CSE 421 Section 3

Problem solving with greedy algorithms

Administrivia

Announcements & Reminders

- **HW1**
  - Regrade requests are open
  - Answer keys available on Ed

- **HW2**
  - Was due yesterday, 10/9
  - Remember the **late problems** policy (NOT assignments)
    - Total of up to **10 late problem days**
    - At most **2 late days per problem**

- **HW3**
  - Due Wednesday 10/16 @ 11:59pm

How to write an algorithm

Problem solving strategy overview

Read and **summarize** the problem

Decide to use **known algorithm** or **techniques from scratch**

not covered this section

no idea

have idea

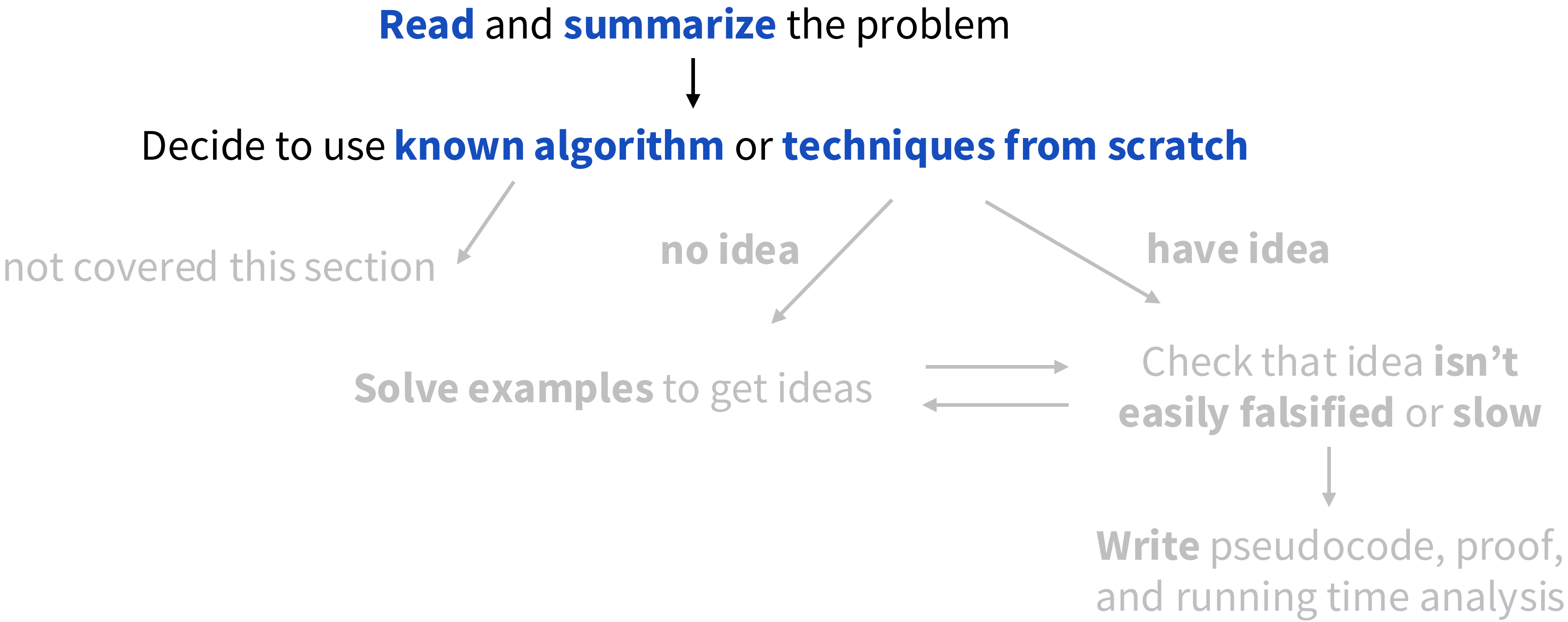
Solve examples to get ideas

Check that idea **isn't easily falsified** or **slow**

Write pseudocode, proof, and running time analysis

Getting started

Getting started



Problem summary

When reading a long word problem, it is useful to **summarize** it. A common way is:

Input: …
Expected output: …

- mathematical definitions of any special words used above

Problem summary

When reading a long word problem, it is useful to **summarize** it. A common way is:

Example

Input: Two sets P and R of n people each, with preference lists

Expected output: A stable matching

- **preference list:** an ordered list of people in the other set
- **stable matching:** a perfect matching for which there is no (p, r) where p and r prefer each other over their current match

Problem 1 – Line covering

Your new towing company wants to be prepared to help along the highway during the next snowstorm. You have a list of integers $t_1, t_2, \ldots, t_n$ in increasing order, representing mile markers on the highway where you think it is likely someone will need a tow (entrances/exits, merges, rest stops, etc.). To ensure you can help quickly, you want to place your tow trucks so that from every marker, at least one truck is at most 3 miles away. Find a minimum length list of sites where you can place tow trucks to satisfy the requirement, written as a list of integers $a_1, a_2, \ldots, a_m$ in increasing order. Note that the sites that you pick need not be a subset of the marked locations.

a) Write a summary of the problem.

