Coding Interview Study Guide

What To Expect

When it comes to tech interviews, there is a lot of variation in what you might encounter. In my personal experience, I've gone through every type of interview you could imagine:

- Solving plenty of coding problems
- Created an app over a week to send to the dev team for review
- Created a clone of Twitter in a weekend in a language I didn't know and presented it
- Given an IQ test as an initial applicant screening process (yes, seriously)
- Wasn't asked a single coding question and instead walked through my portfolio binder (this was where I landed my first real job!)
- Plenty more

While this might seem scary at first, try not to fret over how uncertain the interview process can be. The process might vary, but as long as you read through this guide, study everything for your appropriate level, and practice (and practice some more) you should do fine. And if after that you still don't pass the interview, just remember: tech interviewing **is a skill.** The more you do it the better you will get. And it only takes **one yes** to make it all worth it!

Company Size

One of the biggest determinants in the type of interview you can expect is largely based on company size. While this isn't an absolute rule, a good guideline to keep in mind is:

- The **smaller** the company, the more **quantity** matters (you can build)
- The **bigger** the company, the more **quality** matters (how you build)

Smaller Companies

One thing to understand is that the smaller the company you're interviewing with, the more likely you are to have a unique interview process. For example, I've interviewed at several smaller companies that have asked me to program an app for them and then walk them through it. This is because they need to make sure candidates can actually build things since their dev team is small and each developer is important to the team.

These types of interviews might seem off putting to a lot of people because you're being asked to do extra work and jump through hoops for an interview. While that's fair, I have actually always preferred these interviews because they allow me to demonstrate myself to someone.

As someone who didn't graduate from college, being able to show a company my abilities was something I had to rely on.

TIP Check out my course on building a portfolio for even more advice on this topic!

Regardless of your opinion on these types of interviews, just know they can occur and especially at smaller companies.

Medium to Large Companies

On the opposite side of things, the bigger the company, the more streamlined the interview process will be. This is largely because it isn't efficient enough to interview each applicant in depth and they need to create more formal processes. You aren't likely to see some unexpected interview process if you go apply to some medium/larger company (like Walmart, Chick Fil A, Toyota, etc).

You will probably be asked to solve some initial coding problem or given some automated tests at these types of companies. This initial test allows them to filter out a lot of candidates since they likely have more people applying than a smaller company.

Big Tech & FAANG*

Finally, there are the Big Tech/FAANG companies. These are major companies that are completely tech centric and typically based out of the Silicon Valley/San Francisco Bay area (like Google, Meta, Apple, Microsoft, Amazon, etc). These companies have an interview process unlike most others, and I've gone in depth about them at the bottom of this guide. Just know, landing a job at these companies takes a **lot** of work compared to normal companies.

Note: *Some* smaller companies in Silicon Valley might also use similar interview processes (Discord, DataBricks, even some startups). This is largely because of how mainstream the Big Tech interview process is out there.

*FAANG stands for (Facebook, Apple, Amazon, Netflix, Google). It's an older term used to describe those types of Big Tech companies. Since Facebook changed their name to Meta, this name isn't as valid, but it's still somewhat common.

Past Applicants

Now I know that all sounds a bit scary with how uncertain the interview process can be. But luckily, there's a huge tool at your disposal: past applicants!

Once you find out you've landed an interview with a company, the first thing I recommend you do is go to a site like Glassdoor and look them up. Glassdoor has great info about the company, salary details (which you should use when determining what salary to ask for/negotiate/etc), and more. My favorite feature they have though is reviews on the interview process. Anyone who has interviewed with the company can go and write a review about what it was like, whether or not they got the job offer or not, and their thoughts/advice.

