Karan Chawla
December 19, 2019

Introduction

BLANK! Every student has had this feeling at some point, whether during a presentation or during a final exam. When your mind goes blank, the task in front of you suddenly seems impossible. In life, some situations have higher stakes than others. The presentation might be for investors. The exam could determine your grade.

As computer science undergraduate students, one of the highest stakes we face is the result of a technical interview. Many students pursue an internship for the summer and find this programming-style interview standing in the way. People who have not yet accumulated technical interview experience may find the task of preparing to be daunting. Since countless online sources exist, how to begin preparing can also seem like a mystery. This comprehensive yet brief guide acts as a jumping off point. The guide explains the technical and behavioral portions of the typical programming interview to allow students to maximize their chances of success.

The tips and tricks in this guide are based on my personal experience and a variety of reputable external sources. I have performed well on programming interviews for companies including Google and Facebook in the software engineering category and aim to share a few of the lessons I have learned. The guide is broken up into the following four main sections:

1. Third-Party Resources
2. Problem-Solving Strategy
3. Leetcode Preparation
4. Interview Behavior

While I do believe that all of these sections will provide helpful information, if you feel comfortable with certain aspects of the interview already, feel free to jump around to explore the topics you would find most useful.

Third-Party Resources

The internet has many resources that help with interview preparation, but without prior experience, people have difficulty deciding which sources to turn to. In addition to online sources, many resources with valuable information are in print. The subcategories below act as a way to separate out third-party resources, as each style comes with unique strengths. The later sections of this guide will also refer to these sources when discussing optimal preparation.

Textbooks

One of the cornerstones of algorithmic preparation is Thomas Cormen's Introduction to Algorithms textbook. This book is widely regarded as a must-read for thoroughly understanding fundamental algorithmic principles. The early chapters of the textbook cover concepts about introductory data structures. The typical programming interview question is a twist on these fundamental concepts, which means any student could benefit from a refresher in these topics. This textbook is less geared toward preparation through specific programming questions and more in line with a course textbook.

Preparing by analyzing interview style questions is also valuable practice. For this purpose, students use the textbook Cracking the Coding Interview (CTCI), by Gayle Laakmann. Laakmann is the founder of CareerCup, a company that specializes in helping people prepare for coding interviews. CTCI is famously known for its tricky programming questions and behavioral tips, both of which provide valuable insight.

Coding Preparation

Leetcode is perhaps the most important online platform for coding preparation. If you talk to computer science students who have done these interviews before, Leetcode is sure to come up in conversation. The free platform has a database of interview style programming

questions along with an editor on the side that users can program in. The user can then test their algorithm for the problem on built-in test cases that ensure only the best solutions pass. Just as lawyers do trial runs and pilots use simulators, people preparing for their interview need to sharpen their skillset with these ready-made problems.

Geeks for Geeks is a website that walks through common algorithms for searching, sorting, etc. and describes how to optimize the process. In interviews, programming an efficient solution is important, so reading about the iterative improvements Geeks for Geeks describes is educational. This page on the website lays out common categories of questions, like array manipulation, strings, tree-based searches, etc. and also links to Leetcode.

General Tips/Handbook

According to Yangshun Tay's programming interview guide on Github, specific techniques can be used across a variety of challenging questions. Tay presents a few of these techniques for different data structures and goes on to explain the basics behind these strategies. Among these techniques are dynamic programming, recursion, and tree traversals, which I can confidently say appear commonly in interviews.

Problem-Solving Strategy

While the specifics of solving an interview question depend on the question itself, a generalized framework for tackling challenging problems helps keep students relaxed during the interview. Usually, the interview starts with quick introductions. At times, the interviewer will ask about certain elements of your résumé. After no more than five minutes, they will present you with a programming problem.

The first step is always to make sure you understand the problem statement. Do not be afraid to ask questions to the interviewer to clarify any doubts you have about the problem. Many students quickly jump into coding right away without fully considering the problem and end up confused. The second step is to think of an inefficient solution for the problem, which is often done with brute-force methods. This step may sound unintuitive, since we are aiming to program an efficient solution, but starting off with an easy-to-find solution and iteratively improving helps describe your train of thought. The third step as mentioned is to iteratively improve by thinking about how this problem is similar to others you have seen in the past.

