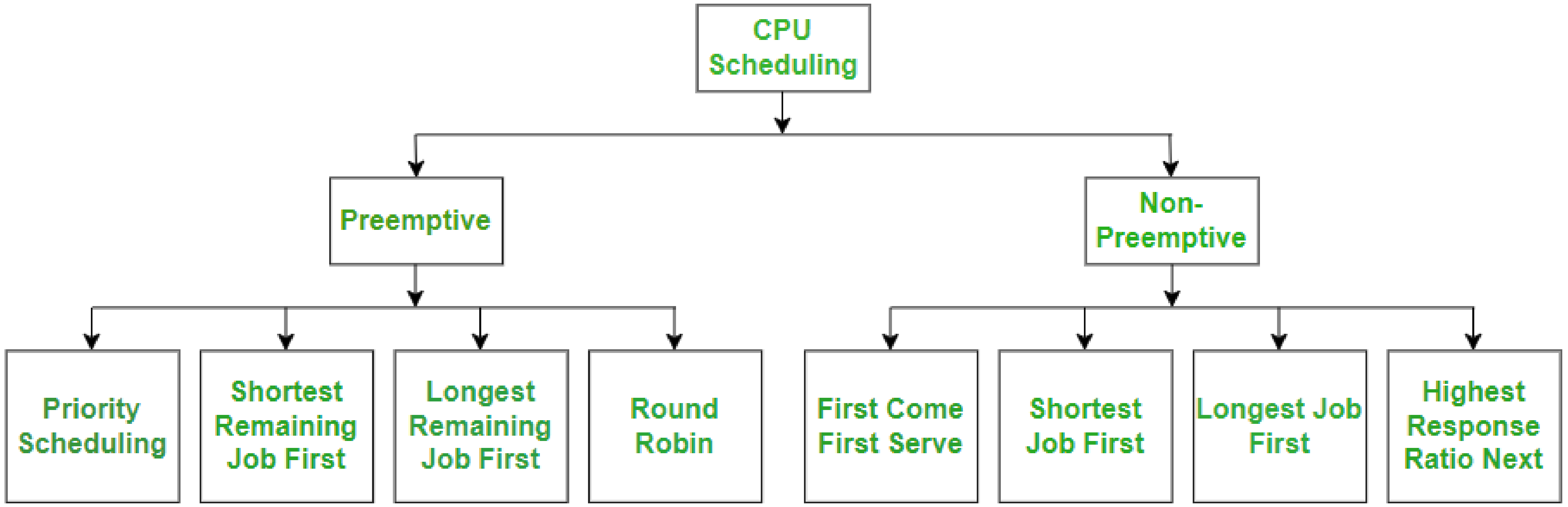
Priority Queue

- I will label my data with priority (1 = high)

- Dr. Roscoe's Tuesday todo list:
  - Prep CS 374 Tuesday lecture - 1
  - Write a sentence for research paper - 2
  - Decide Math 104 homework problems - 3
  - Prep Math 104 Tuesday lecture - 1
  - Organize Canvas course - 1
  - Knit hat - 3
  - Read a research paper - 2

Desired Functionality

- When I dequeue from my priority queue, it gives me the next thing I need to work on.
- ...there are many options.



Subjectivity and Biases

How to choose what to do?

- My ordering of tasks may not be what you would choose.
- The solution we choose is not necessarily objective.

- Conflicting priorities ➔ No easy answers.

