Problem Solving and Tech Interview Practice

CSE 417 21AU
Lecture 24

We’re doing two things at once

1. Practicing a problem solving process.

Sooner or later, you’re going to get stuck (on homework or in real life).

Sitting there saying “I don’t know the answer yet” is frustrating.

Giving yourself specific questions and techniques makes your life easier.

2. Give advice for technical interviews

It turns out, they’re trying to evaluate whether you’ll be able to solve problems. And watching your process is a big part of what they’re doing.

Take all industry advice with a grain of salt

I’ve spent essentially my entire career in academia

I’m relying on a combination of what I’ve heard from colleagues and problem-solving strategies that work in my experience.

Resources

Cracking the Coding Interview (McDowell) or another book like it.
Copies are at the library!

General Advice from Kasey Champion

Leetcode (we…borrowed…more than one homework problem from there).

General resume advice (and other resources). Some specific to Allen School.

Technical Questions

Your interviewer wants to see

1. What you know
2. Can you communicate technical ideas
3. Can you problem-solve

Getting the “right” solution helps with point 1. Not points 2 and 3.

You also need to talk through what you see in the problem/how you solve it.

The best way is to practice. Find a friend, a set of interview questions, and a whiteboard and practice.

Technical Questions

Show you can problem-solve.

It's tempting to "skip" the problem-solving steps if you know the answer. It's a good idea to still do all of them.

You can go quicker, but give them something.

Don't feel obligated to use this exact method – others exist, but you should hit these points. **Even if you know the best algorithm from the time you read the problem statement.**

A sample Problem: https://leetcode.com/problems/house-robber/

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security systems connected and **it will automatically contact the police if two adjacent houses were broken into on the same night**.

Given an integer array `nums` representing the amount of money of each house, return *the maximum amount of money you can rob tonight* ***without alerting the police***.

TEBOW IT

Talk

Example

Baseline (aka Brute Force)

Optimize

Walk-Through

Implement

Test

https://commons.wikimedia.org/wiki/File:Tim_Tebow_in_the_dugout.jpg

Tim Tebow:
Football player turned football analyst turned simultaneous-baseball-player-and-football-analyst turned football player turned football analyst again.

Acronym from Kasey Champion; similar process in *Cracking the Coding Interview*

Talk

Give you time to take deep breaths
Confirm you haven't missed anything
Can sometimes get a simplifying assumption added.

Make sure you understand every technical term in the prompt (if any).

Ask questions to make sure you've got the idea.

Ask about simplifying assumptions.

Pick out any key pieces of the problem

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security systems connected and **it will automatically contact the police if two adjacent houses were broken into on the same night**.

Given an integer array `nums` representing the amount of money of each house, return *the maximum amount of money you can rob tonight* ***without alerting the police***.

Talk

If I enter houses 2 and 4 only, I don't set off an alarm, right?

The houses are just in a line, I'm only worrying about "one side" of the street?

I'm planning just one night?

The amount of money is an integer? Can it be negative?

I don't have to record which houses, just the final number?

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security systems connected and **it will automatically contact the police if two adjacent houses were broken into on the same night**.

Given an integer array `nums` representing the amount of money of each house, return *the maximum amount of money you can rob tonight* ***without alerting the police***.

Talk

Self-checks

Do I know the method signature (inputs and return type)?

Is there any information in the prompt that I feel is useless?

There usually isn't any, so maybe ask a question about that

Example

You'll need these later!

Make a sample input or two, find the answer and **confirm it's right with the interviewer**

Example 1:

```
Input: nums = [1,2,3,1]
Output: 4
Explanation: Rob house 1 (money = 1) and then rob house 3 (money = 3).
Total amount you can rob = 1 + 3 = 4.
```

Example 2:

```
Input: nums = [2,7,9,3,1]
Output: 12
Explanation: Rob house 1 (money = 2), rob house 3 (money = 9) and rob house 5 (money = 1).
Total amount you can rob = 2 + 9 + 1 = 12.
```

Baseline

Finally, designing the algorithm. What we came here for.

First algorithm that comes to mind.

Doesn't matter how slow.[1]

This is a good problem solving technique, sometimes you want to optimize the first thing you think of.

Sometimes you don't, but it's still useful for you as a human.

You're going to be nervous. It's a lot easier to think creatively with the baseline in your back pocket. You won't end this interview having written no code.

[1] ok, it does matter how slow, but that's not the point ***yet***

Baseline

If you see what to do right away that's fine! You can short-circuit this step. But if it doesn't work come back here! I don't like to spend more than a few minutes with no back-up plan.

Remember the examples you did! Those can help now. Were you doing a brute force on those to solve them?

Baseline

Well I guess we could check all 2^n sets of houses.
Will be really slow, but will work.

Let's try to do better.

Optimize

Sometimes you're going to speed up the baseline (say with better data structures).
Other times you're starting from scracth

Now, let's get a better algorithm.

