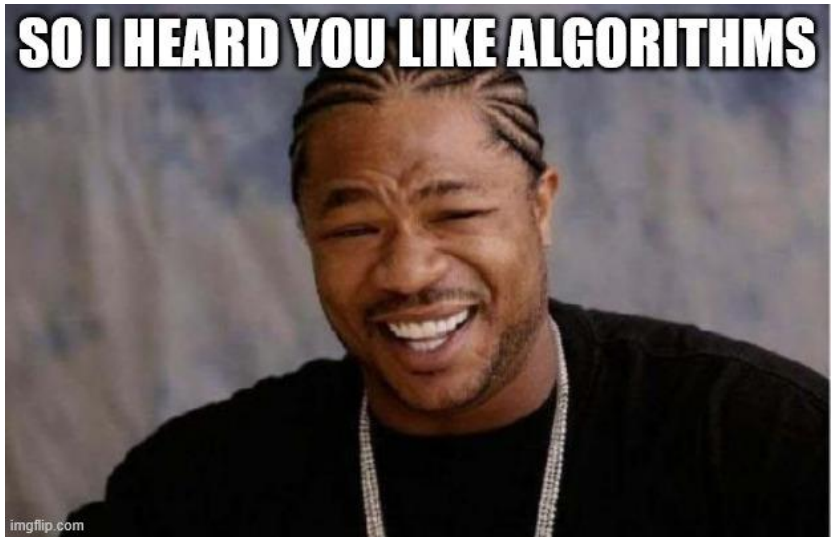
Advanced Algorithms

Welcome back to campus and to this class!



Plan for today:

- Algorithms review and Course Goals
- Structure of the course
- Exercise set

Who am I

- Prof. Michael Zlatin
- Ph.D. from Carnegie Mellon in Pittsburgh, Pennsylvania. Still getting used to this coast.
- I like all sports, currently volleyball and rock climbing.
- Etc. etc.

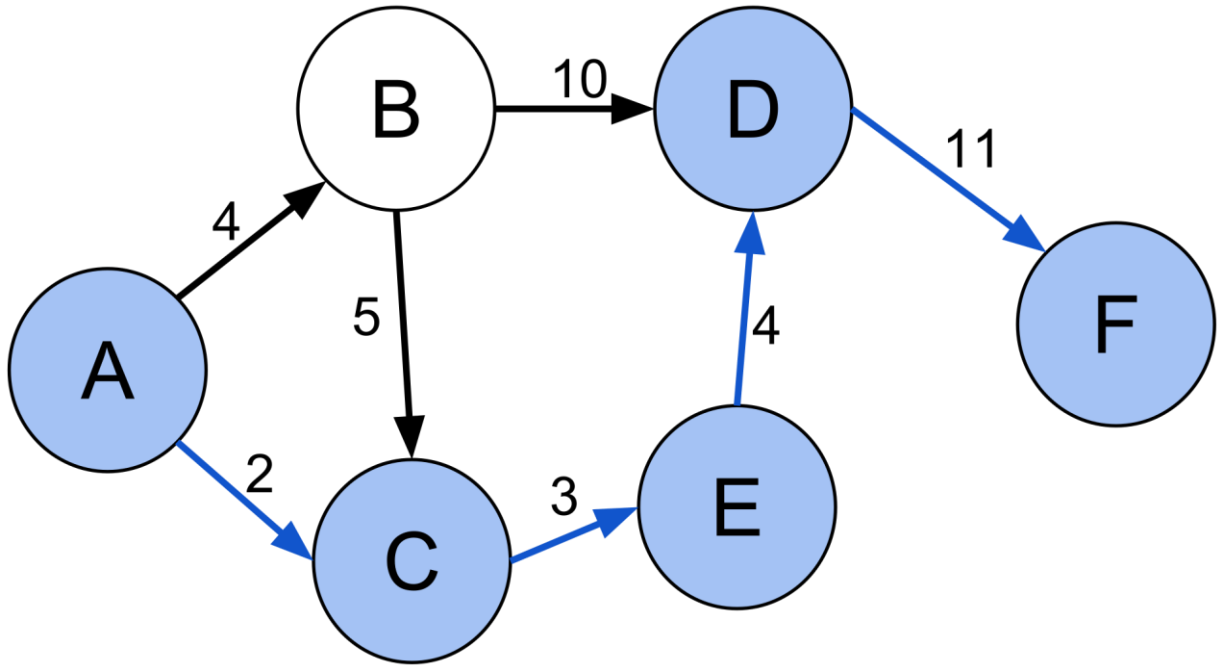
Hiking in Switzerland. Me (left), cow (right).