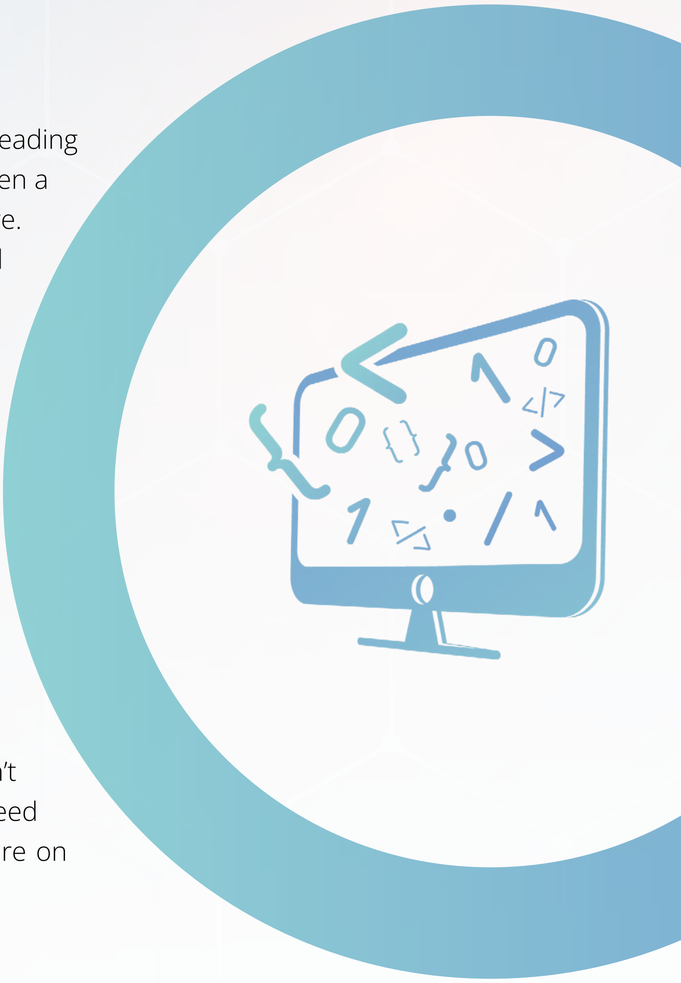
This is one of your **biggest** tools to use, so don't skip it. After all, it's basically someone giving you the secrets of what to expect and how to prepare for it.

A Quick Story

When I was interviewing at LinkedIn, I went to Glassdoor and began reading about previous interviews. I noticed a lot of Android applicants were given a blank Android application and were asked to add the same small feature. Knowing this, the day before my interview I opened up my computer and created a blank application. I then tried to implement the exact feature mentioned. It took a few Googles and slipping up here and there, but I got it working.

Then, I deleted the application and created a brand new one. I tried implementing it again and this time only made a few slip ups. I repeated this process **10 times** until I could do it with ease and not a single mistake. The following day I had my interview. As you can probably guess, they asked me to add that feature. As you can also guess, I aced that interview. 😊

This is not some unique occurrence I've had. In fact, I've done similar things for many past interviews I've had with equal success. You don't have to be the best **candidate** from a technical perspective. You just need to be the best **interviewer.** And in my experience, that part relies more on preparation and a friendly personality than anything technical.



Preparing

How To Prepare

For an in-depth look at how to prepare for technical interviews, be sure to check out my course on it. For an abbreviated version though:

- Go through the concepts in this study guide and make sure you are familiar with the things at your appropriate level
- Coding interviews are typically done without the help of coding tools. You usually have to write code by hand on either a whiteboard or on something like Google Docs. Since you know that is something to expect, make sure to practice doing it! Coding by hand is something you can get good at doing, but don't expect you will be good at it without ever practicing. It's much different than when you have nice tools like autocomplete and error highlighting
- Repeat, repeat, repeat. Practice really makes perfect. If you feel like you are struggling in certain areas, make sure to spend extra time trying to learn them better
- Take a look at my Interview Prep course for Non Technical things as well. You might be surprised to know that a massive part of passing your interview has nothing to do with how well you code! The soft skills matter just as much (and sometimes even more)

What Do I Need to Know?

As you've probably gathered by now, tech interviews vary a lot which can make it difficult to give you a straightforward answer on what you should and shouldn't know. That being said, there is something to keep in mind depending on your level of expertise (and it is nearly identical to what we discussed in company sizes):

- The more **junior** you are, the more **quantity** matters (you can build)
- The more **senior** you are, the more **quality** matters (how you build)

As you might imagine, people are going to expect someone junior to know much less than someone who has been in the field a few years longer. The newer you are to programming, the less you need to worry about studying a lot of different Computer Science concepts. Your focus is mainly on learning to build as much as you can and building up a portfolio. There are a few things I've highlighted for you to study below, but as you'll see, it is much less than someone who has been in the field a bit.