What did you see in your examples?

What does this remind you of?

We're finding the maximum something in an array, that might remind you of maximum subarray sum. Maybe DP?

Let's start there…

Optimize

Let's think recursively. Start with the last house.

We're currently thinking about house i.

If we want to include i, then we want…

If we want to exclude i then we want…

Optimize

Let's think recursively. Start with the last house.

We're currently thinking about house i.

If we want to include i, then we want…

If we want to exclude i then we want…

Optimize

Let's think recursively. Start with the last house.

We're currently thinking about house i.

If we want to include i, then we want…

The best robbing plan from 0 to $i - 2$

If we want to exclude i then we want…

The best robbing plan from 0 to $i - 1$. We don't have to include $i - 1$ through.

Let OPT be the value of the best robbing plan for the array from 0 to i.

Optimize

You could also get an

OPT/Include split like we've often seen if you jump right there.

Walk-Through

Now we've got our idea…time for a very quick check

What happens on one of your small examples?

`nums`

0	1	2	3	4
2	7	9	3	1

`OPT`

0	1	2	3	4

Walk-Through

You might notice bugs or edge cases (like the base cases here!) if you go slowly

Now we've got our idea…time for a very quick check

What happens on one of your small examples?

`nums`

0	1	2	3	4
2	7	9	3	1

`OPT`

0	1	2	3	4
2	7	11	11	12

Implement

Tips: Use default data structures.

Your interviewer probably doesn't remember all the tiny details of java default libraries either, so feel free to say "I'm going to assume the library has an X method" or "it works in this way" if you can't remember whether it's the Java or C++ version that works a certain way.

You might never have written code on a whiteboard (or typed without an IDE auto-completing things). You should practice those skills!!

Use comments and break-off easy methods/repeated code.

If you're running out of time, your interviewer might let you skip it.

```
int robbingNumber(int[] input){
  int[] opt = new int[input.size]; //opt robbing number where we can access homes 0,…,i.
  opt[0] = input[0];
  opt[1] = Math.max(input[0], input[1]);
  for(int i=0; i < input.size; i++){
        opt[i] = Math.max( opt[i-1], opt[i-2]+input[i]);
}
  return opt[input.size-1];
```

Test

If you haven't already, do your big-O analysis

Then test your code!

- Use the examples you made early on
- Think about small cases (empty input, null input)
- Think about edge cases (repeated numbers)
- Double-check you've used all assumptions

Another Problem

Drive, a new rideshare company, focuses on getting their customers long-distances by having many individual drivers take them through separate parts of the journey. Drive has decided on a set of specific locations where pickups and drop-offs can happen, and they have the cost to ride between every pair of locations.

To incentivize you to try the system, Drive has given you a token for one free ride. Because the system is new, they haven't mapped out how you'll go long distances. It's up to you to decide which locations to visit on the way from your starting point to the end point and which one you will use your coupon on.

Describe an algorithm that will let you get from your given start and end points while spending the minimum amount of money (with your token).

A Very Fun Trick

Super optimized trick: Make two copies of the graph, for every edge (u, v) also add a weight $\mathbf{0}$ copy from u in copy 1 to v in copy 2 (along with ones within each copy at weight equal to the time to drive).

Jumping from copy 1 to copy 2 is using your token. Dijkstra's from start in copy 1 to destination in copy 2 is your final answer right away.

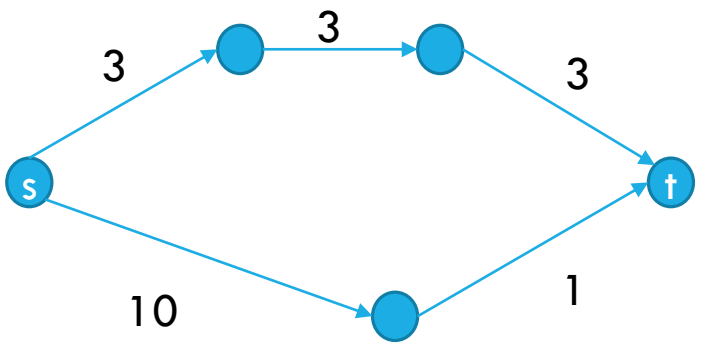
Multiple graph copies is surprisingly useful to represent these "you can do a weird thing a constant number of times" questions.

Other options

One at a time, make every edge weight 0 and run Dijkstra's.

A factor of m slower than optimal, but definitely works.

Cannot just find a shortest path and use a token on the most expensive ride.



One More For the Road

Given a list of integers, list all pairs $(A[i], A[j])$ such that $A[i] + A[j] = 10$.

Optimized options

There are multiple options here:

Data structures-based (use a hash table, lookup $A[i]$'s potential paired value)

Clever trick: sorting makes this easier!

Sort the list. You can use binary search to find where the "potential pair" should be.