

CS 256: Algorithm Design & Analysis

Lecture 1: Welcome & Logistics

Shikha Singh

Announcements

- **Welcome to the first class!**
- Things to grab:
 - **Exercise I** (due next lecture)
 - **Syllabus**
- Make sure to sign the **attendance sheet** that is circulating
- Course website (<https://williams-cs.github.io/cs256-f26/>)

About Me

- Go by Shikha or Prof Singh (she/her)
- Recently returned from parental leave
- Typically teach intro and theory courses
 - CS 134, CS 256, CS 357, CS 361
- Research area
 - Data structures
 - Algorithmic Game Theory
 - Recently: *learning-augmented algorithms, algorithms for AI-guardrails*

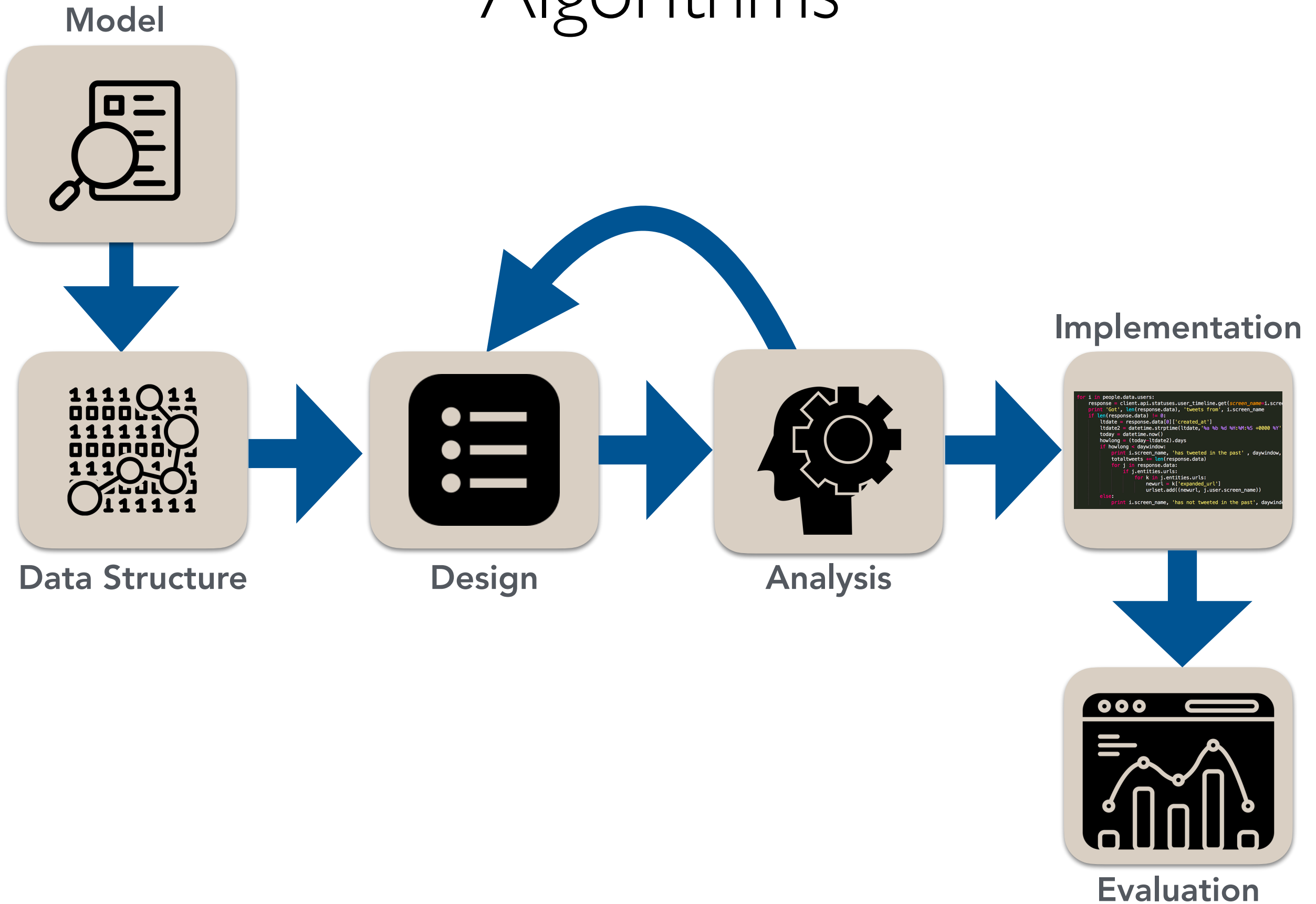


two human intel builds



About CS256: Algorithms

Algorithms



CS256 Overview

- How to model/formalize computational problems
- Learn about various algorithmic design paradigms
 - Design new algorithms using them
- **Analysis** and **proofs**:
 - Focus of the course!
 - Learn how to prove that an algorithm is correct
 - Learn how to analyze formally the time and space complexity of algorithms

