Overview of Technical Interviews

Week 2 CSE 492 J

A Brief History of Technical Interviews

- Invented by Microsoft, based on Bill Gates' love of puzzles
  - Quickly adopted in the 90s by software firms
  - Now used by the vast majority of software engineering hiring managers
- Google historically has been looked to as the leader in the space
  - Where the Intellectual Horsepower and Estimation questions come from
  - Google has moved away from these and thus so has the rest of the industry
  - IQ based questions have repetitively been shown to be discriminatory, in some states are illegal to have public school students even take these types of questions
- Lots of companies are experimenting with alternatives now, but no clear successor has arisen
  - "Take home" coding tests
  - Pair programming tests
  - more!

Poll Time!

www.pollev.com/champk

What types of evaluations have you encountered when applying to technical positions?

Why technical interviews suck

- They do not simulate work
  - Timed
  - Stressful
  - “Toy” problems
- Companies have inconsistent processes, questions and evaluations
  - Engineers typically pick their own question
  - Feedback is typically in large buckets prone to inconsistency
  - Translation from interview performance to “pass” is inconsistent
- Engineers are not well trained to be interviewers
  - There is no such thing as a “technical bar”
  - Engineers just gate keep based on the gates they were presented with
  - Some believe in hidden requirements

Why are we learning how to do these?

- Companies still use them
  - Companies love data
  - Engineers borrow process and policies, not just code
  - Terrified of “false positives”
- They are unstandardized “standardized” tests
  - Biased towards represented communities due to sharing of “tribal knowledge”
  - Companies still look for “culture fit” *barf*
- You should be interviewing the companies just as much as they are interviewing you
  - Bad practices, weird vibes are all indicators of a bad company culture
  - Don’t ignore red flags

Where does the Interview fit in the process?

Typical Recruiting Process

1. Application
2. Coding Challenge
3. Technical Screen
4. “Onsite” Interview
5. Offer
6. Negotiation
7. Acceptance

Typical Recruiting Process

1. Application
    a. Online (company websites, LinkedIn, Indeed, etc)
    b. Mostly ask for information on your resume
    c. Time consuming, tedious, repetitive, but definitely worth the effort!
    d. You won't hear back from most of them.
    e. Referrals are very helpful
2. Coding Challenge
3. Technical Screen
4. "Onsite" Interview
5. Offer
6. Negotiation
7. Acceptance

Typical Recruiting Process

1. Application
2. Coding Challenge
    a. Sent out by companies via email (so check your inbox often!)
    b. Typically expire in a week or two
    c. 60min - 90min time limit to solve 1 - 4 leetcode style questions on websites like Hackerrank
    d. Sometimes come after a recruiter call
3. Technical Screen
4. “Onsite” Interview
5. Offer
6. Negotiation
7. Acceptance

Typical Recruiting Process

1. Application
2. Coding Challenge
3. Technical Screen
    a. Video or audio call with an engineer/engineers (MAKE SURE YOU HAVE CELL SERVICE + WIFI)
    b. Usually 1 or 2 20 - 60 minute long interviews
    c. Coding questions on a shared screen
    d. Typical technical interview structure (We will talk more about this later!)
4. “Onsite” Interview
5. Offer
6. Negotiation
7. Acceptance

Typical Recruiting Process

1. Application
2. Coding Challenge
3. Technical Screen
4. “Onsite” Interview
   a. Free lunch and free
   b. Anywhere betwee
   c. Face to face with
   d. Coding questions on
   e. Typical technical interview
5. Offer
6. Negotiation
7. Acceptance

include a “lunch interview”

Parts of the Technical Interview

Typical Technical Interview Structure

1. Introductions (Under 5 minutes)
2. Project discussion (5 minutes ish)
3. Coding question (45 minutes ish)
4. Your questions (5 minutes ish)

1. Introductions

- “Hello.”
- “How are you?”
- “Who are you?”
- “Why do you want to work here?”

Be nice, be polite, be confident.

2. Project Discussion

Have 1 or 2 project(s) that you feel confident talking about in front of people who know what's up.

- "Are school/internship projects okay?"
- Don't be afraid to brag about what you and your team did, but be prepared for questions.
- "We" is a good word, but don't overuse it.

2. Project Discussion Example

"I spent this summer working at an advertising network, specifically trying to drive engagement on our video ads by A/B testing new ad content and formats. I worked primarily in the backend and used Python and R for data analysis. I produced an 8% improvement in click-through rates across the board over six weeks of testing."

Follow up questions:

- How long did you work on this project?
- How big was the team working on this, what was your role specifically?
- Why did you choose that technology stack?
- What was the biggest bug you encountered and how did you fix it?
- If you redid the project what would you do differently?

3. Coding Question

- This is the bulk of technical interviews.
- Don’t freak out, I promise you will be fine.

(But also start reviewing 142/143/332/373 if you haven’t started already.)

4. Your Questions

- Show your interest
- Learn more about the company

Prepare some questions before you go in, and make sure to ACTUALLY listen to their answers.

