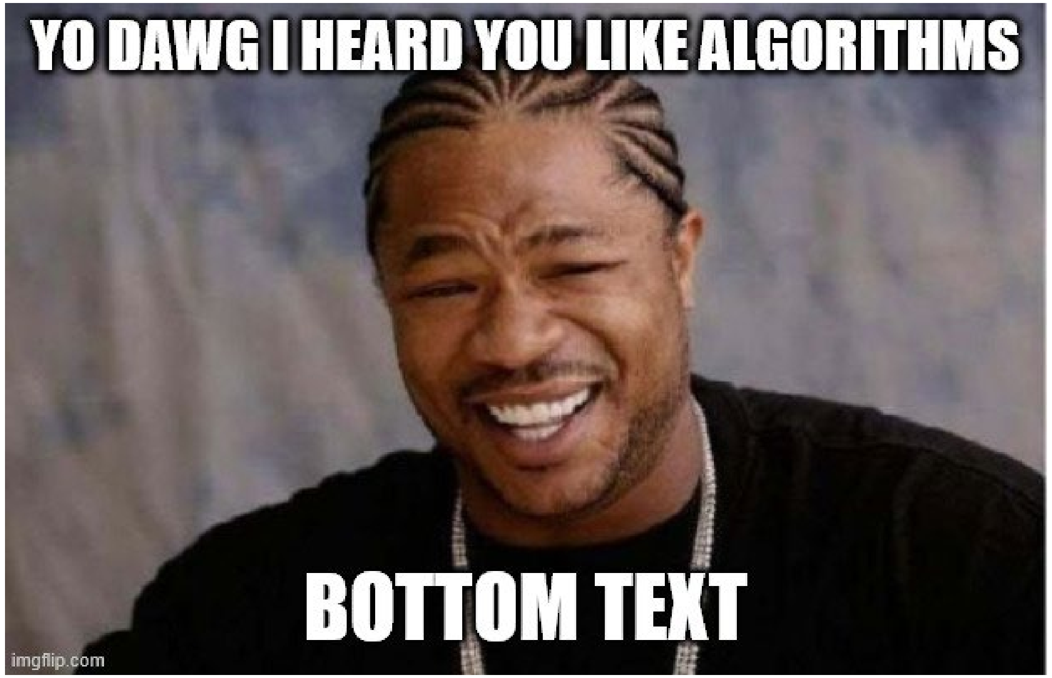
CS 181 – Advanced Algorithms

Welcome back to campus and to this class!



