Mehul Chaturvedi

267 Followers

Interview preparation roadmap that got me into Amazon

Mehul Chaturvedi Sep 2 · 15 min read

Source : press.aboutamazon.com

Over the past months, many of my friends and my juniors have asked me about my preparation journey, my interview experiences, and how I was finally able to crack the interview of Amazon for the role of a Software Development Engineer.

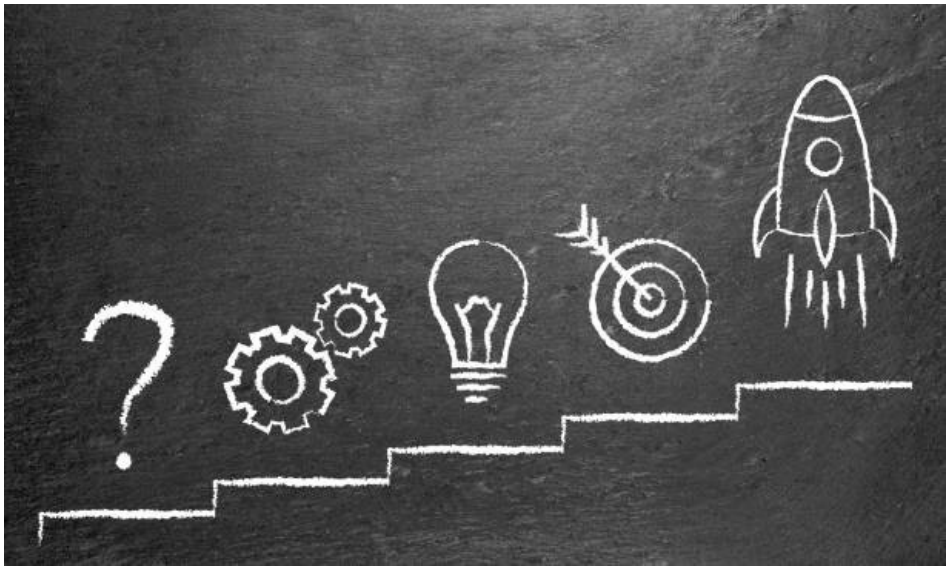
So here I am, finally bringing you the roadmap for the preparation for your dream company. I'll try to keep it short and to the point, and will try not to bore you guys with irrelevant talks, will try to cover everything without assuming anyone's background, so even if you have zero coding experience before, after reading this article, you'll be able to create a clear mind-map to proceed the journey to your dream company.

. . .

How it all started

First of all, to introduce myself, I did my Bachelors from IIT Guwahati (2021) with a major in Civil Engineering (Yes, you read the last two words right!), and I got placed in Amazon as a Software Developer in December 2020.

I started my preparation in the mid of my second year since I wasn't sure about what to pursue as a career. I'll try to break those 1.5 years into different stages for a better understanding. I highly recommend the readers to set aside 15 minutes for this article and read it till the end.



Source: iStock

. . .

Preparation strategy

First of all, forget all the worries and make the path below clear in your mind, and start following it step by step.

Basics of C++/Java + OOPS → Data Structures and Algorithms → Web-development skills and projects → InterviewBit → Competitive programming (optional) → CS fundamentals → Leetcode → Brushing up

Stage 0: Points to follow along with each stage in the preparation

Follow the below points from the Day 0 of your preparation. No matter what, these points are vital. I cannot stress more on them:

- **Make notes in a notebook**: Handwritten notes always make you understand better and are best for revision. You'll realize the importance of them a month before the interviews.
- **Write what is important:** Now, do not just start writing everything you read or every question you solve. Just write down stuff you feel is very new to you or difficult to understand/remember. You can also write down solutions to problems you spent a lot of time and learned something beautiful.
- **Do not skip anything:** Try to understand why you did something that way and not the other way. How would you know what problems should be solved using this method, why this problem is solved using this method, try to understand the math behind the problems. Try to understand the core behind all the things you learn, and don't just ignore/cram up something without understanding it.
- **Think before you code:** Never start coding without having the logic clear in your mind. First, read the problem, try to understand what all is given to you and what the problem wants from you. Then try to figure out the possible solution to the problem in a notebook. And when you are somewhat sure about the logic, then start coding. This will not only improve your accuracy but also speed over time.
- **Accuracy >>>>> Speed:** Believe it or not, accuracy is the most important thing while learning any new skill. Speed comes with practice. Try to create an analogy

with learning to drive a car. You do not just start riding it at 100kmph. You first go it as slow as possible so that you get the car in your control. First, you try to understand it, and after some time, you can drive it without putting much stress on your mind. It all just happens automatically, and your legs develop the muscle memory to switch between clutch and brake, etc., etc. And now you are automatically able to drive it at 100kmph.