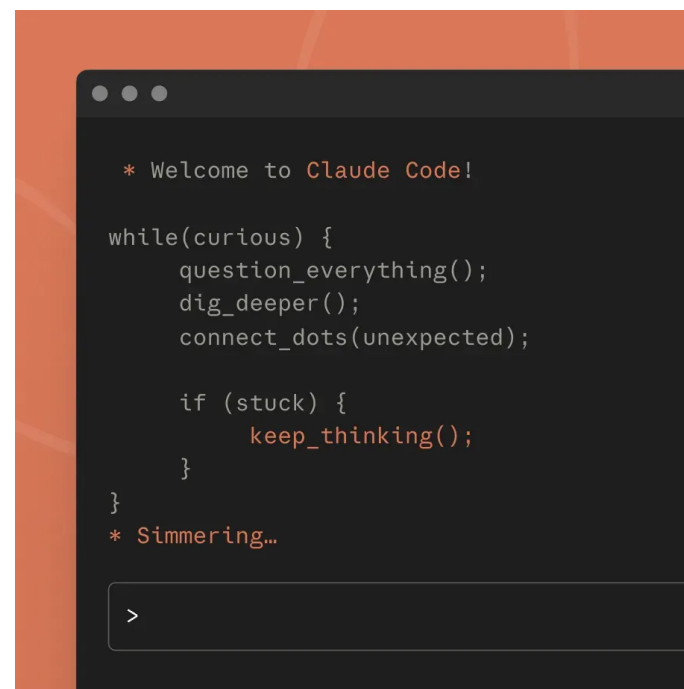
Prerequisites

- CS 136: Data Structures
 - practice with common data structures
 - arrays, lists, stacks, queues, graph data structures, etc
 - simple sorting and searching algorithms
 - how to break down computational problems into smaller pieces
 - asymptotic Big O complexity of common algorithms
- MATH 200/220:
 - Proof techniques, elementary combinatorics
 - Logic, sets, functions, relations, etc
 - Elementary graph theory

Hard prerequisite! If you have not taken, please come talk to me

CS Education in the Age of AI

- Lots of anxiety about the "worth" of a CS education in the age of AI
- Easier and faster than ever to **generate** (almost functional) code
 - Means code is likely to become more universal
- Computer science $\neq$ computer programming

A screenshot of a code editor window with a dark background and orange accents. The text inside the editor is as follows:

```
* Welcome to Claude Code!

while(curious) {
  question_everything();
  dig_deeper();
  connect_dots(unexpected);

  if (stuck) {
    keep_thinking();
  }
}

* Simmering...

>
```

