CSE 331: Algorithms & Complexity

“Reductions”

Prof. Charlie Anne Carlson (She/Her)

Lecture34

Friday Nov 21st, 2025

University at Buffalo

Schedule

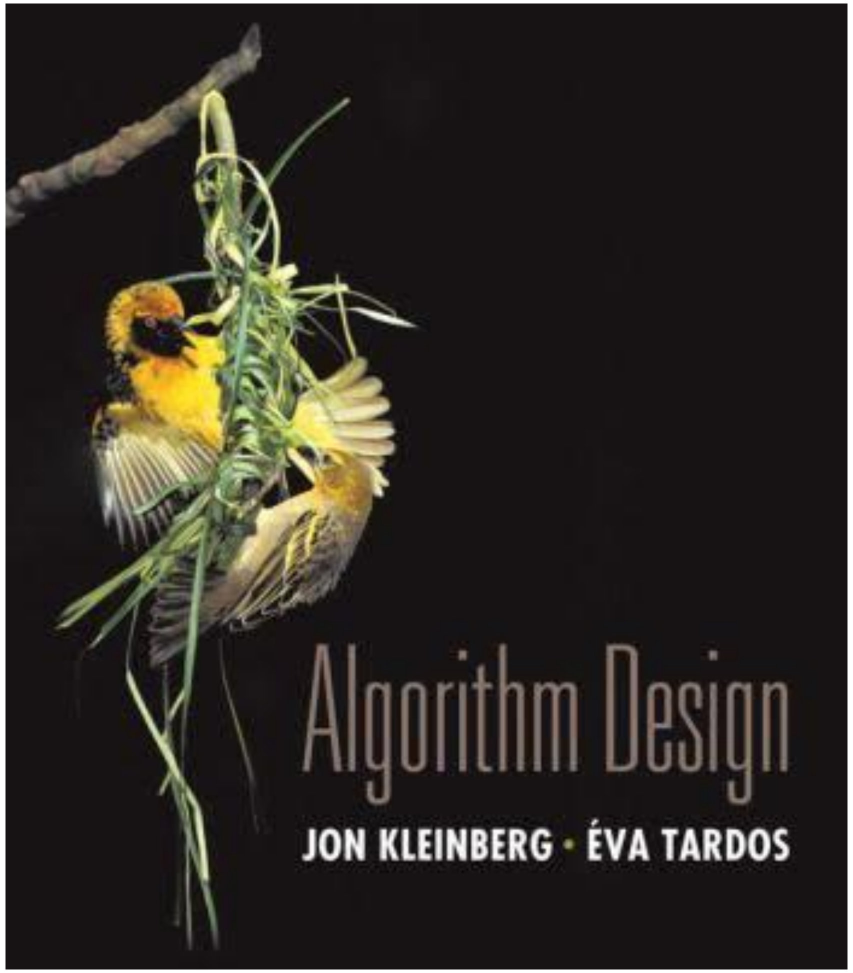
1. Course Updates
2. Recap
3. Hardness
4. Reductions

Course Updates

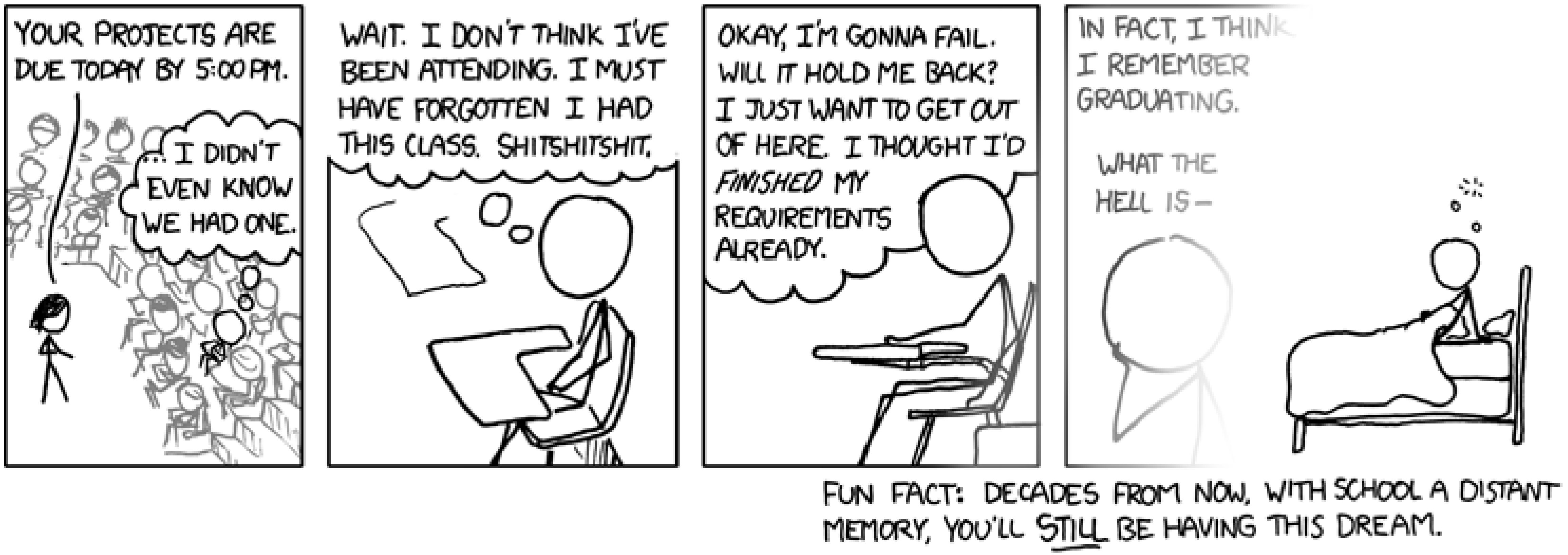
- HW 8 Out
    - Due December 2nd
- Group Project (Autolab Up)
    - Groups Fixed
    - Code 3 Due November 24th
    - Reflections 3 Due December 1st
- Next Quiz is December 1st
- Final on December 10th (Info Next Week)