As you get more senior, these Computer Science concepts start to matter more. You're going to be expected to write higher quality code, and most of the advanced topics below are used to help optimize algorithms in certain ways.

So while I can't guarantee what you'll need to know and not know, I've used my best judgment (based on quite a bit of experience) for determining what you should study. I've marked each topic one of 3 colors depending on your seniority level:

🟢	**Green**	This is something you should probably know and might be asked in an interview
🟡	**Yellow**	I'd be surprised if you needed to know this for an interview. If you have extra prep time, you can spend some time learning these, but I wouldn't make them a priority
🔴	**Red**	You shouldn't need to know this. If you're more junior, please don't waste time trying to learn things marked red. It's not needed and will probably overwhelm you

Is This Practical Info?

I'll be pretty candid with you: a lot of concepts you might need to know to pass coding interviews aren't really used in most jobs. This is going to become more true as you start getting more senior and learning higher level concepts. For example, a senior with 5+ years of experience might be expected to know what a Binary Tree is, but in reality, the vast majority of programmers out there won't ever actually use one.

I'm not a fan of telling people they need to know something if it doesn't serve them much value in practice. At the same time, I don't want to tell you that you can disregard a topic and then have you get asked about it in an interview.

So in order to be straight with you while also making sure you're prepared, in each topic below, I've noted whether it is something that will be practical for you or just needed for interviewing. Obviously I don't own a crystal ball so I can't tell you my estimates are 100%, but I've gone through quite a lot of interviews at quite a lot of companies, so I feel pretty confident in my general estimates.

So with that said, I've used 3 descriptors for each topic:

Practical	I think you might use this in normal day to day programming
Interview	I don't think you'll use this much (if at all) outside of interview questions
Interview unless doing heavy data processing	I don't think you'll use this much outside of interview questions unless your job requires you to work with and manipulate large amounts of data (Example: Working on Google's search engine).

Category Types

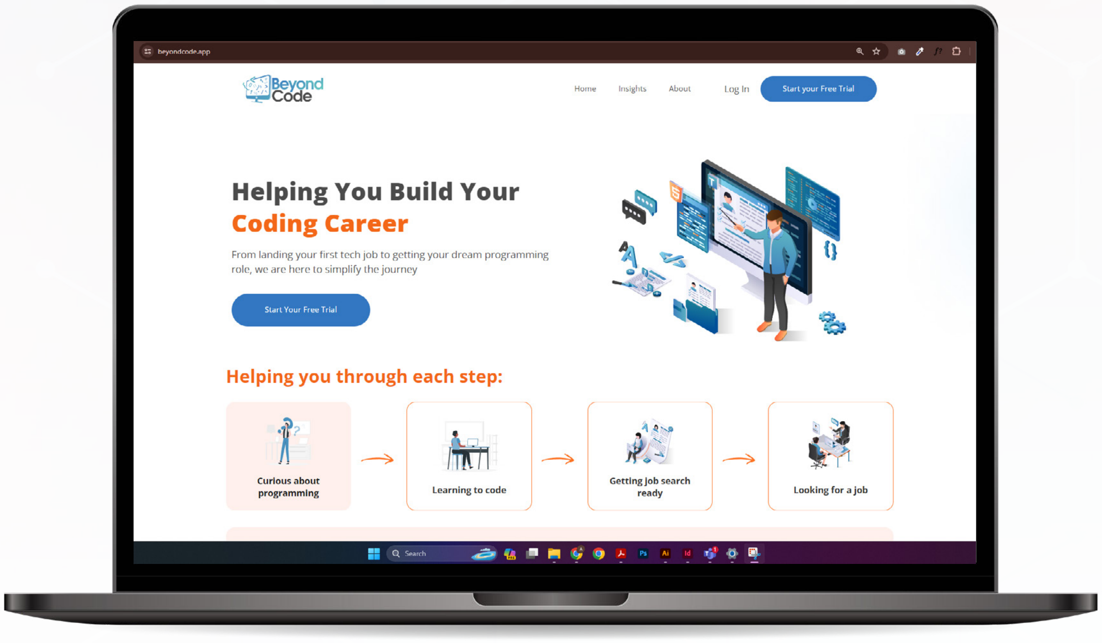
Lastly, I wanted a way to present the things you need to know depending on your seniority. I decided to break up the checklist into 4 different categories: Beginner, Intermediate, Senior, and Big Tech. Read the descriptions and see which you should go by when using the checklist.