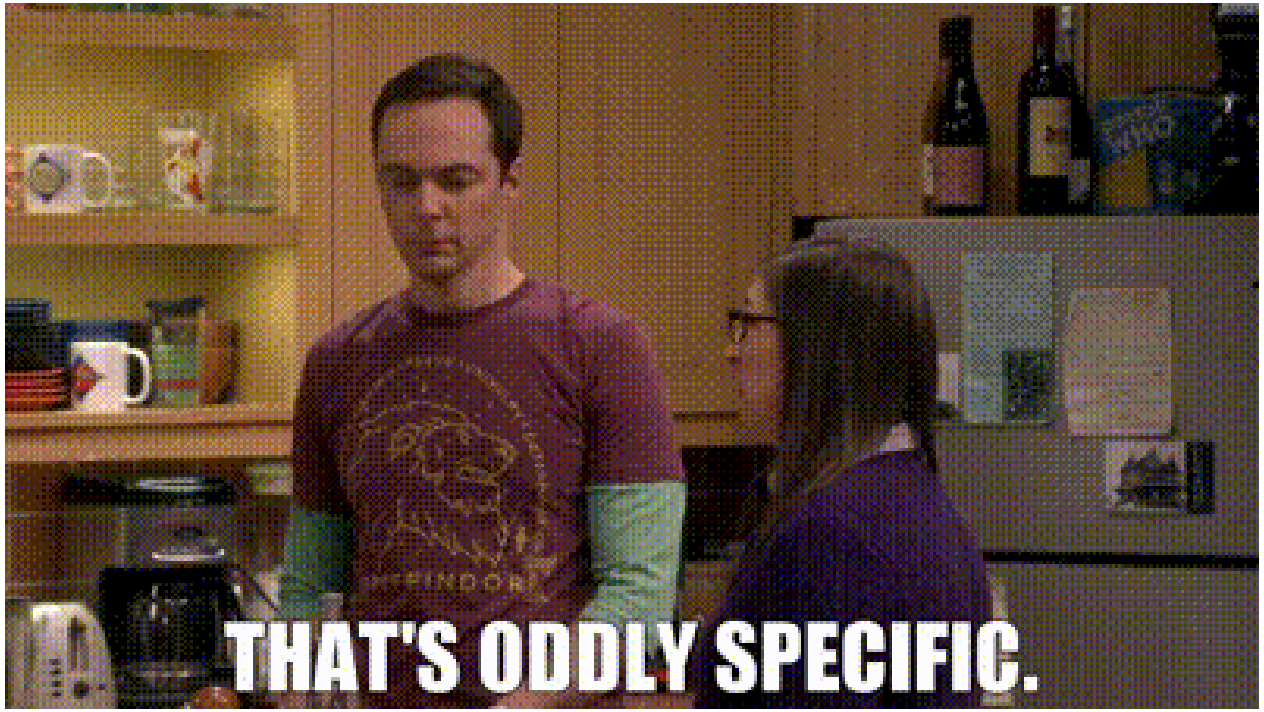
Who am I

- Prof. Michael Zlatin. I often go by Mik.
- 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).

Algorithms . . .

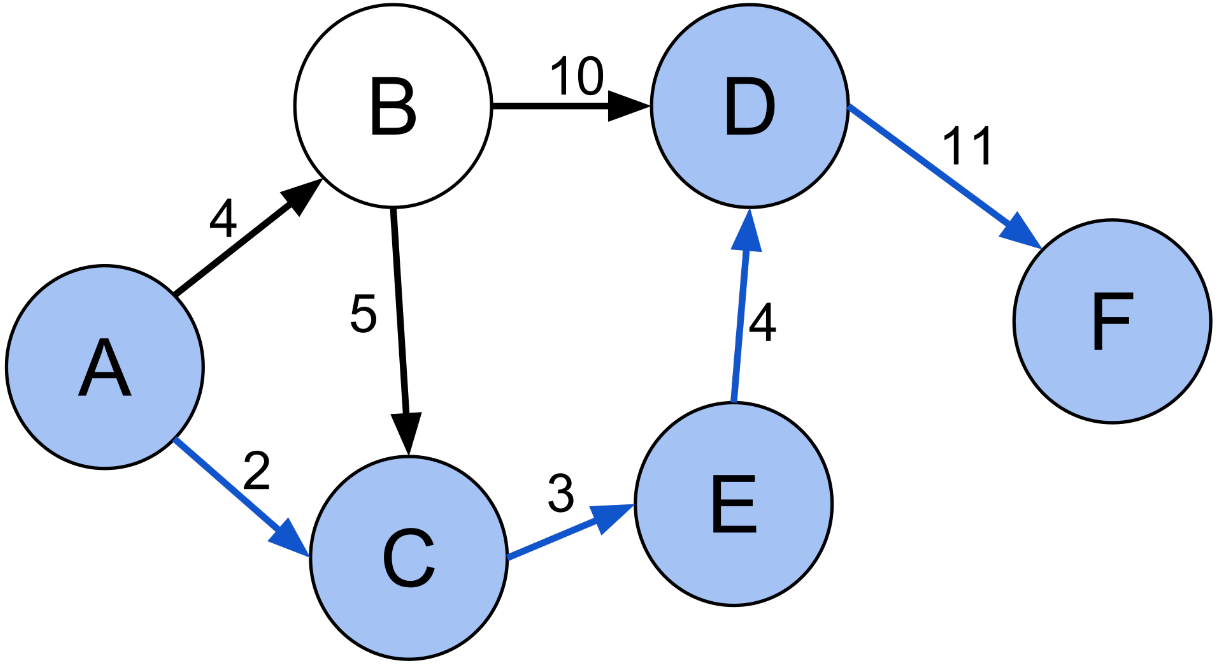
Problems:
- Sorting a list of numbers
- Graph traversal
- Shortest way to get from node a to node b in a graph
- Minimum spanning trees
- Flows?