Reading

- You should have read:
  - Started 8.1
- Before Next Class:
  - Finish 8.1
  - Start 8.2, 8.3, & 8.4



Schedule


YOUR PROJECTS ARE DUE TODAY BY 5:00 PM.
...I DIDN'T EVEN KNOW WE HAD ONE.
WAIT. I DON'T THINK I'VE BEEN ATTENDING. I MUST HAVE FORGOTTEN I HAD THIS CLASS. SHITSHITSHIT.
OKAY, I'M GONNA FAIL. WILL IT HOLD ME BACK? I JUST WANT TO GET OUT OF HERE. I THOUGHT I'D *FINISHED* MY REQUIREMENTS ALREADY.
IN FACT, I THINK I REMEMBER GRADUATING.
WHAT THE HELL IS –
FUN FACT: DECADES FROM NOW, WITH SCHOOL A DISTANT MEMORY, YOU'LL STILL BE HAVING THIS DREAM.

https://xkcd.com/557/

Schedule

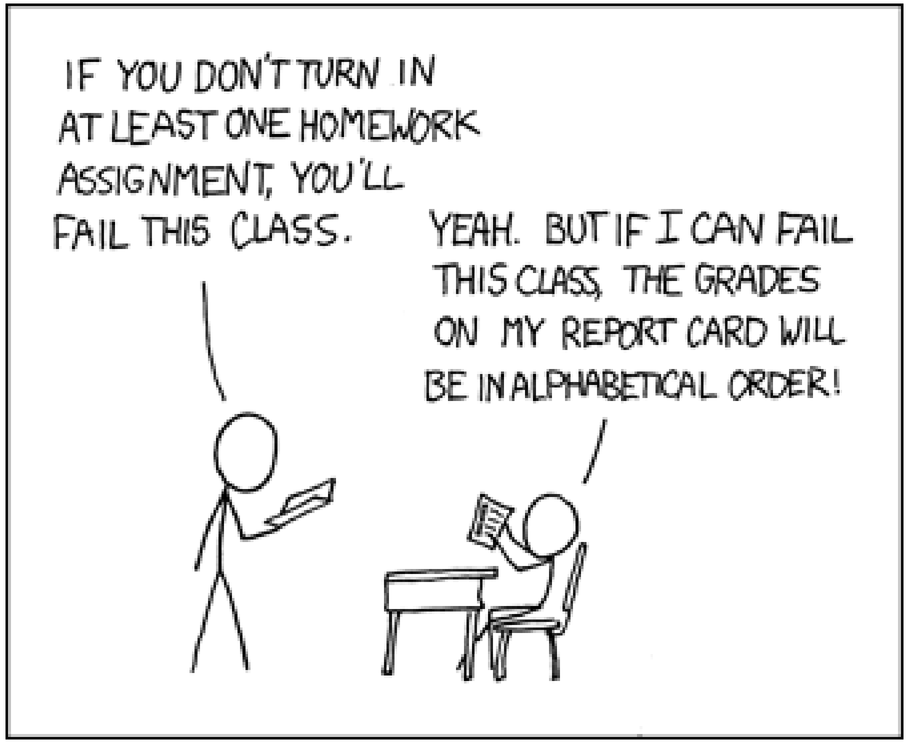
- Monday Nov. 24th : P vs NP
- Wednesday Nov. 26th : No Class
- Friday Nov. 28th : No Class
- Monday Dec. 1st : NP-Completeness
- Wednesday Dec. 3rd : Satisfiability
- Friday Dec. 5th : COLORING!!!!!!!
- Monday Dec. 8th : Wrapup

Deadlines

- Monday Nov. 24th :
  - Project Code Problem 3
- Monday Dec. 1st :
  - Project Reflection Problem 3
  - Quiz 2 (in class)
- Tuesday Dec. 2nd :
  - HW 8
- Friday Dec. 5th :
  - Project Code Problem 4 & 5
- Tuesday Dec. 9th :
  - Project Reflection Problem 4 & 5
  - Project Survey
- Wednesday Dec. 10th :
  - Final Exam

Feedback

- Monday Nov. 24th :
  - HW 6 Graded
  - HW 6 & HW 7 Solutions Posted
  - Reflection Problem 2 Graded
  - Practice Final & Practice Quiz 2
- Monday Dec. 1st :
  - HW 7 Graded
  - Project Reflection Problem 1 Graded
- Tuesday Dec. 2nd :
  - HW 8 Solutions Posted
- Monday Dec. 10th :
  - HW 8 Graded
  - Reflection Problem 3 Graded


IF YOU DON'T TURN IN AT LEAST ONE HOMEWORK ASSIGNMENT, YOU'LL FAIL THIS CLASS.
YEAH. BUT IF I CAN FAIL THIS CLASS, THE GRADES ON MY REPORT CARD WILL BE IN ALPHABETICAL ORDER!

https://xkcd.com/336/

A Few Notes

- Please ping me if you are still waiting for me to respond to an email. I think I caught up on most of them but if you have fallen through the cracks, let’s get it settled ASAP.
- If you are going to miss the final exam (for an acceptable reason) let me know ASAP via email.
- You will be given assigned seating for the final and must plan to bring your school ID.