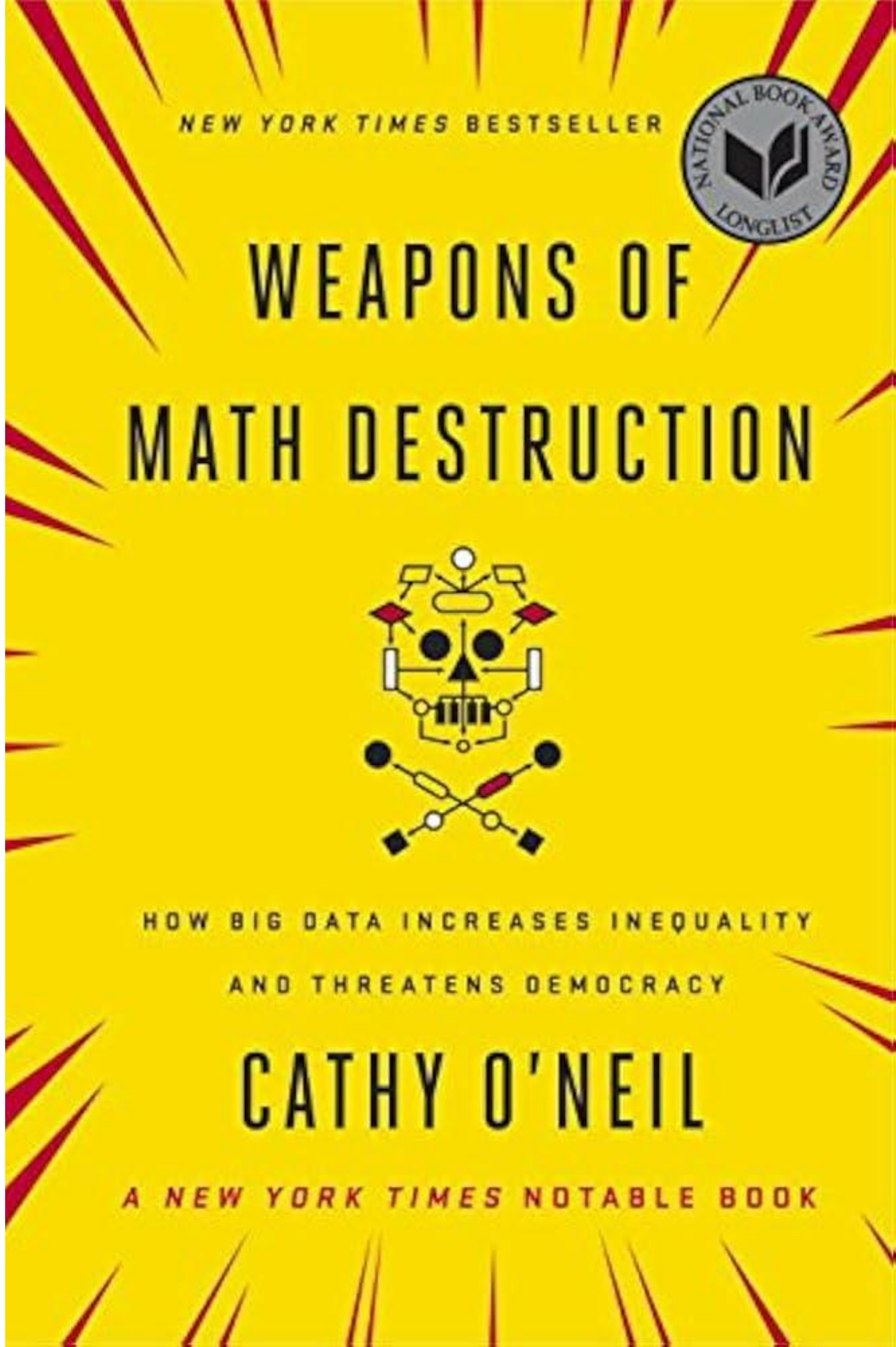
You have assumptions

- You bring assumptions to the table about how the structure works.
- You have biases about what is "easy" to compute
- Others (your peers/industry professionals/the internet) may have differing opinions.

Book recommendation

"An algorithm is an opinion formalized in code."

- The tools you use,
- How you choose to use them,
- They're *your* decisions.


NEW YORK TIMES BESTSELLER
NATIONAL BOOK AWARD LONGLIST
WEAPONS OF MATH DESTRUCTION
HOW BIG DATA INCREASES INEQUALITY AND THREATENS DEMOCRACY
CATHY O'NEIL
A NEW YORK TIMES NOTABLE BOOK

There are few wrong answers in this class.

- …There are simply *less good* answers.