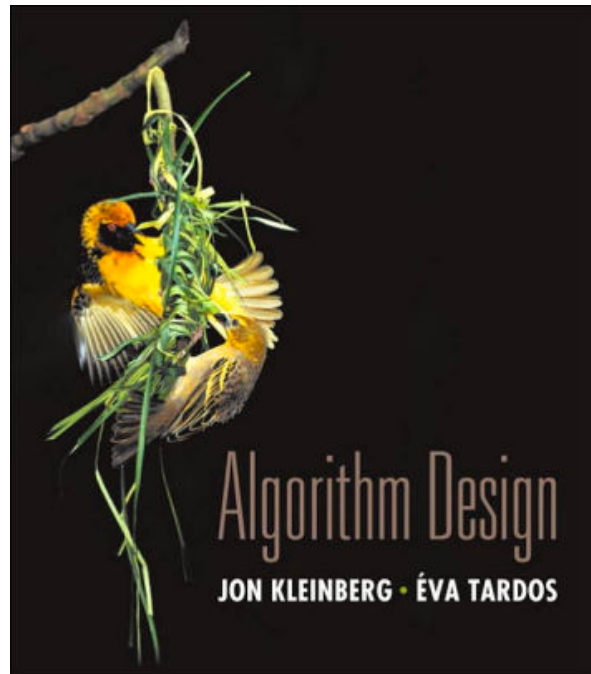


Correctness is Critical

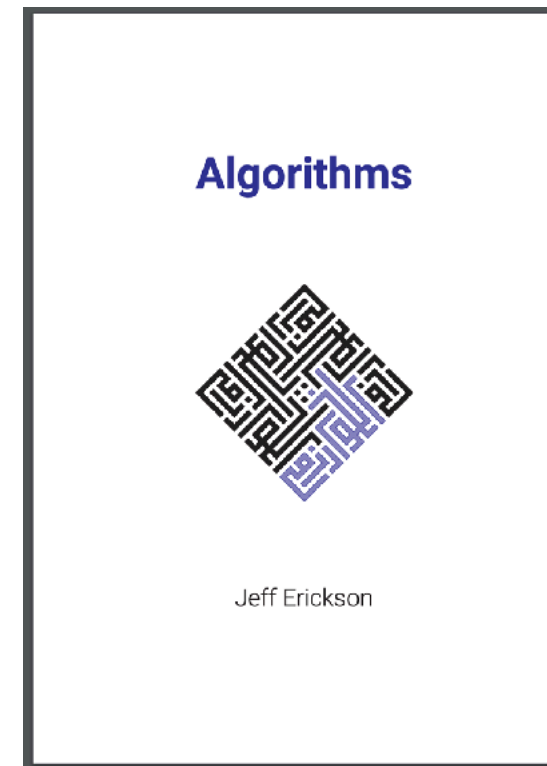
- CS is about understanding fundamentals of computation
- CS Skills that have become more important than ever:
 - **Modeling** and **formalizing** abstract problems into concrete tasks
 - Understanding "what" and "why" behind algorithmic decisions
 - Verifying **correctness**: is the algorithm following the specifications and how to prove correct behavior or find counterexample/bugs
- CS256 provides training with all of these skills!
 - Specific topics are just practice with the general cognitive process involved in these steps

Course Logistics

Textbooks:



Three copies reserved in the Schow library for reference



Available online at <http://jeffe.cs.illinois.edu/teaching/algorithms/>

Slides: Kleinberg and Tardos book has excellent slides for reference that I'll also be borrowing a lot from.

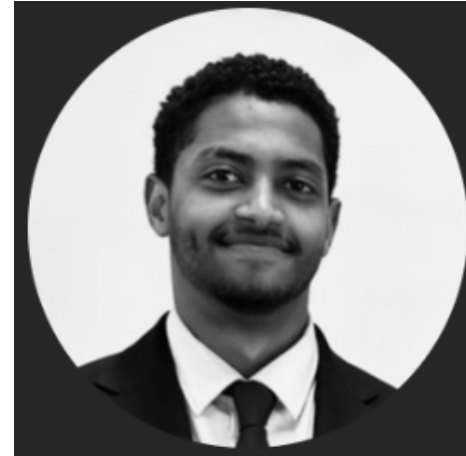
TA Team & Hours



Charlotte Chen



Bennet Ptak



Nathan Gebreamlak



Ronald Deng

- Great TA team this semester!
- All TA hours will be held in TCL 206
- Schedule will be posted on course calendar

Course Logistics

Course website: <https://williams-cs.github.io/cs256-f26/>

Grading breakup:

- Daily exercises: 10% (lowest 4 will be dropped)
- Weekly problem sets: 10% (lowest 2 will be dropped)
- In-class tests: 10+20+20% (weight re-adjusted +/-5% towards higher score)
- Scheduled Final (30%)
- Attendance and Class participation (+/- 5%)
 - 3 excused absences no-questions-asked
 - For excused absence: email ahead of time

Daily Exercises & Readings

- Must be done **individually** (no discussion with classmates)
- Handed on paper (almost) every lecture
- Must hand it in the next lecture
 - No make ups/remote/late submissions: will drop lowest four
- Typically based on reading of the next lecture
 - You must read the assigned algorithm and answer mechanical questions about its execution
 - Helps you build a foundation for the lecture's analysis
- **Rubric:** Check, check plus, check minus grading

Weekly Problem Sets

- Due Weds at 10 pm
- Must be done **in pairs**
 - Partners assigned on GLOW
- Longer-format problem and proofs
 - Practice with problem solving, technical writing and to prepare for exams
- Rubric linked [here](#)
- Pair problem solving best practices



Overleaf and Gradescope

- Solutions must be typeset in LaTeX using the template provided
 - Overleaf link to assignment (you can make a free account)
 - Must copy and create your own "project" for each problem set
 - You can invite your partner as a collaborator on your "project"
- A **joint PDF** should be submitted on Gradescope
 - Sign up using the course code XEVXD2
 - **IMPORTANT.** Assign ques to pages of the PDF when submitting
- Assignment 0 is out (Individual assignment)! Due Wed 09/16

Written Tests

- Three in class tests (50 mins) roughly at the end of each month
 - Exam 1 (October 5, please correct in syllabus handout)
 - Exam 2 (October 30)
 - Exam 3 (Nov 20)
- Each will test the unit covered that month
- A cumulative final exam will be held during finals period

About Class Participation

- I like interaction in my classes!
- Many opportunities to participate in the class through:
 - Comments/questions during lectures
 - Group problem solving on the board
- Attendance policy:
 - **Required** but flexible
- Class participation does not mean dominating classroom discussions or interrupting your peers

Bottom line: Help create a vibrant, positive and inclusive classroom community!