Feel free to work with the people around you!

Reduction vs. techniques from scratch

Read and **summarize** the problem

Known algorithms

- Stable matching
- Graph algorithms
- …etc.
- Network f
- Linear pr

Techniques from scratch

- Greedy algorithms
- Divide and conquer (week 4)
- Dynamic programming (week 5)

None of these seem right for today's problem,
so we'll try a greedy algorithm!

Does the problem remind me of an algorithm I've seen in class?

Generating ideas

Generating ideas

Read and **summarize** the problem

Decide to use **known algorithm** or **techniques from scratch**

not covered this section

no idea

have idea

Solve examples to get ideas

Check that idea **isn't easily falsified** or **slow**

Write pseudocode, proof, and running time analysis

Generating ideas

If using **techniques** from scratch:
(for today, greedy algorithms only)

no idea ↙ ↘ **have idea**

Solve many examples by hand

- In the beginning, don't worry about general strategy
- Think about what **patterns** appear
- If your brain is magically solving small examples, try bigger ones

idea! →

← **yes :(**

Ask yourself questions

- Can I break my strategy with a nasty example?
- Does my strategy ever waste time? Can I optimize it?

↓ **all good!**

Ideas for greedy algorithms

- What's a **greedy algorithm**?
    - Follows a rule to keep picking something
    - Doesn't consider the future
    - Doesn't go back to fix things

- Coming up with many greedy ideas should be easy. Finding the correct greedy idea will usually require trial and error or insight.

Problem 1 – Line covering

b) We will practice generating ideas.
    i. Solve these by hand. Don’t worry too much about greedy strategies yet.

1, 2, 4, 10, 12

0, 1, 3, 5, 7, 8, 13, 14

Feel free to work with the people around you!

Problem 1 – Line covering

1, 2, 4, 10, 12

0, 1, 3, 5, 7, 8, 13, 14

ii. Suppose you came up with the greedy idea:

"Put a truck on the first uncovered marker."

Check that this idea works on the above examples. Then, try to break this idea by coming up with an example where it doesn't work.

Feel free to work with the people around you!

Problem 1 – Line covering

iii. Come up with a new greedy idea that solves your new example. Does the idea work? If not, continue the process until you have a working idea.

Feel free to work with the people around you!

Writing up your idea

Writing up your idea

Read and **summarize** the problem

↓

Decide to use **known algorithm** or **techniques from scratch**

not covered this section

no idea

have idea

Solve examples to get ideas ⇄ Check that idea **isn't easily falsified** or **slow**

↓

Write pseudocode, proof, and running time analysis

Writing up your idea

Once you have an **efficient, working idea**:

1. Translate it into **pseudocode**.
    - More precise than English, but easier to understand than code.
    - No hard rules, but see handout from last week for common styles.
2. **Prove** the pseudocode correct.
    - We'll cover greedy-specific tips today!
3. Write up the **running time** analysis.

Problem 1 – Line covering

c) Write the pseudocode for the solution.

Algorithm proofs refresher

- As always, prove **validity**, **termination**, and **correctness**.

- Correctness always means:

  "My algorithm's output matches the problem summary's expected output."

- For greedy algorithms, correctness means "My output is an **optimal solution**."

  In other words, two things to prove:
  - "Output is a valid solution."
    - "The list $a_1, \dots, a_m$ is in increasing order and covers all markers."
  - "Output is optimal."
    - "All other valid solutions use at least m trucks."

Algorithm proofs refresher

For optimality, there are some common strategies:

- **“Greedy stays ahead”**: For all other solutions, show by induction that at every step, your solution is at least as good.
- **“Exchange argument”**: For all other solutions that differ from yours, show how to replace a part of the other solution, so that the quality improves or stays the same (but never decreases).
- **“Structural argument”**: (less common) Find a “hard subset” of the input that immediately implies why other solutions must also be as bad as yours (or worse).

Problem 1 – Line covering

d) Write a proof that your pseudocode is correct.

Each line is valid:

Termination:

“The output is in increasing order.”:

“The output covers all markers.”:

Problem 1 – Line covering

d) Write a proof that your pseudocode is correct.

Now for the harder part. For this section, try to write a “greedy stays ahead” proof!

“All other valid solutions use at least m trucks.”

i. What is the “greedy stays ahead” claim?

Problem 1 – Line covering

ii. Prove the “greedy stays ahead” claim using induction.

Feel free to work with
the people around you!

Problem 1 – Line covering

e) Analyze and prove the running time with big-O in a few sentences.

Summary

Read and **summarize** the problem

↓

Decide to use **known algorithm** or **techniques from scratch**

not covered this section

no idea

Solve examples to get ideas

have idea

Check that ideas are **correct** and **efficient**

↓

Write pseudocode, proof, and running time analysis

Thanks for coming to section this week!