- Scheduling problem alternatives:
  - Use hashmap instead?
  - Sort by time estimated to take?

Specificity is key.

- **Read between the lines** to answer questions and solve problems.

- What has been stated to be important?
- What could be inferred to be important?
- What has not been explicitly stated, but affects a "good" solution?

- This class will have broad questions, but your solutions must be specific.

How to get there

- Think inside the box.
  - Apply your known tools.
  - Try to logically approach the problem with what you know.
- Think outside the box.
  - Be close readers: read between the lines.
  - Investigate, is there another way this could be solved?
  - Can you structure the problem differently?
  - Can you shift your perspective?
- Ultimately, you must use your brain.

There are few wrong answers in this class.

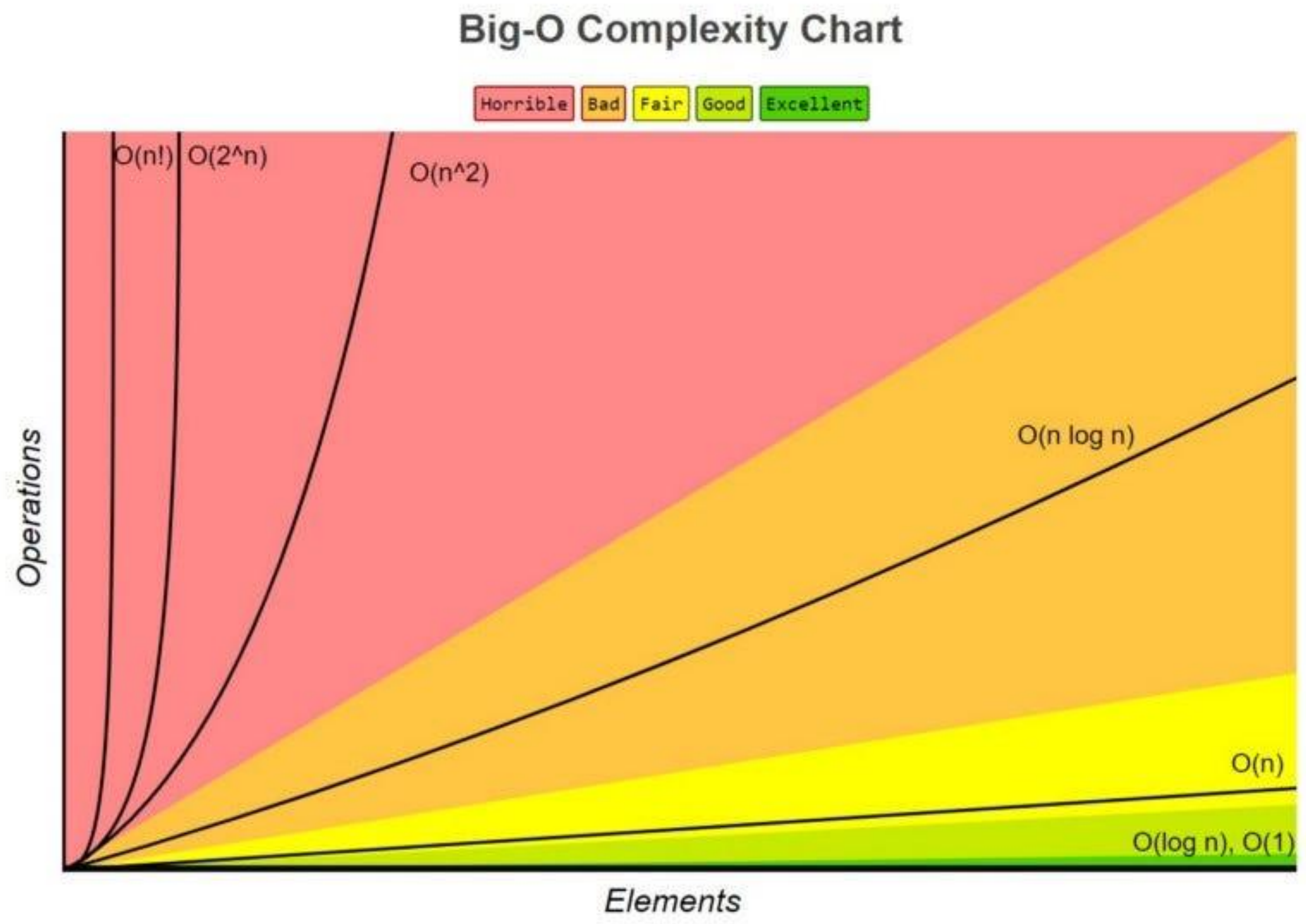
- …There are simply *less good* answers.

