The One year plan for competitive coding

Anjali Viramgama Follow
Jul 13 · 5 min read ★

About my hustle before cracking interviews.

It took me one year to go from a noob programmer to someone decent enough to crack coding interviews for getting internships and gaining experience. I still have a long way to go, but the first step to being a good programmer is working in the real world and getting experience, which can be best gained by internships. And if you want an internship, you have to crack the interview first. Which brings us to this blog.

I have broken down my one-year plan, which I diligently followed, and will hopefully help you with your planning if you are in the starting stage.

Prerequisite: Knowing the basics and syntax of one programming language. Most students tend to know Java, C, or Python from their colleges/highschools. You can stick to the one you are comfortable with from these three, but if C is your preferred language, I would recommend you to switch to C++. My first language was C, which made me switch to C++. I learned Java on the side, enjoyed it more, and decided to practice competitive coding in Java, and so every interview I have ever cracked was by using Java. I had zero experience in python, but after joining Facebook, all of the code I have written as an intern is in Python. So my point is, there is no superior language amongst these three, try not to worry about which one to choose. Just pick one, crack interviews in that one, and you can learn the rest on the go depending on where you get placed.

Here's the plan:

Month 1: Big O, Arrays and Strings

Month 2: Linked Lists

Month 3: Stacks and Queues

Month 4: Trees

Month 5: Hashmap, Dictionary, HashSet

Month 5: Graphs

Month 6: Recursion and Dynamic Programming

Month 7: Sorting and Searching

Month 8: Reading(about system design, scalability, PM questions, OS, threads, locks, security basics, garbage collection, etc. basically expanding your knowledge in whatever field required, depending on your target role)

Month 9, 10, 11, 12: A mix of medium and hard questions in your preferred website. Practice by participating in contests, focusing on topics that you are weak at, mock interviews, etc.

Notes:

- Here's how I approach every topic in each month — Let's say you are in month 4, and focusing on trees. You need to first understand what trees are, different types of trees, and be able to define class Node and Tree. You then need to be able to perform basic operations like adding, finding, and deleting an element, pre-order, in-order, post-order, and level-by-level traversal. Lastly, you practice different tree questions available on Hackerrank, Leetcode, or a website of your choice. You should target the easy questions first, and once you are comfortable, move on to medium and hard.
- The last 4 months are for solving a mix of different questions, via contests or otherwise, which is necessary because when you are practicing tree questions, you know you have to use a tree. But if you are given a random question, how will you know a tree would be the best approach? Also, always look for the most optimal solution in forums after solving it yourself.
- You have an entire month, and if you manage to dedicate 40–70 hours a week, you'll be able to master trees in such a way that if a tree question is thrown at you in an interview, you'll be able to mostly solve it since you trained your mind to think that way with intense practice. If you are a student, dedicating this much time is definitely doable, even with side projects, homework, etc. Your grades might take a hit (my As became Bs in that one semester(month 9,10, 11, 12) when I was dedicating over 8 hours a day to competitive coding) but it was worth it.
- You should also try to build projects or do research on the side while preparing.
- Some people learn better by participating in contests in CodeForces, CodeChef, etc. while others prefer practicing questions. Again, there is no benefit of one over the other, do what you personally prefer.
- I do not believe in practicing particular topics for a particular company, some websites claim to have a set of questions dedicated to a particular company, eg: cracking the Google interview. I think the goal should be to be a better developer overall, focusing on just a few topics that Google tends to test candidates on may not be the best way to follow.
- Interviewers also judge you based on your LinkedIn, Resume, past experiences, courses taken, Github, degrees and certifications, projects, research papers, etc. Practicing competitive coding does not guarantee a job, but it does guarantee you'll be able to crack technical interview rounds most of the time, and you'll also be a better developer overall, which might help you when you build projects.
- Lastly, don't stop. It may seem easy at first when you are motivated, but that fuel dies in a month or so. Keep your goal in mind, of course, it's going to be hard, but the only ones who make it are those who stick to the plan. You can edit the plan if you need to, but once done, stick to it, even on your lazy days, even when you have a college fest or a party to attend, even when you are sleepy. Like I said, the ones who succeed are the ones who "**stick to the plan**".

This sums up my schedule at a high level. I plan on digging deep, and my next blog will only focus on month 1 (Big O, Arrays and strings), the one after that will be month 2, and so on. I hope this was helpful, let me know if you want me to also write about any other topic on the side, or if you have any queries. I'd appreciate it if you could ask your questions on Instagram since I prefer to keep LinkedIn for professional opportunities, but either is fine.

Thanks! Signing off!

Anjali Viramgama

Software Developer Intern at Facebook

LinkedIn | Instagram

Competitive Programming Google Facebook Coding Technology