Course Overview

- Stable Matchings
- Analysis
- Graphs
- Greedy Algorithms
- Divide and Conquer
- Dynamics Programming
- **NP and Computational Intractability**

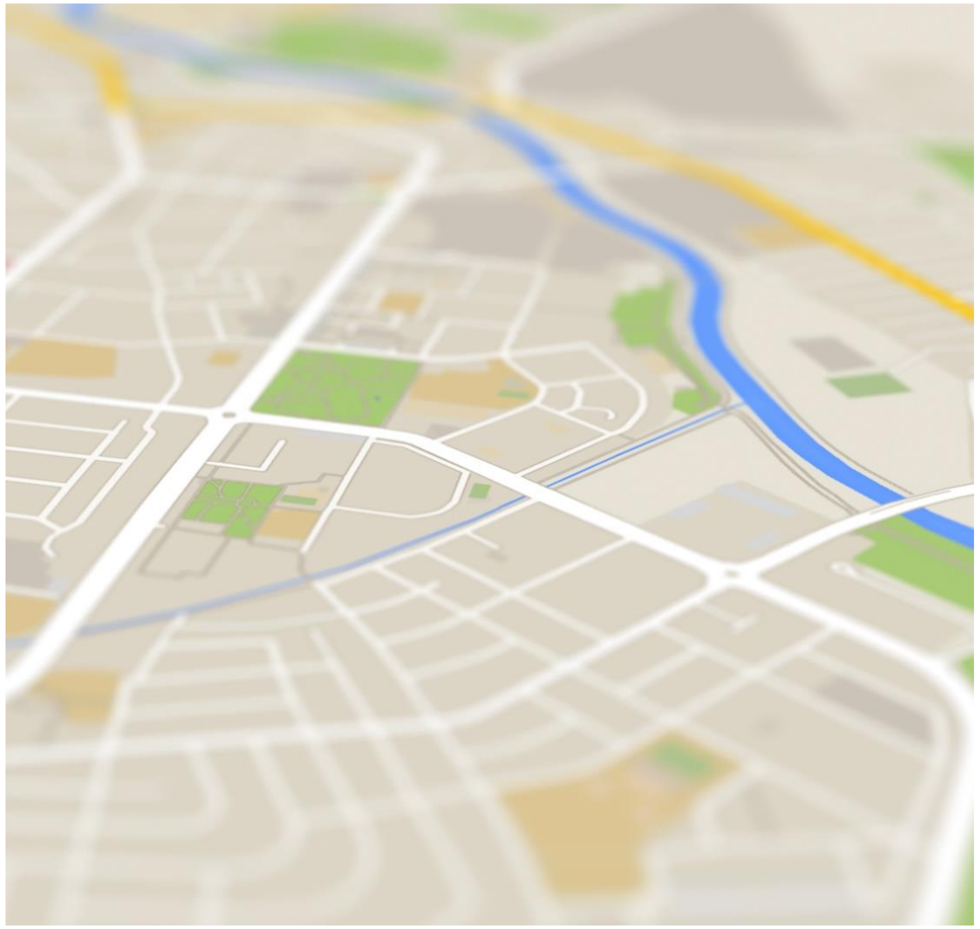
What can be done!

What can't be done!

Shortest Path

- **Input**:
  - A directed graph $G = (V, E)$.
  - A weight for each edge $(i, j) \in E$, c_{ij}.
  - Source vertex $s \in V$.
  - Destination vertex $t \in V$.
  - **No negative cycles!**
- **Goal**: Find a path from s to t with minimum total cost:

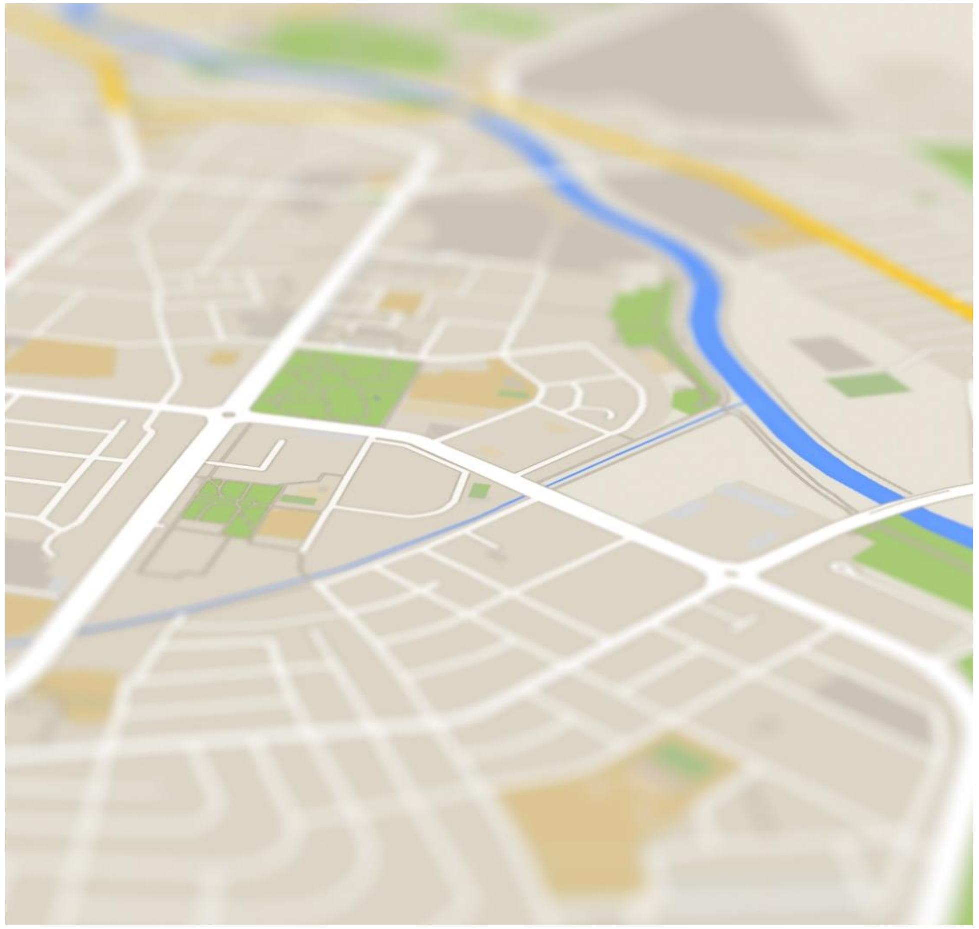
$$\mathrm{cost}(P) = \sum_{ij \in P} c_{ij}$$