…Versus…

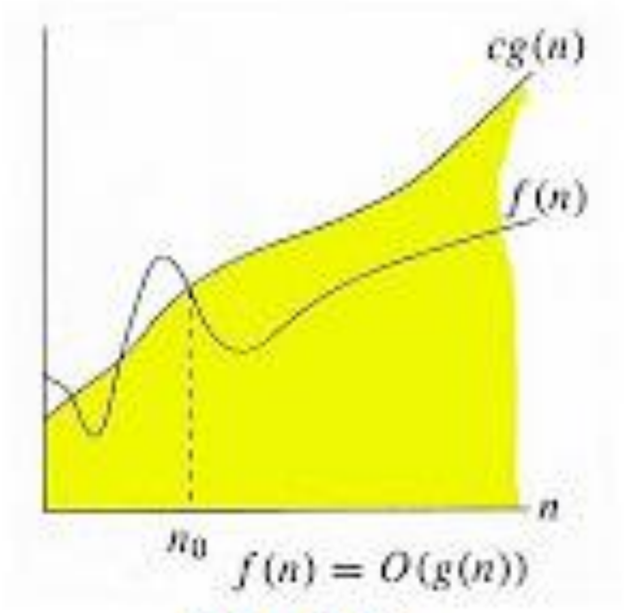
"An algorithm is an opinion formalized in code."

What does "good" mean, anyway?

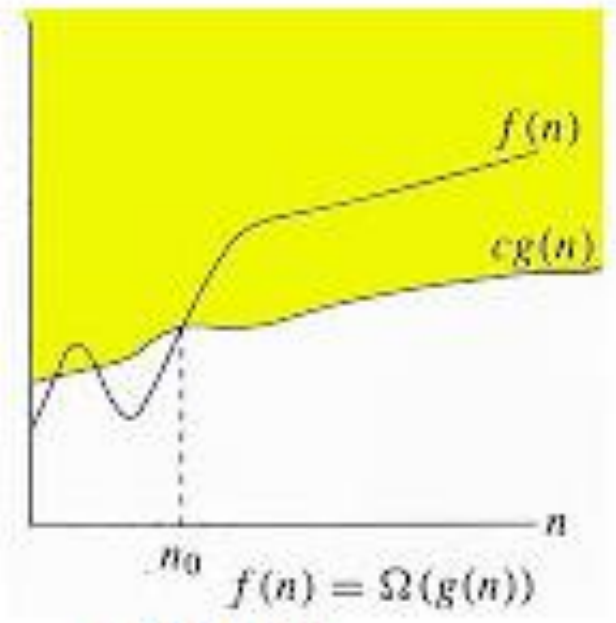
End of CS 222


Big-O Complexity Chart
Horrible
Bad
Fair
Good
Excellent
O(n!)
O(2^n)
O(n^2)
O(n log n)
O(n)
O(log n), O(1)
Operations
Elements