Advanced Algorithms

That’s a bit vague . . .

What is an algorithm?

An algorithm is an answer to a class of problems



What is the length of the shortest path from A to F in this directed graph?

Intro Algorithms

Problems:
- Sorting a list of numbers
- Graph traversal
- Shortest way to get from node a to node b in a graph
- Minimum spanning trees
- Flows?

Solutions:
- Bubble sort, merge sort
- Breadth First Search, Depth First Search
- Dijkstra's algorithm, Bellman-Ford, FW . . .
- Prim's, Kruskal's
- Ford-Fulkerson, Edmonds-Karp

Algorithmic Paradigms

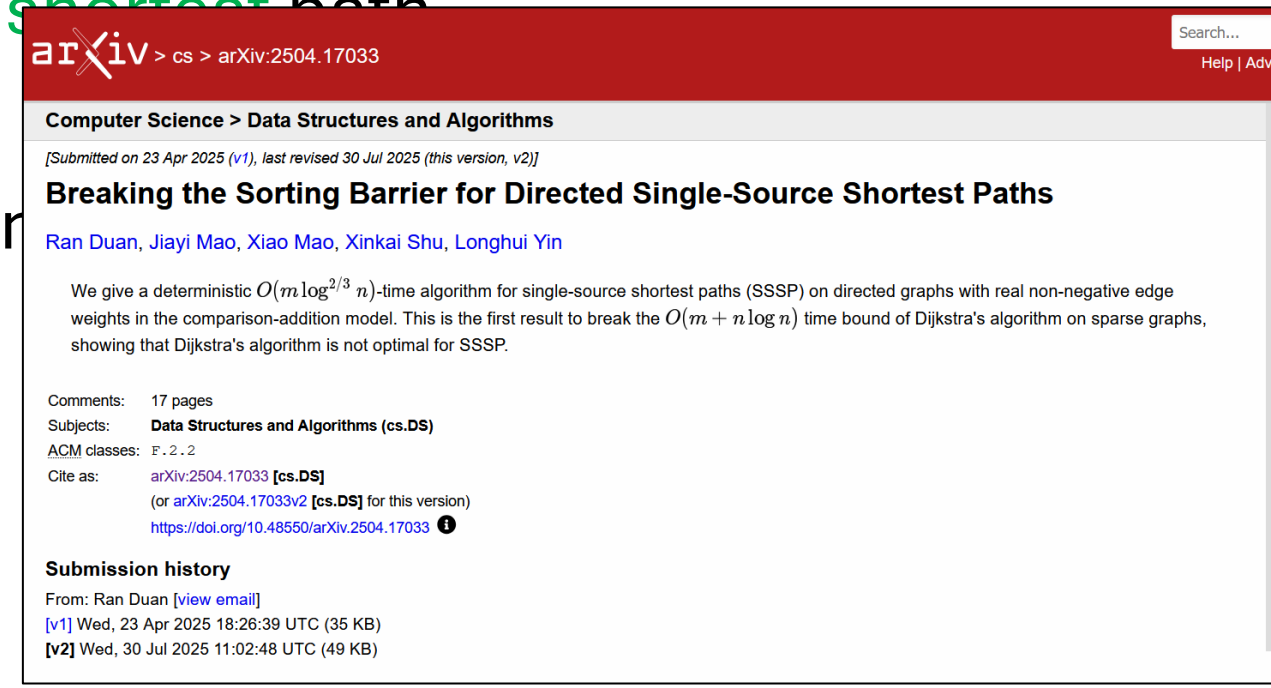
How do we compare algorithms?