After you believe you have completed an efficient solution, take a pause in order to calculate the runtime complexity of your approach. Interviewers typically will ask you about this detail if you do not offer the complexity yourself. Finally, you should take a minute to create a sample test case and run through your algorithm with this input. Often times, programmers find a small mistake or edge case just by thinking about the result of one sample input.

According to Geeks for Geeks and Tay's Handbook, a few categories of programming questions are very common and require specific techniques to solve. For instance, some problems will require you to find an optimal solution for a given setup. In this case, you should think about whether solving a smaller subproblem can lead you to a solution for the overall problem, which is a trait of Dynamic Programming. For problems with tree-based data structures, you can use in-order traversals, breadth-first search traversals, and depth-first search traversal to accomplish many tasks. If you want to read more about these techniques, you should read Tay's breakdown of the different categories.

In a general sense, if you are familiar with these techniques, you have a greater chance of crafting the optimal solution for your interview problem. Remember to pace yourself, but also do

not waste time on extremely specific edge cases. One common pitfall occurs when people are tasked with solving an easy problem. Sometimes, these people will spend excess amounts of time making sure every detail of their implementation for the easy problem is correct, when in reality the interviewer meant for this problem to lead into a second, more complicated task.

Overall, this general problem-solving strategy helps add a much-needed structure to one's thought process. If the problem seems difficult, all you have to do is take a deep breath, make sure you understand the task at hand, and slowly build up your solution. If you practice with the common techniques used to solve these questions, you will be more prepared for interview day.

Leetcode Preparation

Since practice makes perfect, Leetcode is the place to go for the most effective training. The online platform has hundreds of programming questions ranging from easy to hard difficulty ratings. Creating an account can be confusing, but the best page for interview questions is the "Algorithms" category. Before you start on any coding problems that catch your eye, you need to select your programming language of choice.

Choosing the Right Language

Selecting your programming language is an important consideration and will affect your language-specific preparation. Often, the company you are interviewing for will ask you for your preferred programming language ahead of time. Your decision should reflect a high level of expertise in your chosen language. I will go over the strengths of two of the most common choices: Python and Java.

Python is a programming language built around scripting, which means you won't have to worry about data types, return types, and syntax issues. Scripting languages are useful during

interviews because programmers have to account for these details in other languages, which takes up valuable interview time. Nonetheless, you still need to learn about the basic libraries and built-in data structures for Python, including sets, dictionaries, arrays, etc. The built-in support for these data structures is convenient and useful for solving many problems, given you know which structure to use in different scenarios.

Java is a programming language with strict typing and many built-in data structures. Java has a strict enforcement of data types and syntax rules. At times, this requirement makes creating perfect Java code during an interview difficult. However, the strict rules also force programmers to come up with structured code right off the bat. This structure gets the interviewee to think deeply about the inputs, outputs, and data structures needed for the problem, which can save debugging time down the road.

The question of which programming language to choose has no obvious answer. People usually prefer to choose based on their personal experience and also their preferred style of programming. Overall, Python has fewer restrictions, but Java has more structure.

Leetcode Tips

After choosing the programming language, you should attempt to solve Leetcode questions to gain experience. Start with the easy rated problems, even if you believe these problems are too easy for you! Often, students look at a problem and think the problem is easy but cannot translate their solution into working code. The process of coding even elementary algorithms helps you familiarize yourself with language-specific details and commonly missed edge cases.

For each problem you complete on the website, you should keep track of the data structures you used to reach an optimal solution. Certain operations like search and delete are

efficient on specific data structures, so remembering these details will come in handy during the interview. Interviewers are impressed by candidates who have a thorough understanding of the tradeoffs for operations between different data structures. Also, these differences play a role in the overall runtime complexity of your solution, which often determines whether you pass the interview.

Practicing on Leetcode is not a one-day affair. Completing five questions per day for one week is significantly more helpful than staying up one night to do 35 questions. For the best results, you should make Leetcode a daily portion of your preparation. You should also consider using a website like CoderPad, which simulates the text editor used in a real interview setting. If you follow this advice and consistently practice, interview day will seem much less daunting.

Interview Behavior

The technical programming interview is unlike interviews in other fields, but the core of all interviews is human-to-human interaction. This aspect of the interview trips up many students, since people usually code in silence. According to Facebook and other tech companies, students are expected to talk about their thought process, elaborate on the solution they have constructed, and reason about the solution's runtime complexity.