Longest Path

- **Input**:
  - A directed graph $G = (V, E)$.
  - A weight for each edge $(i, j) \in E$, c_{ij}.
  - Source vertex $s \in V$.
  - Destination vertex $t \in V$.
- **Goal**: Find a **simple** path from s to t with **maximum** total cost:

$$\text{cost}(P) = \sum_{ij \in P} c_{ij}$$



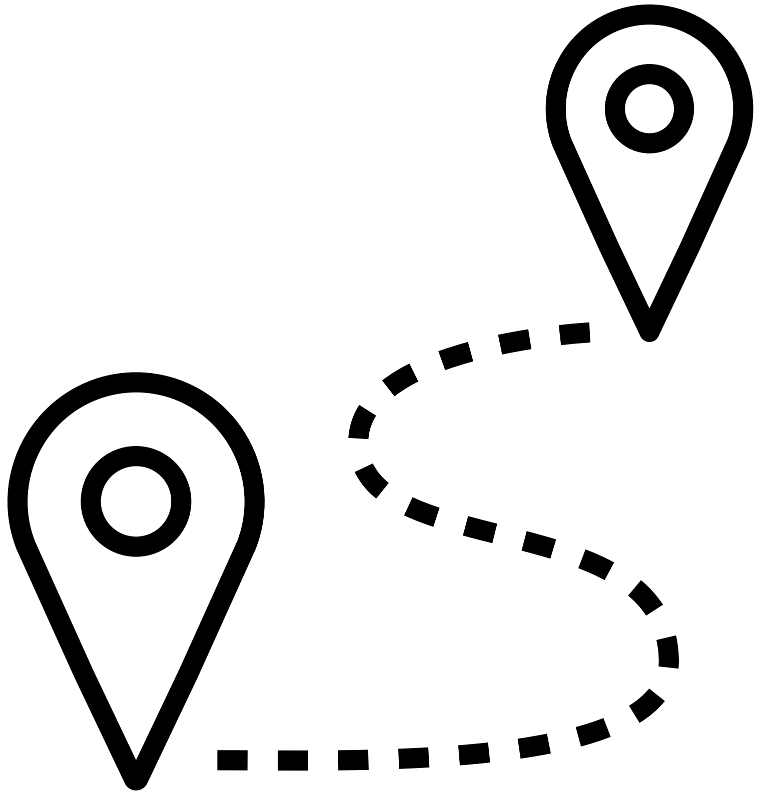
How hard could it be?

How hard could it be?

- We have a polynomial time algorithm for the shortest path problem.
    - BFS, Dijkstra’s, Bellman-Ford
- We “believe” there is no polynomial time algorithm for the longest path problem.
    - We say “believe” because we don’t know exactly but more on that soon…

Decision vs Optimization Problem

- We have seen a lot of optimization problems in this class that ask for things like shortest, longest, or maximum weight objects.
    - An optimization problem asks to find the "best" object in a set.

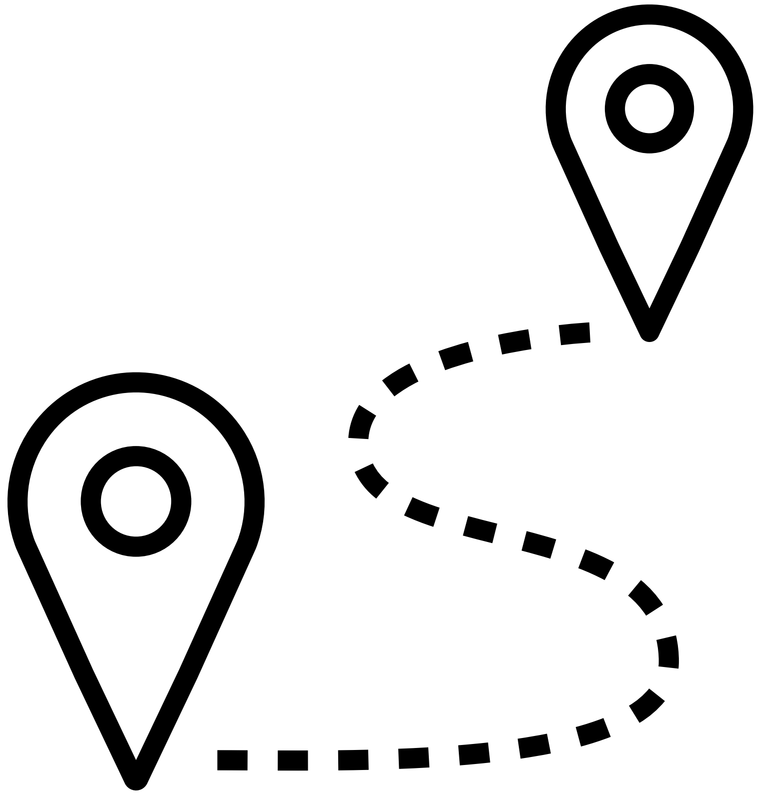


Decision vs Optimization Problem

- We have seen a lot of optimization problems in this class that ask for things like shortest, longest, or maximum weight objects.
    - An optimization problem asks to find the "best" object in a set.
- We haven't talked about decision problems as much which ask if there exist an object with specific property.