Beginner	You are someone new to programming and have between 0-2 years of experience.
Intermediate	You've got experience working as a developer for about 2-4 years.
Senior	You've been in tech a while: 5-10+ years
Big Tech/ FAANG	You are hoping to interview and land a job at a major tech company like Google, Meta, Microsoft, Amazon, Netflix, etc. I've included an extra section at the bottom of this guide to go more in depth on this process.

	BEGINNER	INTERMEDIATE	SENIOR	BIG TECH/FAANG
ALGORITHMS & BIG-O				
What is an Algorithm?	☐ Practical	☐ Practical	☐ Practical	☐ Practical
Common Coding Problem Algorithms	☐ Interview	☐ Practical	☐ Practical	☐ Practical
What is Big-O	☐ Interview	☐ Interview	☐ Interview unless doing heavy data processing	☐ Interview unless doing heavy data processing
Calculate Time Complexity		☐ Interview	☐ Interview unless doing heavy data processing	☐ Interview unless doing heavy data processing
Calculate Space Complexity		☐ Interview	☐ Interview unless doing heavy data processing	☐ Interview unless doing heavy data processing
DATA STRUCTURES				
Arrays	☐ Practical	☐ Practical	☐ Practical	☐ Practical
Map	☐ Practical	☐ Practical	☐ Practical	☐ Practical
HashMap	☐ Practical	☐ Practical	☐ Practical	☐ Practical
Stack	☐ Interview	☐ Practical	☐ Practical	☐ Practical
Queue	☐ Interview	☐ Practical	☐ Practical	☐ Practical
Set	☐ Interview	☐ Practical	☐ Practical	☐ Practical
HashSet		☐ Practical	☐ Practical	☐ Practical

DESIGN PATTERNS				
What are Design Patterns?	☐ Practical	☐ Practical	☐ Practical	☐ Practical
SORTING & SEARCHING				
What is sorting	☐ Practical	☐ Practical	☐ Practical	☐ Practical
Know 1-2 Sorting	☐ Interview	☐ Interview	☐ Interview unless	☐ Interview unless
Algorithms			☐ Doing heavy data processing	☐ Doing heavy data processing
Write a Sorting Algorithm		☐ Interview	☐ Interview unless doing heavy data processing	☐ Interview unless doing heavy data processing
What is Searching	☐ Practical	☐ Practical	☐ Practical	☐ Practical
Linear Search	☐ Practical	☐ Practical	☐ Practical	☐ Practical
Binary Search	☐ Practical	☐ Practical	☐ Practical	☐ Practical
ADVANCED CONCEPTS				
Hashing		☐ Practical	☐ Practical	☐ Practical
Linked List		☐ Interview	☐ Interview unless doing heavy data processing	☐ Interview unless doing heavy data processing
Graphs		☐ Interview	☐ Interview unless doing heavy data processing	☐ Interview unless doing heavy data processing
Trees		☐ Interview	☐ Interview unless doing heavy data processing	☐ Interview unless doing heavy data processing
Binary Trees		☐ Interview	☐ Interview unless doing heavy data processing	☐ Interview unless doing heavy data processing
Binary Search Trees		☐ Interview	☐ Interview unless doing heavy data processing	☐ Interview unless doing heavy data processing
Tries			☐ Interview unless doing heavy data processing	☐ Interview unless doing heavy data processing

Heaps	🟥	🟥	☐ Interview unless doing heavy data processing	☐ Interview unless doing heavy data processing
Recursion	🟥	☐ Interview	☐ Interview unless doing heavy data processing	☐ Interview unless doing heavy data processing
BFS	🟥	☐ Interview	☐ Interview unless doing heavy data processing	☐ Interview unless doing heavy data processing
DFS	🟥	☐ Interview	☐ Interview unless doing heavy data processing	☐ Interview unless doing heavy data processing
Memoization	🟥	🟥	☐ Interview unless doing heavy data processing	☐ Interview unless doing heavy data processing
Dynamic Programming	🟥	🟥	☐ Interview unless doing heavy data processing	☐ Interview unless doing heavy data processing



Helping You Build Your Coding Career

From landing your first tech job to getting your dream programming role, we are here to simplify the journey

Helping you through each step:

Interviewing in FAANG/Big Tech?