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

An algorithm is an answer to a class of problems



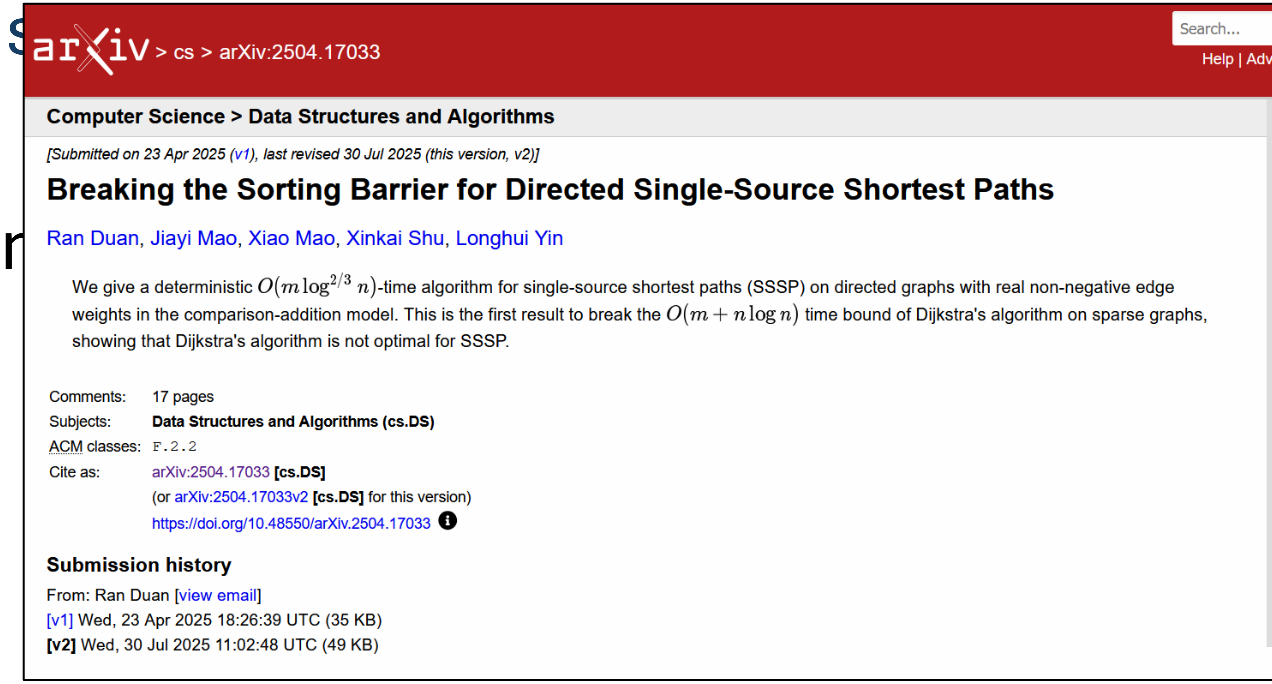
What is the length of the shortest path from A to F in this graph?

How do we compare algorithms?

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

This year: “improvement” on
Dijkstra’s: $O(m \cdot \log^{2/3} n)$.

Best Paper Award at STOC 2025


arXiv > cs > arXiv:2504.17033

Search...
Help | Adv

Computer Science > Data Structures and Algorithms

[Submitted on 23 Apr 2025 (v1), last revised 30 Jul 2025 (this version, v2)]

Breaking the Sorting Barrier for Directed Single-Source Shortest Paths

Ran Duan, Jiayi Mao, Xiao Mao, Xinkai Shu, Longhui Yin

We give a deterministic $O(m \log^{2/3} n)$-time algorithm for single-source shortest paths (SSSP) on directed graphs with real non-negative edge weights in the comparison-addition model. This is the first result to break the $O(m + n \log n)$ time bound of Dijkstra's algorithm on sparse graphs, showing that Dijkstra's algorithm is not optimal for SSSP.

Comments:	17 pages
Subjects:	**Data Structures and Algorithms (cs.DS)**
ACM classes:	F.2.2
Cite as:	arXiv:2504.17033 **[cs.DS]**
	(or arXiv:2504.17033v2 **[cs.DS]** for this version)
	https://doi.org/10.48550/arXiv.2504.17033

Submission history

From: Ran Duan [view email]
[v1] Wed, 23 Apr 2025 18:26:39 UTC (35 KB)
[v2] Wed, 30 Jul 2025 11:02:48 UTC (49 KB)

Can we do it faster?