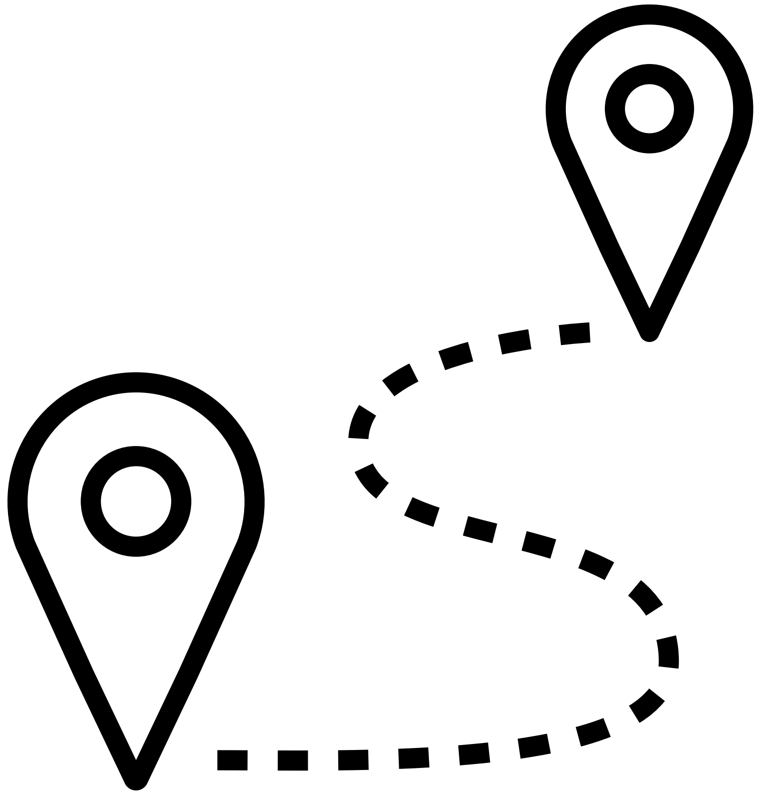
Decision vs Optimization Problem

- Shortest Path Optimization Problem:
  - What is the shortest path between s and t?
- Shortest Path Decision Problem:
  - Given a weight W, does there exist a path of length at most W between s and t?
- **Question**: If you can solve one, can you solve the other?



Decision vs Optimization Problem

- Shortest Path Optimization Problem:
  - What is the shortest path between s and t?
- Shortest Path Decision Problem:
  - Given a weight W, does there exist a path of length at most W between s and t?
- **Question**: If you can solve one, can you solve the other?
  - **Answer**: Yes, check if shortest path is shortest or do a search.



Witness or Certificate

- **Question**: How would you prove that there is a shortest path of length at most W?

Witness or Certificate

- We can **witness** or **certify** the answer to a decision problem by demonstrating the object provided one can check that it has the correct property efficiently.
  - Path of length at most W.
  - Path of length at least W.
  - Schedule with at least k jobs.
  - Points that are at most δ away from each other.

P and NP

- **P**: Decision problems for which there exist a polynomial time algorithm that solves it.
- **NP**: Decision problems for which there is always a polynomial time verifiable certificate for the solution.

P ? NP

What?

- **P**: “I can find the solution efficiently.”
- **NP**: “I can verify the solution efficiently.”

P ? NP

NP = Nondeterministic Polynomial time

- **Question:** If you can verify the solution quickly, how can you find the solution? How quick is it?

P ? NP

NP = Nondeterministic Polynomial time

- **Question:** If you can verify the solution quickly, how can you find the solution? How quick is it?
- **Answer**: I can guess/enumerate all possible solutions and efficiently check them to see if they are solutions. It is slow....

P ? NP

P vs NP

- Both P and NP are sets of problems (languages).
- We know that P is a subset of NP.
- We don’t know if P = NP and if you solve this, you get money, fame, and respect!

P vs NP

P vs NP

- People have been working on P vs NP for a long time.
- While some may be less sure that P != NP, we can mostly all agree that even if it is, there are problems in NP that are going to be very “hard.”
- **Question**: What might it help to know a problem is ”hard”?

P vs NP

P vs NP

- **Question**: What might it help to know a problem is ”hard”?
  - **Answer**: We don’t want to waste time trying to solve problems that we know are very hard to solve.

WHEN A USER TAKES A PHOTO, THE APP SHOULD CHECK WHETHER THEY'RE IN A NATIONAL PARK...
SURE, EASY GIS LOOKUP. GIMME A FEW HOURS.
...AND CHECK WHETHER THE PHOTO IS OF A BIRD.
I'LL NEED A RESEARCH TEAM AND FIVE YEARS.
IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