- **Learn from your mistakes:** Do not feel low when you make mistakes or say when you're not able to solve some problem. Be happy at that moment, and give it a thought that the problems that you were able to solve already did not contribute much to your learning. They just improved your speed and gave you confidence (which is also important). Now you finally got something new to learn, so now give your best to solve that problem, read editorials, watch some video solutions, ask someone, but in the end, make sure that you know all the concepts related to that problem. Next, time if something like that comes up, you should be able to solve it. That is how you'll progress.
- **Have a coding buddy:** Have a coding buddy with whom you can discuss doubts, and you both guys can track each other's progress along this journey.
- **Speak when you think:** Try to speak to yourselves while solving a problem (nobody will call you mad). This habit will be beneficial during virtual interviews because this way, the interviewer will be able to understand your thought process better and will be able to judge you better, even if you were able to solve a question only partially.

Stage 1: Basic knowledge of a language

Now the question comes, how do I gain the basic knowledge of some language?

Well, there are multiple options for all the people, and you may choose according to your ease:

1. Suppose you do not want to spend any money on courses from different paid websites, no worries. There are tons of good videos available on YouTube, and they are more than enough. Follow the below list in order:

- **C++ playlist by CodeWithHarry**: This covers everything in C++ that you need to kickstart your interview preparation journey. It also covers some concepts of OOPS, which is really good.
- **DS Algo Playlist**: This one also covers the basics of C++ language but also covers many data structures and algorithms, after covering which a person could become independent and move to the next steps of the journey. After covering this, you'll have an excellent idea of programming, and you'll be able to solve many

easy/medium problems on your own, or at least you will be able to understand their solutions.

If you are following **YouTube**, keep in mind that you need to do code-along with the videos, when the YouTuber is solving a problem, pause the video after the problem statement and first try to code on your own, and give at least 15–30 minutes to the problem if you are not able to get any thought for the solution, before watching the solution from the video.

2. If you have a budget and want everything in one place, **Coding Ninjas** would be the best for you.

- **C++ Data Structures and Algorithms**: This one course will cover almost everything, including C++ syntax, data structures and algorithms, and also OOPS. It is well crafted to give you the content and practice all in one place. This will set up an excellent base to move on to the next steps without any worries. They also have a doubt resolution system, which is really helpful, and you get a TA assigned almost instantly for your doubt, which in my opinion, is an excellent feature. There are many pros of this course; I am not going to list all of them here. You can read them on their website.
- **Competitive programming**: Optionally, if you have some time and have some extra budget, you can also take this course for Competitive programming. I took both of these courses. But yes, the cost was pretty low during my preparation time.

Stage 2: Web Development Projects

Now, as you have gained some good knowledge of DS-Algo and have set up a good base, you should now create some projects that will help you develop **your CV**.

Udemy is the best place for it, given that the courses are project-based, very well explained, and are pretty cheap as well. I also liked the doubt resolution system of Udemy, you get a reply in less than a day for any of your doubts, and the courses are also super updated. You can choose any stack for development and take the relevant courses for the same, I'd personally suggest the below courses in order, although if you have time, you can take some extra courses according to your interest, the ones below are the ones which I'd strongly suggest you complete:

- **The web developer Bootcamp**: Colt is the best instructor. This guy explains everything that way, so you get it instantly. This course covers almost everything that you need to create your own full-stack website, it does not assume any

background of the student, and even a guy with zero knowledge of development can benefit from the course. This course has some cool projects on the way, which will give you a good idea of how to build real-life applications. Using all this knowledge, you can create some really good projects to add to your CV.

- **React — The Complete Guide**: A fantastic course to start building React applications. The instructor is extremely great. He explains everything in different ways and shows us all the possible ways to work with React. After doing this course, you'll become an expert in React as it covers almost everything and in an adorable way. You can do some projects using the knowledge of this course too, which you can add to your CV.