Dijkstra's Algorithm

- **Feasibility:** always outputs a valid path from s to t in the graph
- **Optimality:** The path is always a shortest path
- **Running time:** on a graph with n r
  $O(m + n \log n)$ time.

Last summer: "improvement"
on Dijkstra's: $O(m \cdot \log^{2/3} n)$.

Best Paper Award at STOC 2025



Can we do it faster?

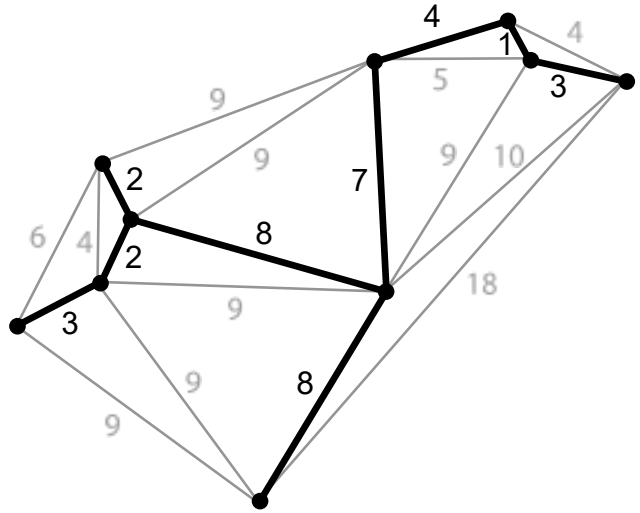
This is a good question

Faster algorithms for fundamental problems:

For example, MST:

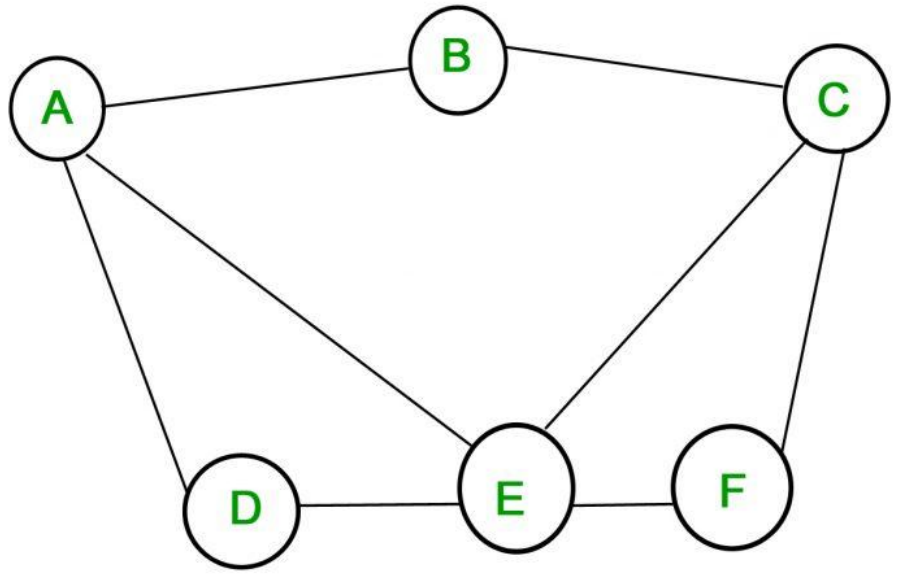
- Naïve brute force: 2^m
- Kruskal [1956]: $O(m \cdot \log m)$
- Chazzele [2000]: $O(m \cdot \alpha(m))$, where α is the inverse Ackerman function.

$$\alpha\left(2^{2^{2^{2^{16}}}}\right) \approx 4$$



Time complexity and input size

- Depends on how you store the graph.
- Graph on n nodes with m edges:
  - Adjacency matrix: $O(n^2)$
  - Adjacency list: $O(n + m)$
- We will always just assume it is given as an adjacency list.