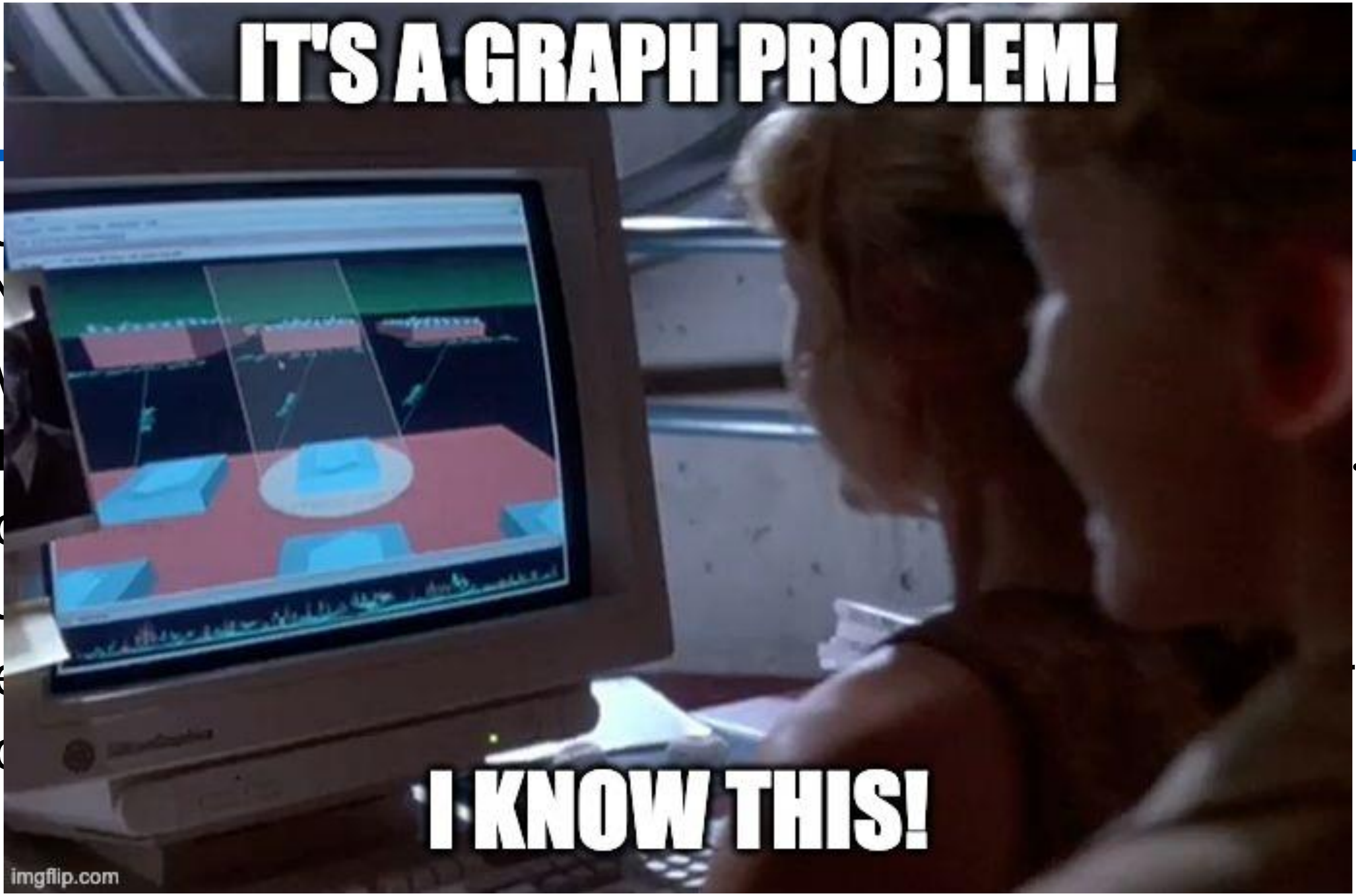
https://xkcd.com/1425/

Reductions

- You are given a network of computers to manage.
  - You are given a list of computers along with a list of which computer are directly connected to each.
  - Each connection has a cost for sending data over it.
- Each computer has a unique archive of information.
- Your boss wants you to figure out a way to find the cheapest way to send information from one computer to another upon request.

Redu

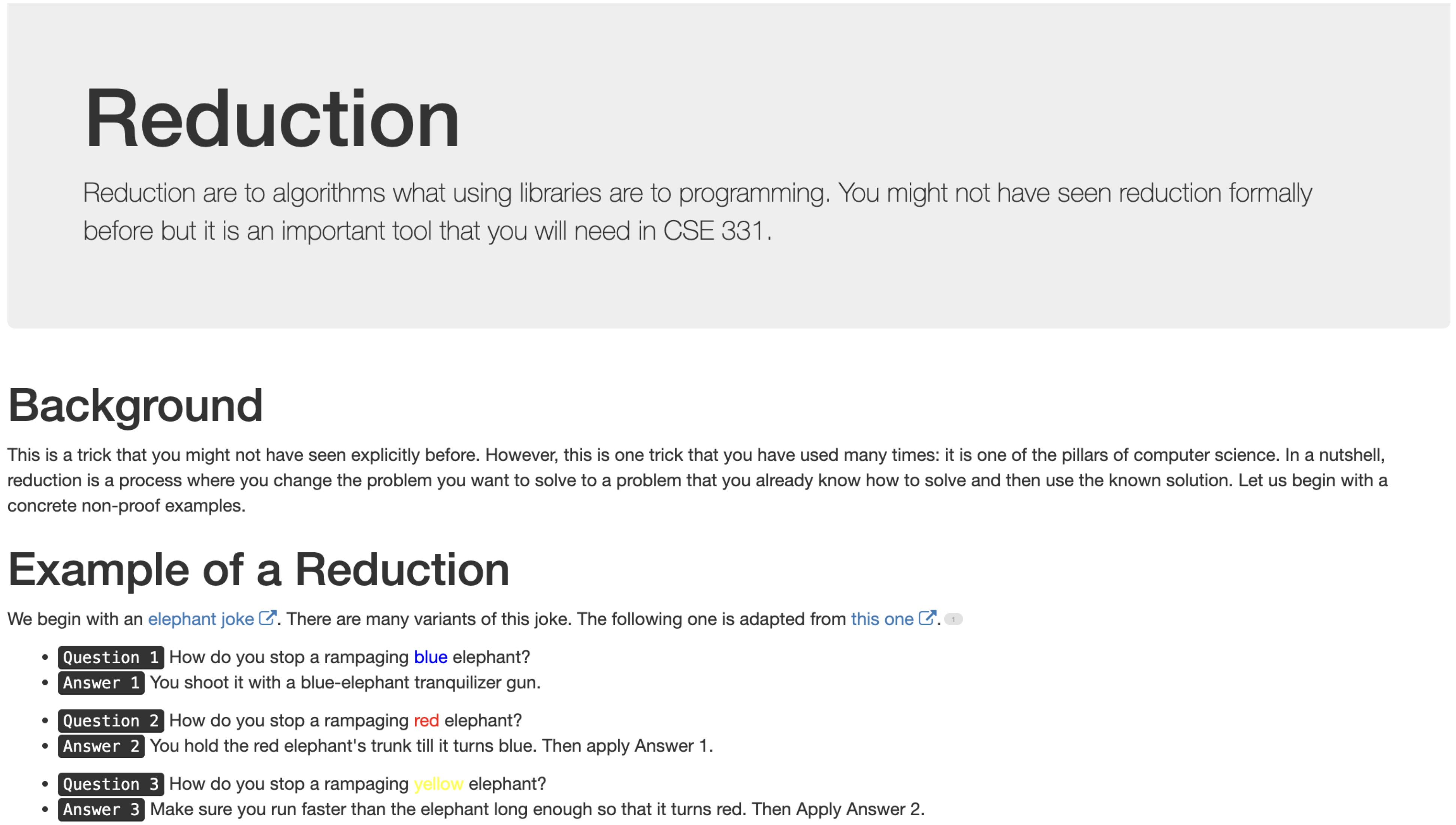
- You
  - Y
    w
  - I
    .
- Eac
- You
  che to
  and