For project ideas, you can do a simple Google search, there are tons of ideas available you can pick any of the ideas and build a project around that, and it'd be a cherry on the top if you are providing some use cases in the real world from your projects so think out of the box, be creative.

Stage 3: InterviewBit

By now, your CV must be having some good projects, and now you need not worry about your CV much, now all you need to do is practice, practice, and practice.

- Start a new session on InteviewBit and devote the next **two months** entirely to it. I cannot stress more than how important the programming section of InterviewBit is. It is like a bridge you ought to cross to clear the interviews.
- So, start with the first problem of the first section and do it till the end. Do NOT skip a single problem. Try to understand each and every problem to the core. You should know the solution so well that you'd be able to explain it to anybody.
- Do not forget to take notes.
- Experience on this platform is closest to an actual interview. Sometimes your code passes all test cases but might not be time/space optimal (as required during an actual interview). InterviewBit reports these submissions as suboptimal, providing you additional feedback.
- Some problems might take a lot of time, that is entirely ok, it happens to all, slowly and gradually you'll be able to solve such problems faster.
- Also, the streak feature of this platform keeps the motivation to solve problems up.

Stage 4: Codeforces (Optional)

- If you have time and have started enjoying solving coding problems, then codeforces

will boost your learning curve and help you become a better and fast problem solver.

- You can start giving contests on Codeforces. You can start with Div-3 rounds. After every contest, try to analyze your performance, always try to upsolve one extra problem. Let us say you were able to solve A, B, C during the contest, then try to solve D after the contest, give some time to it, read the editorial, ask someone, watch a YouTube video solution because that is how you'll progress.
- Do not worry about the rating. Just keep on learning from your mistakes. Ratings will improve automatically.
- I'd suggest you at least give regular contests on Codeforces if not solving daily from the problem set because that helps to get used to the coding test/Interview pressure. It also improves your speed and accuracy a lot. The problems here are not repetitive, so you develop a good problem-solving ability, which is really helpful during interviews.
- Suppose you also want to solve more problems from Codeforces. In that case, I'd suggest figure out first which rating problems you are not much comfortable solving, then get all the problems of that rating, using the problem rating filter on Codeforces problem set section, and start solving from top to bottom till you get comfortable on that level and keep on increasing the difficulty level that way.
- Also, turn off the problems tags. This technique proved to be really helpful for me.
- You can also get a list of some good problems that Codeforces has on the platform called A2oJ. It has various ladders for different levels.

Stage 5: CS Fundamentals

It doesn't matter what your background is. Whether you are from a CS background or not, you should know some of the CS fundamentals to be able to clear tests or interviews. You need not know everything. There are a few topics listed below according to the priority that you must know:

- **OOPS**: This is really, really, very important. You should know each and everything in OOPS and in such a way that no matter what question the interviewer asks you, you should be able to answer it. You must have already covered it in the previous stages, but still, to brush up, you can go to GFG and read everything from there.
- **OS and DBMS:** A lot of questions get asked from both topics in tests, as well as interviews, you need not know everything regarding them, but you should have a good understanding of some crucial subtopics of the same. You can go to GFG for the same, and that should be enough for the interviews. If you have some time, you can watch some playlists which are available on YouTube. They cover everything in depth.
- **Networking:** You can find some really good playlists on YouTube for the same and read more about it on GFG. This topic is not asked as much as the topics above are asked, so divide your time accordingly.

Although you must have understood these topics to a good extent by now, to gain some confidence and to know what type of questions exactly gets asked in tests and interviews, you can do a simple Google search for each topic like, **"Top OOPS questions asked in interviews"** then pick some top results from Google search and make sure that you can at least answer each of them, if not then go back and read about that particular

subtopic again. **GFG** has **short notes** available for all these topics so that you can read them before your interviews for some quick revision. This will give you a complete finish to your CS Fundamentals preparation.