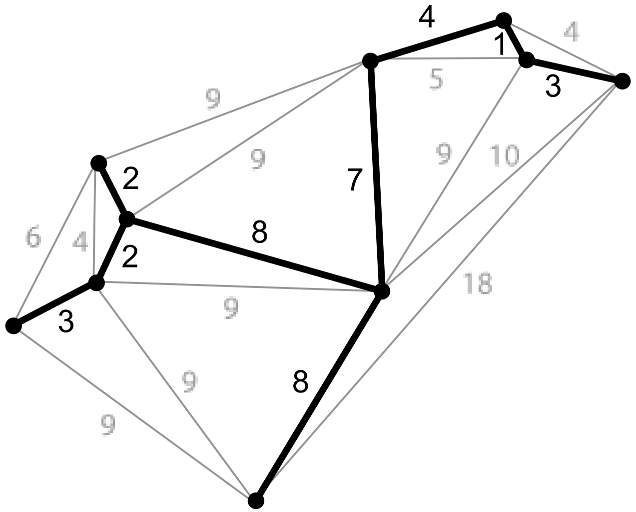
This is a good question

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.

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

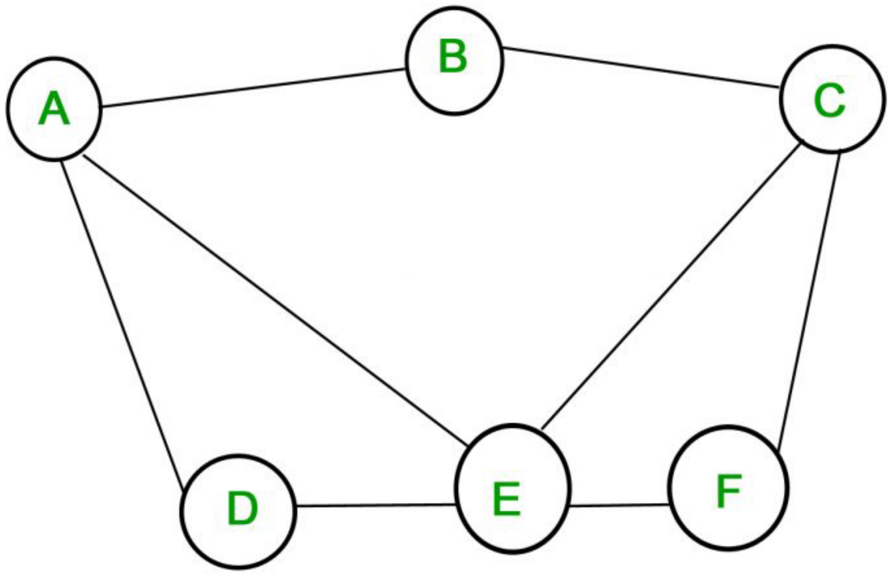


Time complexity and input size

- 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.

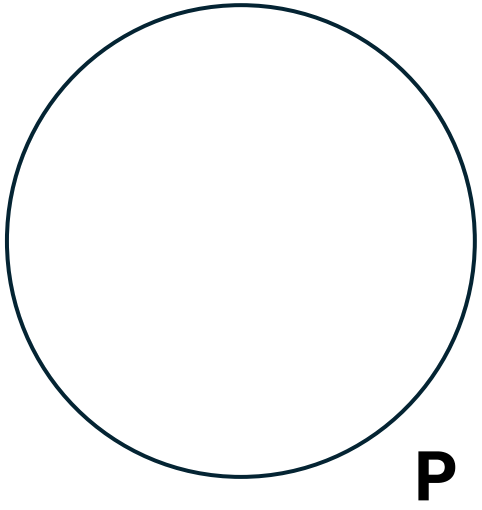


Q: Which problems will we be able to solve in practice?

A working definition: Those with poly-time algorithms

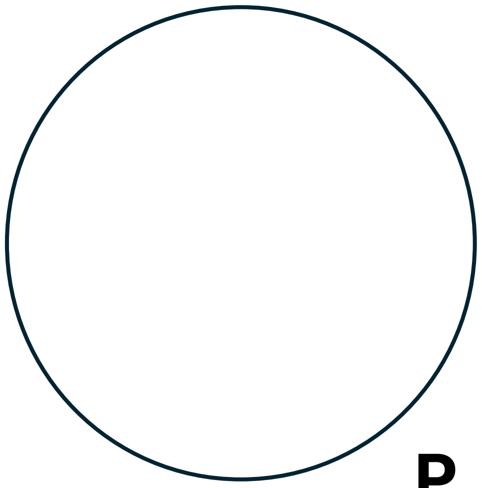
Theory. Definition is broad and robust.

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



P

Can we do it faster?

“It’s all or nothing baby”

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