Reductions

- You can easily map this to a graph problem.
  - Each computer is a node in a graph where each direct connection is an edge.
  - The cost of sending data across connection is a weight for the edge.
  - Then finding the cheapest way to route data is just the shortest path problem.
    - I know an algorithm for shortest path in this case!

Reductions



Reduction

Reduction are to algorithms what using libraries are to programming. You might not have seen reduction formally before but it is an important tool that you will need in CSE 331.

Background

This is a trick that you might not have seen explicitly before. However, this is one trick that you have used many times: it is one of the pillars of computer science. In a nutshell, reduction is a process where you change the problem you want to solve to a problem that you already know how to solve and then use the known solution. Let us begin with a concrete non-proof examples.

Example of a Reduction

We begin with an elephant joke. There are many variants of this joke. The following one is adapted from this one. 1

- `Question 1` How do you stop a rampaging blue elephant?
- `Answer 1` You shoot it with a blue-elephant tranquilizer gun.

- `Question 2` How do you stop a rampaging red elephant?
- `Answer 2` You hold the red elephant's trunk till it turns blue. Then apply Answer 1.

- `Question 3` How do you stop a rampaging yellow elephant?
- `Answer 3` Make sure you run faster than the elephant long enough so that it turns red. Then Apply Answer 2.

Reasons to use Reduction

- You want to show something is easy/possible:
  - You have a real-world problem, and you want to reduce it to a simple graph problem you know how to solve.
  - You have a new graph problem that seems harder but isn’t.
  - You may also want to show how to use already existing algorithms for none graph problems.

Reasons to use Reduction

- You want to show something is hard/impossible:
  - Reductions give you a formal way to show that a problem may not be possible to solve.
  - We may not have a proof that a problem is actually hard, but we do know that people have been trying for a long time to solve it with no success.
    - If you have a new problem, you don’t want to have to fail to solve it for years to justify not being able to solve it. Instead, you reduce.

Oracles

- Consider a problem X
  - E.g. Shortest path, longest path, stable matching, coloring, sorting, etc.
- Note that X has some input and desired output
  - E.g. Graph + Weights => Shortest Path
- An oracle is a black box that can solve X on any input.

Reduction

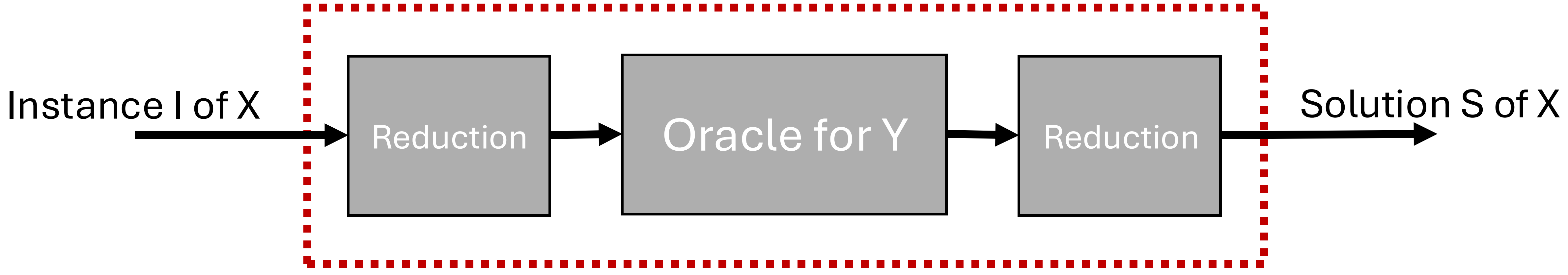
- Consider problem Y that we assume can be solved in poly time.
- **Question**: How can we use this assumption about Y to show that another problem is easy?

Reduction

- Problem X polynomial-time (cook) reduces to problem Y if arbitrary instances of problem X can be solved using:
  - Polynomial number of computational steps, plus
  - Polynomial number of calls to an oracle that solves problem Y.