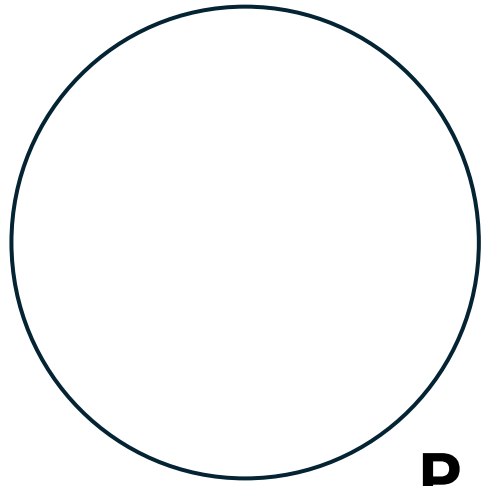


Q: Which problems will we be able to solve in practice?

A working definition: Those with poly-time algorithms

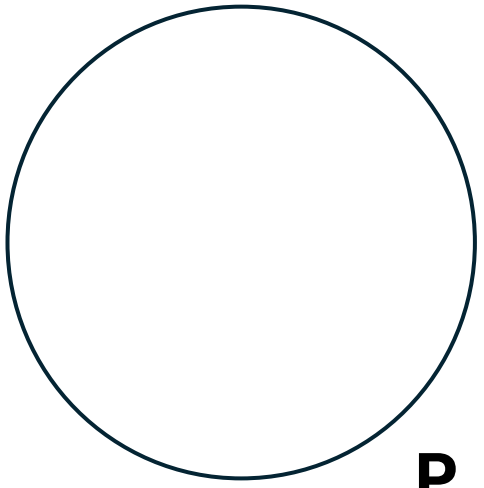
Theory. Definition is broad and robust.

Practice. Poly-time algorithms scale well on larger inputs



Q: Which problems admit polynomial time algorithms?

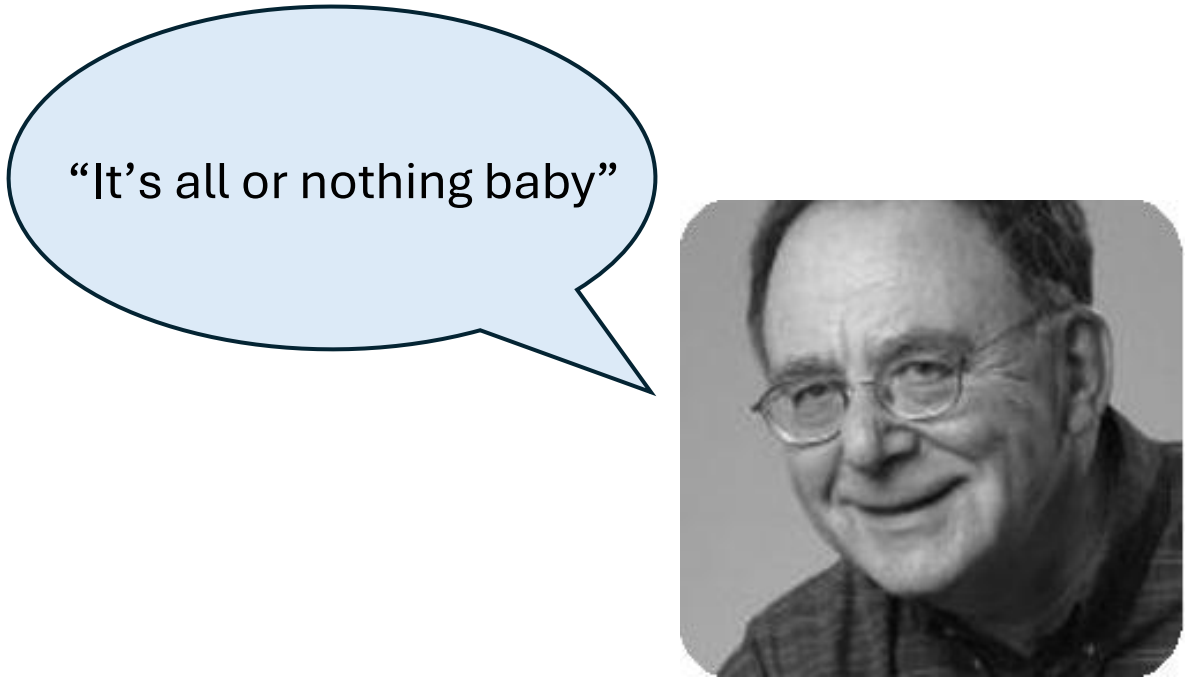
yes	probably no
shortest path	longest path
min cut	max cut
2-satisfiability	3-satisfiability
planar 4-colorability	planar 3-colorability
bipartite vertex cover	vertex cover
matching	3d-matching
primality testing	factoring
linear programming	integer linear programming