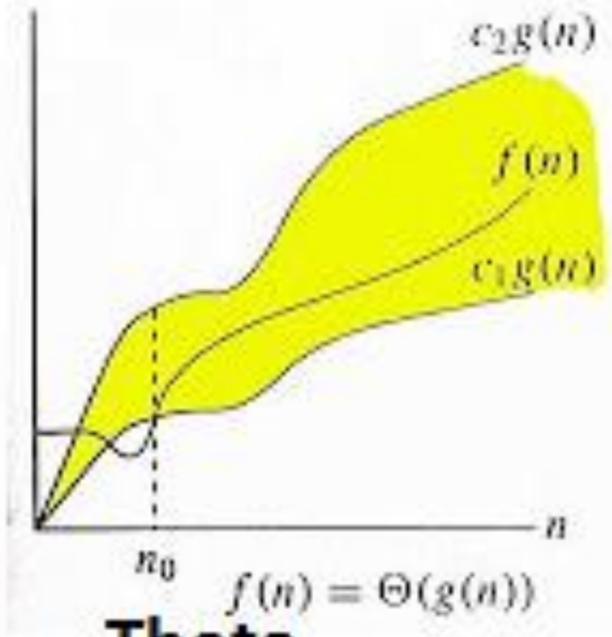
CS 374



Big Oh



Omega



Theta

In Comparison

- You submit an answer, correct $\rightarrow$ It's slow
- Your neighbor submits an answer, correct $\rightarrow$ Much faster

- You get less points
- Neighbor gets most of the points

- Your answer's not technically *wrong*, just not as good as it could have been.

A warning

- Your answers must be **correct…**
- And they must be **efficient.**

- Solutions that differ from mine, but get the job done adequately, in the targeted efficiency range, will get most of the points.

Okay, convince me!

This is not (quite) a writing class

- It is a crucible.

- You will practice writing and presenting code.
- You will practice writing and presenting algorithms.

Why bother with writing and presentations?

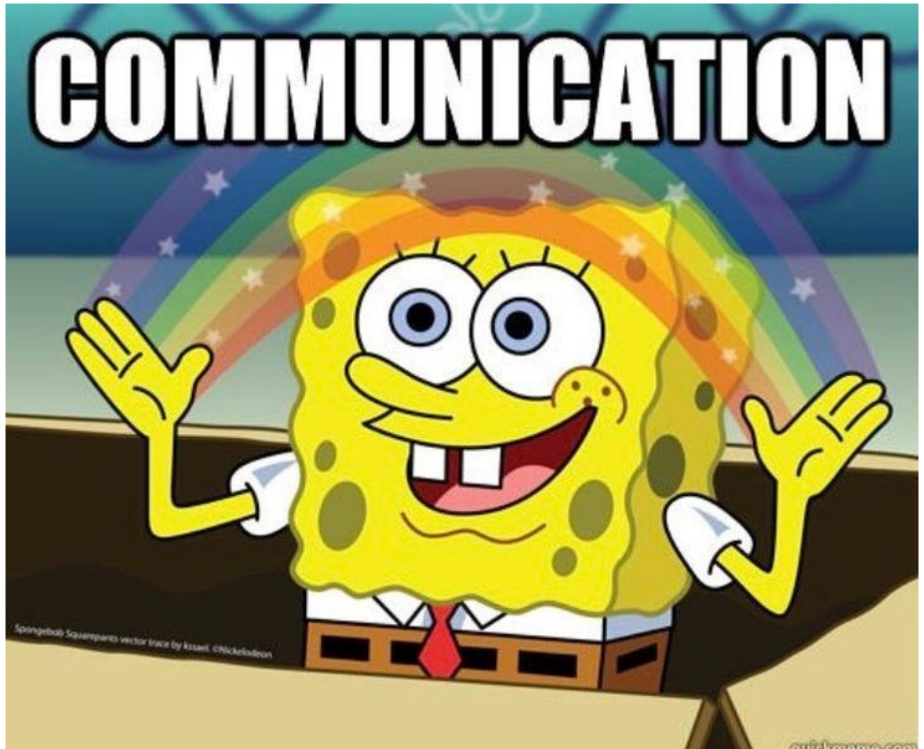
- The goal of this class is for you to understand the workings of algorithms.
- It's important to know how to implement things.
- But in industry and elsewhere, it's not *just* the code you write.
- It's also your ability to explain it to peers, managers, and people with no prior knowledge.
  - Succinctly explain
  - Determine what to include, what not to include.

This sucks. It's a lot of work.