Reduction

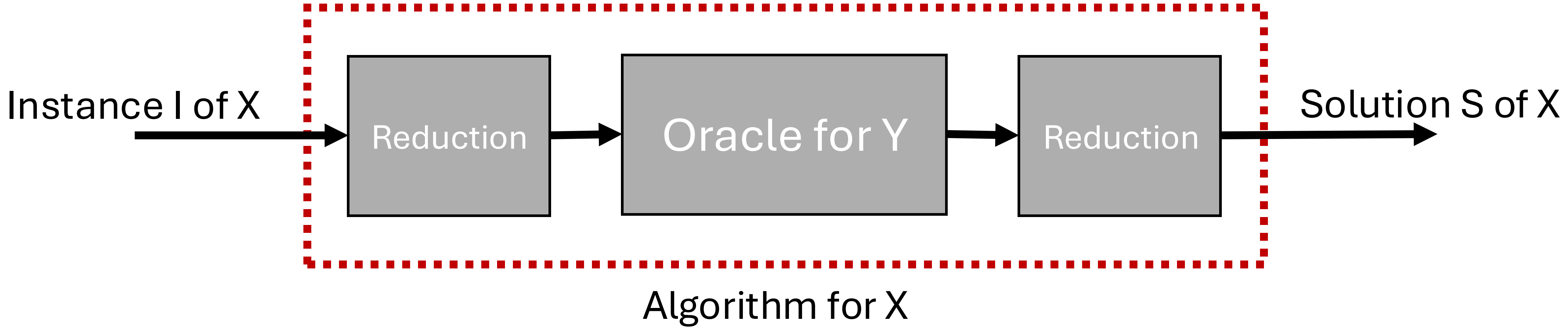
- Problem X polynomial-time (cook) reduces to problem Y if arbitrary instances of problem X can be solved using:
  - Polynomial number of computational steps, plus
  - Polynomial number of calls to an oracle that solves problem Y.



Algorithm for X

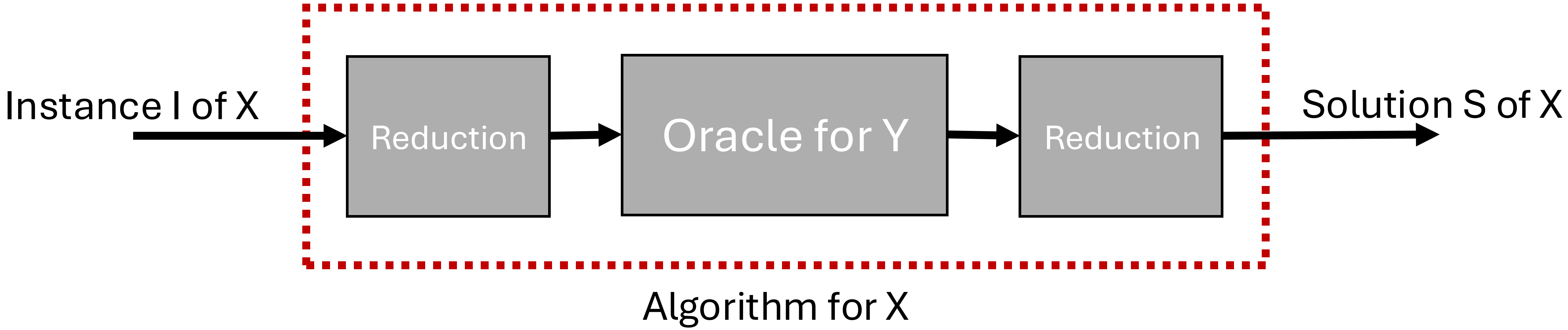
Reduction

- We want both parts of the reduction to run in polynomial time.
    - First part converts instance I of X to an instance J of Y.
    - Second part converts solution T of Y to a solution S of X.



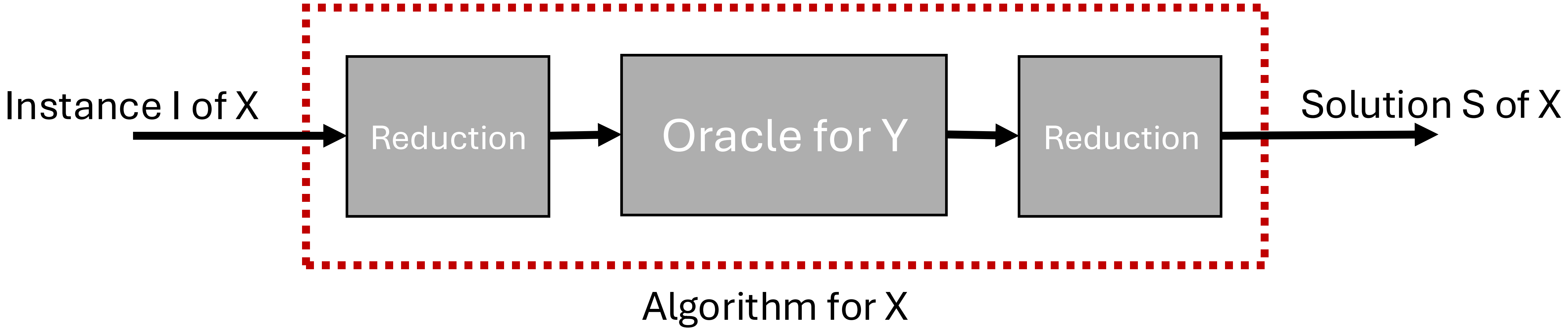
Reduction ($X \leq_p Y$)

- If we can prove that both parts of the reduction run in polynomial time, then if there is a polynomial time algorithm for Y there is a polynomial time algorithm for X.
- "Problem X is at most as hard as Y with respect to poly-time."



Reduction ($X \leq_p Y$)

- If we know there is no efficient algorithm for X, then it follows that there can't be an efficient algorithm for Y.
- "Problem X is at most as hard as Y with respect to poly-time."


Instance I of X
Reduction
Oracle for Y
Reduction
Solution S of X
Algorithm for X