4. Your Questions Examples

- Create a positive interaction between you and your interviewer.
  - What is your favorite part about working for X?
  - Where do you see X in 5 years?
  - What keeps you excited and motivated to show up to work every day?
- This is your opportunity to probe the company and decide if it's a good fit for you. Reflect on your values and come up with some non-aggressive questions you can ask.
  - Large tech companies can sometimes have a reputation for being ultra competitive. Do you have any perspective or advice for a young, hungry new grad that cares about work but also cares about life outside work?
  - Covid-19 has brought to light how important it is to work for an employer that takes are of its people. How did X make you feel safe and valued during the situation?
- Don't ask rude questions
  - Did I pass the interview?
  - How much do you make?

Before the Interview

Pick a language

- Strongly recommend something **object oriented** languages.
  - Java, Python, C/C++ are the most common ones.
  - Kasey says “don’t pick C”
- Knowing your language thoroughly is more important than which language

Review everything

- APIs!
  - Make a notes doc while you do practice problems and write down each API you have to look up
- How to do basic function calls
- Data structures & algorithms you learned
- What you know about the company

Practice

- Treat the interview like a test
- Try to code without your IDE
- Code and talk at the same time
- Time yourself!
- Utilize your resources
  - Crack the Coding Interview
  - Hackerrank
  - LeetCode
  - Glassdoor / Online forums

Question Patterns

Data Structures	Algorithms	Concepts
Linked Lists	Breadth-First Search	Bit Manipulation
Trees, Tries, Graphs	Depth-First Search	Memory
Stacks & Queues	Binary Search	Recursion
Heaps	Merge Sort	Dynamic Programming
Arrays & ArrayLists	Quick Sort	Big O Time & Space
Hash Tables	Shortest Path	

Technical Interview Strategies

Crush **E**very **A**mazing **O**pportunity **I**n **T**ech

Clarify	ask follow up questions
Examples	sample input / output (test cases)
Approach	outline a solution (brute force ok)
Optimize	review approach, briefly discuss runtime and memory usage
Implement	write your code!
Test	review test cases

Practice question!

Given a String, find the duplicate character

1. **C**rush = **C**larify

1. **C**larify

- Clarifying questions
  - “Is there guaranteed to be a duplicate?”
  - “Is there only going to be one duplicate?”
  - “What is the return value?”
  - “What if the input String is empty?”
  - “Does capitalization matter?”

- Common questions
  - Can I modify input?
  - Can I use extra storage?
  - Can I assume good user behavior?

2. **E**very = **E**xamples

2. **E**xamples

“aa” -> ‘a’

“aabbcc” -> ‘a’

“AaBbCc” -> ‘\0’

“abbc” -> ‘b’

“abca” -> ‘a’

“” -> ‘\0’

“abcba” -> ‘a’

3. **A**mazing = **A**pproach

3. **A**pproach

We could create an outer loop that looks at `arr[0]` then an inner loop that compares it to every subsequent index

4. **O**pportunity = **O**ptimize

4. **O**ptimize

Nested loops is a O(n^2) solution, we can do slightly better if we use extra storage.

- We can create a char array, sort it, look for neighbors. O(nlogn) + N extra storage
- We can create a hash map then find collision. O(n) + N extra storage

5. In - Implement

5. Implement

- Write function header first
- Outline control structures
- Start with average case, address edge cases as you go and after
- Do your best to talk as you go
- Keep examples (test cases) in mind as best you can
- DO
  - Use good names
  - Use comments for clarity
- DON’T
  - Bother with too much modularity / maintainability design

5. Implement

```
public static char strDupes(String input) {
   char result = '\0';
   if (input.length() > 0) {
      Map<Character> dupes = new HashMap<Character>();
      for (int i = 0; i < input.length(); i++) {
         if (dupes.containsKey(input.charAt(i)) {
            return input.charAt(i);
         } else {
            dupes.put(input.charAt(i);
         }
      }
   }
   return result;
}
```

6. **T**ech - **T**est

7. Test

Let’s run through the examples we had in the very beginning…

Practice question!

Given an array of n integers, return an array that stores the PrefixSum of the input array.

TEBOW IT

1. Talk
2. Example
3. Brute Force
4. Optimize
5. Walk Through
6. Implement
7. Test

Practice question!

Given an array of n integers, return an array that stores the PrefixSum of the input array.

1. Talk

1. Talk

- Clarifying questions
  - “What do you mean by ‘PrefixSum’?”
  - “What if the input array is empty?”

2. Example

2. Example

If the input array is [3, 4, 1, 2], our output array should be [3, 7, 8, 10].

3. Brute Force

3. Brute Force

We can loop through the array, and for each index, we sum up the integers before it and set its corresponding index in the output array.

4. Optimize

4. Optimize

Instead of the O(n^2) solution, we can do slightly better!

We can loop through the input array once, and at each index i, set the value that i holds to be output[i - 1] + input[i].

5. Walk Through

6. Implement

6. Implement

```
public static int[] prefixSum(int arr[], int n) {
        int[] prefixSum[0] = new int[n];
        prefixSum[0] = arr[0];
        for( int i = 1; i < n; ++i ) {
            prefixSum[i] = prefixSum[i-1] + arr[i];
        }
        return prefixSum;
}
```

7. Test

7. Test

Let’s run through the example we had in the very beginning…