Some interviewers like to take the first five minutes to talk about your personal projects and experiences. During this portion, you should humanize yourself with an interesting story about a team or project experience. Make sure to not take up too much time with a dragged-out story!

When the interviewer presents you with the question, you can ask for a few minutes to think about the problem statement. After all, no one is expected to come up with a solution right

away. When you are coding, however, detailing your thought process to the interviewer can be helpful. The interviewer will understand the approach you are attempting and may even offer suggestions to guide you along the right path. When you don't communicate, the interviewer often has difficulty assessing your thought process and your final code.

You can make mistakes! Students are expected to make errors. If you can catch your error and modify your algorithm to account for the mistake, the interviewer will notice your ability to improve. Sometimes, you have to completely change your solution halfway through the interview when you realize you have been pursuing the wrong path. Take a deep breath to reset, let your interviewer know why your current solution does not work, and don't get discouraged. If you are in a tough spot and are unsure how to continue, try using the following phrases:

- "I see why…but how can I…"
- "My current approach could lack…How should I focus on improving it?"
- "Am I going down the right path?"

These phrases signal to the interviewer that you are actively trying to improve your approach but may need some guidance. You can ask for some guidance, but you should avoid using these phrases too often.

Conclusion

There is no foolproof method of preparation for programming interviews. People often get curveballs and sometimes are not able to solve the coding problem. You should always keep a positive attitude, because each opportunity to interview will give you more experience. If you use the advice and resources from this guide, you can maximize your chance for success. Good luck with your interviews!

Works Cited

Cormen, Thomas, et al. *Introduction to Algorithms*. 3rd ed., 2009.
This algorithms textbook is widely regarded as a must-read for learning basic algorithmic principles that are assessed in programming interviews. I use principles from their elementary chapters in data structures to introduce technical concepts to the reader at the beginning of my guide.

Facebook. "Preparing for Your Software Engineering Interview at Facebook." *Facebook*, https://www.facebook.com/careers/life/preparing-for-your-software-engineering-interview-at-facebook/.
Facebook is one of the largest tech companies in the world, which also means the company has a lot of experience in conducting interviews. The tips present on this page are representative of the practices published online by many big tech companies. These tips are helpful to talk about in my guide, as they provide a good view of what to expect when interviewing with a big company.

Laakmann, Gayle. Cracking the Coding Interview. 4th ed.
Laakmann is the founder and CEO of CareerCup, a company that specializes in helping people prepare for coding interviews. His book, abbreviated CTCI, is famously known among computer science students who are looking for in-depth preparation. While the book has sound programming advice, the book mostly helped me with the behavioral aspect of the interview. This is a facet of the preparation that many students ignore or are unaware of.

"Leetcode Algorithms." Leetcode, https://leetcode.com/problemset/algorithms/.
Leetcode is a website that has given many computer science students an opportunity to test their interview coding abilities. The platform is well known and also offers premium user subscriptions. The site allows users to choose problems to solve and offers test cases for the users to check their code against. The explanations for the solutions of these problems helped me create a general walkthrough for how these problems can be tackled and solved.

"Must Do Coding Questions." Geeks for Geeks, 2019, https://www.geeksforgeeks.org/must-do-coding-questions-for-companies-like-amazon-microsoft-adobe/.
Geeks for Geeks is a website that has helped computer science students tackle difficult algorithmic questions for years. This page specifically deals with the different types of programming questions that may come up during a technical interview, including questions on arrays, strings, trees, etc. The web page helped me break down a few core areas of preparation for the interview and suggest some problems representative of each area.

Tay, Yangshun. "Tech Interview Handbook." Github, 2019, https://yangshun.github.io/tech-interview-handbook/algorithms/array.
Tay is a full-time developer at Facebook and is also very active on Github. His guide has many useful tips for the different categories of programming questions. This material allowed me to

recommend common operations and tricks students should be familiar with. The handbook also refers to various Leetcode problems for each category, which I refer to as well.

Woo, Vincent. “CoderPad.” *CoderPad*, https://coderpad.io/.
CoderPad is a very successful online company that allows interviewers to customize the interview easily in terms of coding language and execution ability. The most common difference between normal coding and certain coding interviews is that the interviewee needs to code without syntax clues and debugging messages that arise from executing their code. Companies use sites like Coderpad to conduct interviews with candidates, so I inform the reader that practicing on this site helps interviewees get accustomed to the format of the real interview.