Technical Interviews

What to expect when being scheduled for a technical interview:

Technical interviews come in several different formats. The process may even include a mixture of multiple types, so it's important to be prepared for the specific format that you are scheduled for. Recruiters will be able to confirm the format and provide any basic prep materials or technology checks you need to make once you are scheduling it.

- **Technical Screening** - These tests are typically unsupervised timed tests where you are asked questions to determine your technical ability. Some tools may monitor your browser usage (or do a full screen recording) to ensure you are only using materials that are allowed.
- **In-person Interview** - On-campus or interviews onsite at the company are often conducted via whiteboard. In this type of interview you will be relying on your basic coding skills, your knowledge of the coding process, and will be expected to showcase how you might collaborate when working with a team.
- **Virtual Interview** - Virtual interviews will have a similar process to in-person interviews and allow you to chat with your interviewer over the phone or on VC while having a shared workspace / IDE to show your work.

Interview Length & Planning Ahead

Coding interviews can range between 30-60 minutes (and sometimes longer). There may also be multiple rounds of interviews, so you may be required to complete more than one technical interview. You will be given this information at the time of scheduling so you can plan accordingly. Always make sure to reserve time before your interview to get settled and make sure your technology is working as expected.

What is the purpose of the interview(s)?

At the very highest level, these interviews are aimed at understanding your coding ability. Live interviews (those involving an interviewer) will also assess how you think about tackling these problems, how you actually go about solving them, and how you communicate it to others.

Interviewers will be looking for a strong knowledge of algorithms and data structures which will signal that you can write efficient code. They will ask questions and have you dive deeper into the problem to try to understand where you are comfortable and where you may have knowledge gaps. These interviews will be designed to push you to your limit of knowledge.

What types of questions are likely to be asked?

- Simple logic puzzles like the ones commonly on standardized tests - find the next number in a sequence, find the main idea of a paragraph, etc.
- Long coding questions where they will ask you to fully implement a specific function that does something like reverse a string, count duplicates, etc.
- Short coding questions where they ask you to find and fix a one line bug in the code
- Code analysis where you will need to analyze the runtime of a particular algorithm

Where to start?

Select your language. In most coding interviews, you will be able to select any language you prefer to use. You should select the language that you feel most comfortable coding in given the timeframe you will have to solve questions.

Ask questions! Read the question and get clarification on any pieces that you don't understand or have total clarity on. Even if you think you are clear on what the question is asking for, you should always **confirm your understanding of the problem** with the interviewer before moving forward to ensure you are on the same page. Questions are sometimes vague on purpose to see how you might handle real-world situations where you need to gather more data.

First **think through the principles that might help solve this problem**. This will help ensure you can answer problems that you haven't seen or solved before. Talk through your thought process. Some questions you should be starting with are:

- Are there different ways you could go about solving this problem?
- What would be tradeoffs from using one method over another?
- Are there any inputs that might limit landing on a solution? How would you want to handle those?
- How will you test your solution to make sure it works?

Design your algorithm and **communicate your thoughts and ideas** so the interviewer is able to follow along. Start with pseudocode to ensure that your solution will work. Note any challenges or areas you'd need to focus on or potential issues (ex: Storage or time complexity). Once you are satisfied and believe you have the appropriate path for a solution, begin translating your pseudocode into actual code. Ensure your code is clean and error free as you go.

Test your solution and actually try to break it. This will showcase how you can go back through your work, identify issues, and propose solutions to fixing and making it better in future iterations.

How do you practice?

Look through common questions and try some yourself. Start with the easier questions and move on to more difficult ones once you feel confident in your ability to solve them while also being able to thoroughly explain your solutions. Understand where you feel confident and where you might need more practice.

- ➔ You can find common questions here or here.

Watch mock interviews to get familiar with how some interviewers may walk you through the questions:

- ➔ https://interviewing.io/recordings/
- ➔ https://neetcode.io/
- ➔ https://www.youtube.com/watch?v=pkyietkjeaI

Do your own mock interviews. Utilize classmates, mentors, friends, or sites that provide mock interview experience (ex: https://pramp.com or https://interviewing.io) to get comfortable with the format of these technical interviews.