Can we do it faster?

Karp: Either all of these are in **P**, or none are:

- SAT
- 3-SAT
- Clique
- Independent Set
- Vertex Cover
- Hamiltonian Cycle
- Subset Sum
- 3D-matching
- Steiner Tree



Richard Karp

NP-hard: If there is a polynomial time algorithm for an NP-hard problem, then there is a poly time algorithm for all problems in NP.

Course Goals

Questions?

- Goal #1: What are the outer reaches of problems in **P**

- Flows, cuts, matchings, linear programming. More paradigms.
- **Utilize** this toolbox effectively and **communicate** solutions well

- Goal #2: How do we handle problems for which we believe an efficient algorithm does not exist?

- Fast algorithms for NP-hard problems: approximation and parameterization
- Problems where the input is not fully known: online and streaming

Logistics

- http://www.cs.pomona.edu/classes/cs181aa/

My office is Edmunds 223

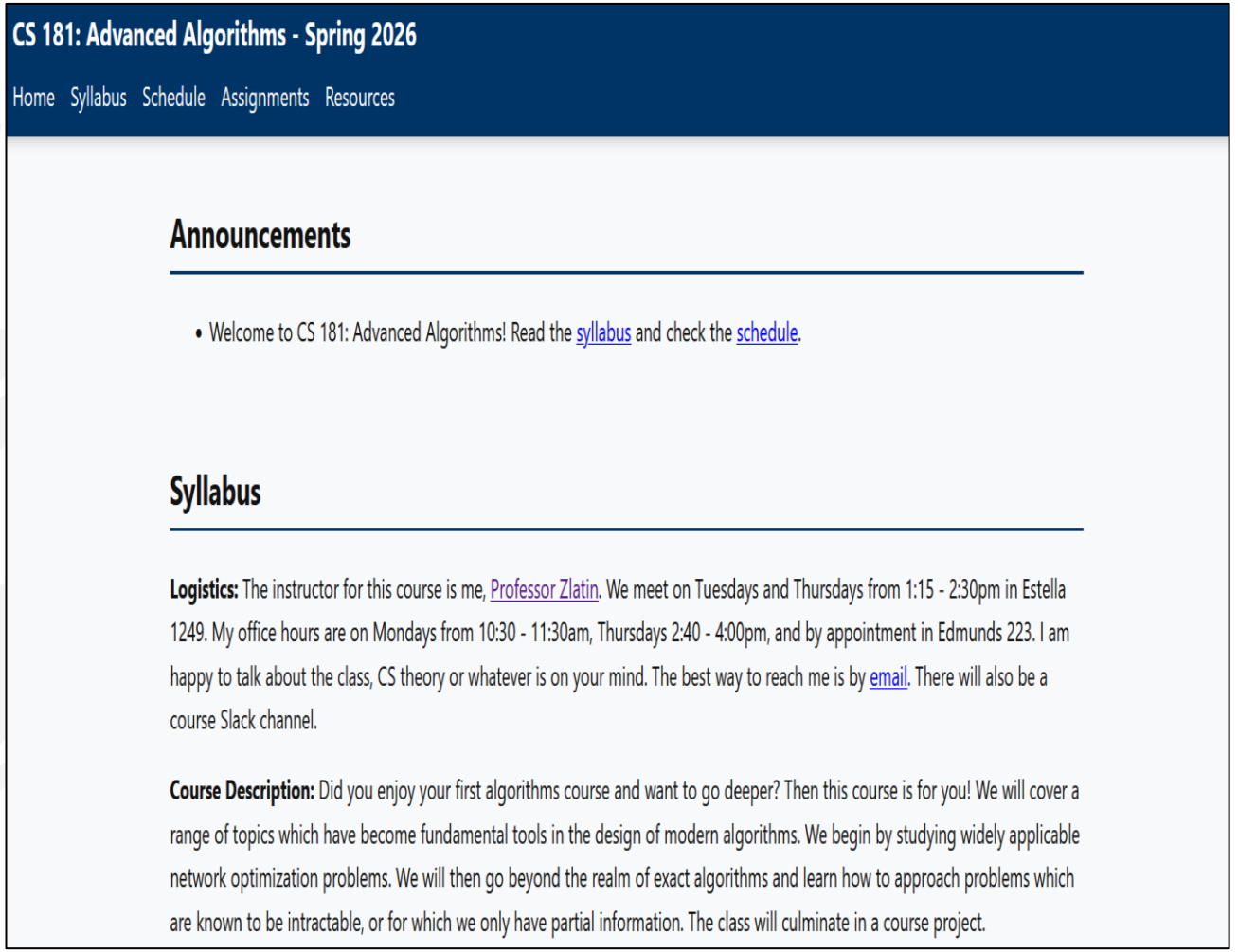
Office hours:

Monday 10:30 – 11:30am

Thursdays 2:40 – 4:00pm

By appointment

If I'm in my office, feel free to knock.



CS 181: Advanced Algorithms - Spring 2026

Home Syllabus Schedule Assignments Resources

Announcements

- Welcome to CS 181: Advanced Algorithms! Read the syllabus and check the schedule.

Syllabus

Logistics: The instructor for this course is me, Professor Zlatin. We meet on Tuesdays and Thursdays from 1:15 - 2:30pm in Estella 1249. My office hours are on Mondays from 10:30 - 11:30am, Thursdays 2:40 - 4:00pm, and by appointment in Edmunds 223. I am happy to talk about the class, CS theory or whatever is on your mind. The best way to reach me is by email. There will also be a course Slack channel.

Course Description: Did you enjoy your first algorithms course and want to go deeper? Then this course is for you! We will cover a range of topics which have become fundamental tools in the design of modern algorithms. We begin by studying widely applicable network optimization problems. We will then go beyond the realm of exact algorithms and learn how to approach problems which are known to be intractable, or for which we only have partial information. The class will culminate in a course project.