Stage 6: Leetcode

Now comes the second most important step after InterviewBit. Believe it or not, Leetcode is enough for cracking interviews with companies like Amazon, Google, and Microsoft. InterviewBit is a subset of Leetcode, so this is why I asked you to do InterviewBit first because InterviewBit has the collection of the most important questions of Leetcode, these questions are asked more frequently. Now below are the points that you should follow while doing Leetcode:

- I'd suggest you start with the 100 most liked problems on Leetcode and solve them from top to bottom.
- After that, you can start solving problems, pick a topic, select medium-level problems, sort them by the number of likes, solve them from top to bottom. And when you feel confident enough in topics, you can turn the tags off and start solving random problems.
- Always give at least 30–45 minutes of thought before looking into the solution.
- Try to solve at least four medium problems each day, and you can try hard problems when you feel confident enough with medium problems. You can also keep it like, say I'll solve 1–2 hard problems in 2–3 days and increase that frequency with experience and your skills. Although, medium-level problems are enough for most interviews.
- Keep a target of around 400 problems on Leetcode before appearing for coding tests or interviews (these do not include easy-level problems).

Source: imgflip

Below are some tips which I found in some articles on Leetcode, and I found them really useful:

- If you feel weaker in a certain algorithm, you can filter the problem set by "Tag." This is NOT recommended for general practice since much of the actual problem-solving skill you want to have the ability to identify the type of algorithm to solve a problem. So, if you have filtered by "Binary Search," you know the solution to the problem will probably be binary search.
- Go through some older problems, and make sure that you have found the optimal solution. Often, the LeetCode online judge will accept suboptimal solutions. If an

O(N) solution exists, but you submitted an O(N log N) solution, most likely, it's going to pass still. The percentiles for runtime/memory are actually a bit misleading, so don't worry too much about that. The only time that it kind of help is if there are two very different runtimes, like O(N) vs. $O(N^2)$, and the runtime distribution will look bimodal.

- The weekly contests are a great way to see where you stack up against the rest of the community. Plus, they apply some time pressure and usually give new problems that you haven't seen before. (I'd highly recommend you to give these weekly contests, they are easier than Codeforces, but yeah, they train you better for interviews and OTs)

- This is new and not available several years ago, but the new Mock Interview is good for adding some time pressure. Unfortunately, there is no way to filter out previously solved problems, so you might get repeats of ones you've done before — but you can view this as an opportunity to revisit some of the older problems. This can also help you identify areas of weakness.

BONUS TIPS

- **Cultivate your LinkedIn profile**: keep it up to date and be professional. Try to avoid posting spam or "please follow me" kind of content. Provide value in the form of articles, videos, or even just hints, and people will want to follow.

- **Prepare good questions:** I'd argue that half of my interview was won because I asked the right questions, like "What would you change in your team if you could?" or "What bothers you the most working for Amazon?". Those kinds of questions not only open a dialogue but also provide the opportunity for the interviewer to vent a little. That's a good feeling. You want to leave your interviewer feeling good…

- **Give a couple of mock interviews** before the actual interview, you can ask your friends for the same, or there are multiple resources available online too.

- **Do the naming right**: During your interview's coding round, make sure that you name the variables such that they depict a clear meaning and are not some random letters like 'a', 'x', 'p'… etc. Make the code easy to understand to the interviewer, write clean code. Double-check the corner cases, it is very important.

- **Read Interview Experiences**: Make sure that you always read as much as interview experiences as possible before giving the actual interview of that particular company, sort them by date(latest first), and start reading them thoroughly one by

one. Do not miss a single question from them at least. You should be able to solve/answer all the questions in those interview experiences.

- **Top Questions**: You should also read **"Top X company questions"** before giving the interview of that company X.
- **Prepare for the HR round:** Do not underestimate the HR round. It is very important to prepare well for it, bring real-life stories to answers, and try to bring the tech side/education side to the highlight, rather than something philosophical. If you prepare for some well-known HR questions, you'll be able to answer almost any HR question very well, as all of them are almost similar to a smaller domain of questions. So structure well the answers of that smaller domain, which you can easily get by some Google searches.
- **Keep learning**: Do not lose hope. Keep giving your best, learn from your mistakes, and one day you'll surely get into your dream company.

P.S. If you feel any queries are left unanswered, drop me a message on your LinkedIn profile or email me at *mehul355180@gmail.com*.

. . .

Learn More

Get a Free SSL Certificate From AWS
How to Get an SSL Certificate From Amazon Certificate Manager (ACM)
enlear.academy

AWS EBS vs EFS vs S3
How to Choose the Best AWS Storage Option for Your Project
enlear.academy

How Does Blockchain Work?
All you need to know about blockchain.
enlear.academy

Amazon | Interview | Software Development | Roadmaps | Faang