AI Policy in the Course

- **Not allowed for any graded assigned problems**
 - HW is meant for you to practice concepts and you are not permitted to use AI help for any solution to turn in as your own work
- General advice:
 - May use it for studying but caution: hard to identify subtle mistakes or **concepts that are out of scope**
 - Useful in **small doses** and **specific mechanical tasks**: worked out examples/ counterexamples, execution trace of an algorithm, etc
 - Detrimental to learning: models are trained to maximize "engagement" and ultimately end up wasting your time
 - Illusion that "feels" like you are being productive but ultimately real learning happens the hard way: **thinking yourself** than delegating it

Friction, frustration & Repetition

- Watching someone do a task **does not** lead to learning



How to Succeed

- Practice, Practice, Practice
 - Exercises and problem sets are low stakes practice
 - Doing these yourself is the single most important way to learn the material and prepare for the tests
- Come to class prepared (do the readings)
 - Ok to skim before class, read more closely after
 - Two textbooks give complementary presentations
- Ask for help early and often
 - Add the course calendar to your google cal
 - Lots of instructor and TA help hours

Academic Integrity Policies

- Read academic integrity policies on the syllabus
- Gist:
 - No collaboration allowed on daily exercises
 - Can collaborate fully (including on the write up) with your problem set partner
 - Can discuss high-level ideas about problem set with other classmates but cannot engage in any step-by-step writing
 - Only refer to course materials and textbook when attempting problems for exercises/problem sets
 - If concepts or definitions outside of what is covered is used, no credit will be given

Academic Integrity Violation

- Base sanction is typically a zero on the assignment **and** a letter grade reduction on final grade
 - Reduced grade is not dropped when dropping lowest scores on exercises/problem set
 - Student is typically not allowed to take the course P/F

Read More on Course Webpage

Academic Integrity Policies

Sources. It is okay to refer to external sources (lecture notes, slides, videos, AI generated content) if it helps you deepen your understanding of an algorithm or concept that was covered in the lectures. However, when it comes to solving specific exercises or homework problems, **you are not allowed to refer to any external resources or AI tools for guidance, only the textbook or resources provided by the instructor.**

- **If you unintentionally come across a solution to an exercise or homework problem, you must request an alternate problem.**

It is impossible to "unsee" a solution or approach to the problem once you read it. Seeing the solution beats the purpose of the assignment which is for you to practice the skills necessary to coming up with your solution. This often involves a bunch of false starts which are helpful in understanding the problem deeper. Seeing the correct approach first makes the problem seem easy in hindsight and hinders learning.

Collaboration. Most weekly problem sets are supposed to be done in pairs. Working with your peers is a great way to learn. You are encouraged to attempt all the problems together and in person with your partner. It is important to engage in a back-and-forth discussion and try to "break" each other's suggestions to fully understand if the proof or algorithm is correct. After attempting all problems, you may split up the writing responsibilities and submit the final joint write up.

Collaboration is not allowed on the daily exercises.

No collaboration is allowed on exams. The use of electronic devices like laptops and cellphones is prohibited during exams.

Academic Integrity. In general, you must adhere to the following academic integrity policies in the course.

- **You must always write your solution in your own words.** Verbatim duplication of any source, even with proper citation, is plagiarism.
- **You should never turn in a solution that you do not understand.** You may work with your partner on the problems, but you must personally understand and be able to explain each solution you turn in.
- **You may not upload or copy paste any part of the problem sets to any AI tools.**

If you are ever in doubt about a specific situation not covered above, you should write to me and obtain explicit permission.

What to Expect from this Class

- Practice with all stages of problem solving
 - Modeling the problem formally
 - Determining algorithmic strategies for solving them
 - **Verifying correctness: extremely important**
 - Analyze complexity: how good is the solution?
- Expect (and welcome) making mistakes
 - Making mistakes is the best way to learn
 - Learning is uncomfortable, but in a good way
 - Common and OK to be frustrated by false starts!

Practice with CS Proofs

- CS256 is meant to provide formal practice with proof writing
- CS proofs may have a slightly different style than math proofs
 - More "prose" form than long chain of math symbols
 - Will see lots of examples (in class, book and solved problems on the assignment)
- Programming labs vs proofs: common roadblock: how do I know that my proof is "correct"?
 - No universal autochecker for proofs!
 - Go line by line and ask "why is this true?"
 - Ask me or TAs for guidance
 - You'll build more intuition with practice

Course Outline

- Review of asymptotic complexity (Big Oh, Omega, Theta)
- Graph Traversals: BFS, DFS and applications
- Divide and conquer (recursion)
- Optimization:
 - Greedy
 - Dynamic programming (recursion without repetition)
 - Network-flow algorithms
- Intractability: NP hard, NP completeness & Reductions
- Advanced topics (if we have time)