- Yes, but that's learning.
- Stretch yourself to do things you are unfamiliar with.

What's a "good" answer?

- Technical components, yes…
- …also explaining it.



How to communicate effectively

- Be concise.
  - Simple sentences. Nothing superfluous, nothing over-explained or under-explained.
- Be specific.
  - Know your audience: be prepared to answer any questions you think they might have.
  - Clearly transition to each step of your explanation.
  - If you skip any steps, you do so with reasonable knowlege that they are trivial.
- Present well.
  - Proofread so there are no typos or incomplete sentences.
  - Present it nice and not messy.
  - Read aloud any sentences with code or math so that it flows.
  - (if giving a presentation) Dress professionally.

How to get there: Study Tips

- Get your ideas down on paper.
- DO NOT turn that version in.
- Take a (brief) break. Walk down the hallway and back.
- Sit down and read it aloud. Explain it to a rubber duck, or your favorite stuffed animal. Explain it to your roommate, your parent, your housemate.
- They probably don't understand it. Assuming they understand all the preliminaries, figure out which most important parts they don't understand.
- Go to the writing center.
- Revise it.

Where to go from here?

- "An algorithm is an opinion formalized in code."
- There are few wrong answers in this class…there are simply *less good* answers.

Methods of Evaluation

Exemplar Rubric

- Efficiency of solution
  - The proposed solution is completely efficient. There is no possibility of a better implementation.
- Completeness of solution
  - The proposed solution is complete. Every edge case is considered. There are no cases the writer forgot to consider. Given any possible correct input, the output will be produced as described.
- Generality of solution
  - The proposed solution is specified with what counts as valid arguments. It will correctly apply to any given arguments, so long as they are valid.

Exemplar Rubric, cont'd

- Conciseness of writing
  - Writing is perfectly concise. There are no superfluous words and nothing is over-explained.
- Specificity of writing
  - Solution is perfectly specific; there are no questions that should be additionally answered. Transitions are clear. If there are any skipped steps, they are clearly understood to be trivially inferred by the reader.
- Presentation of writing
  - There are no typos in the proposed solution. The layout is not messy. All sentences are complete. Any math or code descriptions present flows cleanly with explanations.

Types of Assignments

- Written
    - Weekly
- Verbal/Presentation
    - Frequently informal, in class
    - Project-based
- Coding
    - Project-based

What does success look like?

What does success (A) look like?

- Show up
- Pay attention, close technology
- Speak up and ask questions
  - There are no stupid questions
- Go to office hours frequently
- Form a study group
  - Perhaps with your team?
- Start assignments the day they're assigned.
- Check the rubric. Follow all directions.
- Turn in on time.
- Don't use AI, OR use strictly within defined parameters.
  - Only copyediting or brief consideration of other avenues.
  - No using for broad interpretation.

What does mediocrity (B-C) look like?

- Miss several class periods
- Occasional shopping/doing the Seminar reading
- Ask only when you have a genuine question.
- Go to office hours maybe once.
- Do things on your own
  - Maybe talk to others when you have a question
- Start assignments a few days before it's due.
- Glance at the rubric, but do all the instructions.
- Turn in on time.
- Mostly use AI within parameters.
  - Use it to help guide your responses
  - Lean a little too much on its interpretation of the questions.

What does failure (D-F) look like?

- Miss a majority of class periods
- Go to class but rarely pay attention.
- Never go to office hours unless required.
- Do things on your own
- Wait until the last minute to start assignments
- Assume that turning *something* in will give you enough points to pass.
- Paste assignment directions into AI

My advice

- Decide today where you want to stand.
  - This class is a foundation for any career in tech.
- Block off your calendar.
- Put the time in to make it happen.

I (the instructor) pledge to…

- Come to class on time
- Be available for questions during class
- Be in my office when office hours are scheduled
- Communicate if I cannot make a scheduled commitment
- Return grading within 1-2 weeks of the due date
- Maintain high expectations of you (students)
- Apologize when I make a mistake, and work to fix it.

We're doing the best we can with the tools we have.

Let's go expand our set of tools.