The Process

Interviewing with major tech companies (such as Google, Meta, Netflix, Microsoft, etc) is quite different from small and medium sized companies. While there can be some variation, most Big Tech companies have the same general process:

01. You get in touch with their recruiter. They typically have a ~30 minute phone call with you to make sure they think you'd be a good candidate.
02. If the recruiter thinks you'd be a good candidate, they'll have you do one of the following:
    a. Have a "phone screen" interview with a developer. This typically involves a coding challenge that they watch you complete on Zoom. You will be asked to use something like Google Docs to type out your answer (no help from an IDE or anything).
    b. Take an automated coding test. This is usually timed and has several coding challenges you need to complete.
03. If you pass your phone screen, you have your "On Site" Interviews. Traditionally the company would fly you out to their office for this part, but ever since Covid this has largely been done online in Zoom calls. This On Site is a full day of (usually) 5 interviews. The exact types of interviews can vary, but in general, it will look something like:
    a. A general Algorithm/Coding Problem
    b. A specialized platform interview (Android, iOS, Web, etc. Whichever you're interviewing for)
    c. A System Design Interview where you are asked to whiteboard/draw how you would design a large scaled system/application
    d. Another Coding interview (could either be generic or platform specific)
    e. A manager interview where you talk with who would be your potential manager to make sure you'd be a good fit. This is typically non tech related and more focused on your personality/soft skills.
04. After your On Site, each interviewer will rate you and will come together to make a decision on whether or not you passed.
05. If you passed, your recruiter will contact you with their offer

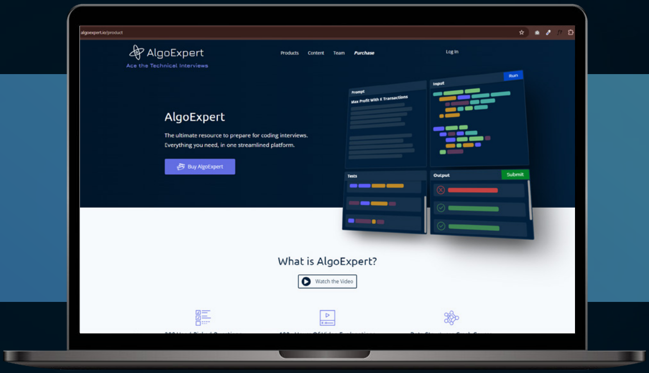
Yes it's a lot

Studying

As you can see in the study guide above, Big Tech interviews put a lot more focus on higher level computer science concepts than normal interviews. While this is a pretty debatable topic on whether that makes sense, it's the reality at nearly all Big Tech companies. So with that in mind, if you want to go into Big Tech, you will probably be spending quite a lot of time learning how to pass the interview process.

From personal experience, I spent the first 5 years of my career at small and medium sized companies. When I decided to try for Big Tech, I lined up interviews at Google, LinkedIn, Facebook, and Amazon. It took me almost 2 months of hard studying to finally be ready to pass those interviews. Even having been a developer for 5 years, the Computer Science concepts you need to know are things most people never actually encounter in the real world.

There are two main resources I recommend:


AlgoExpert
Ace the Technical Interviews
Products
Content
Team
Purchase
Log In
AlgoExpert
The ultimate resource to prepare for coding interviews.
Everything you need, in one streamlined platform.
Buy AlgoExpert
What is AlgoExpert?
Watch the Video

https://www.algoexpert.io/ - This is an amazing resource for learning different algorithms and data structures in depth. I believe it cost me around $100, but it was well worth the investment. It's the main reason I was able to pass my Big Tech Interviews.

https://leetcode.com/ - A very popular site for trying coding problems. It doesn't have as much in the teaching aspect as AlgoExpert, but it has a lot more problems you can practice with.

CRACKING the CODING INTERVIEW
189 Programming Questions & Solutions
GAYLE LAAKMANN MCDOWELL
6th EDITION

Cracking the Coding Interview -
This book is a classic when it comes to Big Tech interviews

Some great videos for what to expect in the Big Tech Interviews

1. Google Coding Interview With A Normal Software Engineer
2. Google Coding Interview With A Facebook Software Engineer
3. Easy Google Coding Interview With Ben Awad
4. Google Coding Interview With A College Student
5. Google Coding Interview With A Competitive Programmer
6. Google Coding Interview With A High School Student