Workload

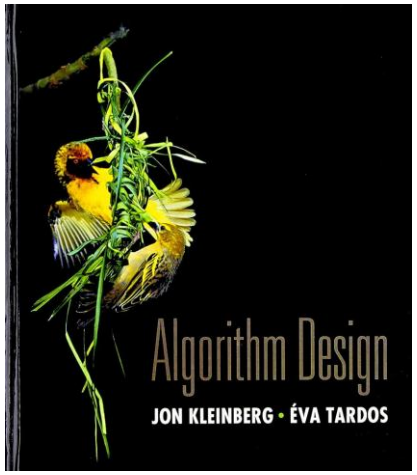
- Come to class and engage

- Exercise sets most weeks → due in class one week later

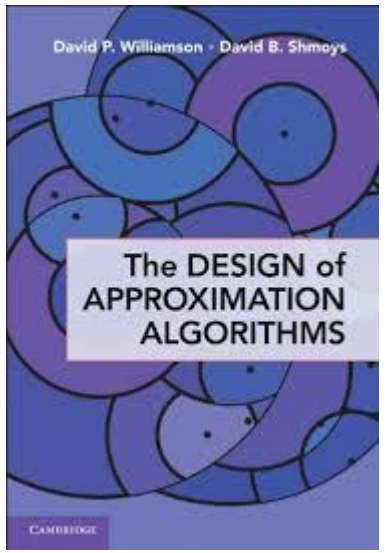
- Three / Four assignments:
  - Test the core skills I want **each student** to take away from this course
  - These are hard, collaboration is strongly encouraged
  - Write up solutions yourself
  - Due on gradescope

- Final Project

Resources



- Algorithm Design by Kleinberg and Tardos
  - → Especially useful for the first part of the course
  - → Copies in the Edmunds computer lab



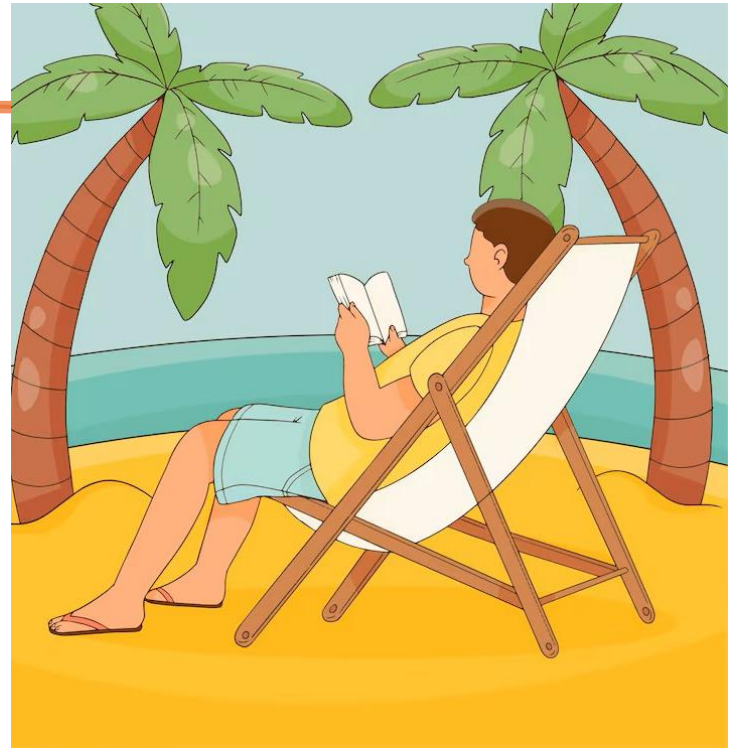
- A Second Course in Algorithms by Tim Roughgarden
  - → This is an online course
  - → Great notes, videos on youtube.

- Design of Approximation Algorithms by Williamson and Shmoys
  - → Best book on approx algos I know, copies in lab

AI policy

May use AI tools to:

- Ask questions, explain concepts
- Generate examples, quiz yourself
- Summarize lecture notes
- For pretty much for any reason you find useful EXCEPT
- You may not use AI tools for assistance on exercises / assignments.



- This not about evaluation, I genuinely believe that this is an important piece of the learning process
- Always feel free to come to me for help on assignments, I can give hints or guide your thought process

Slack

- Do people find this useful?
- Are you in it?

Complete the course survey!