**Edbert Chan**

**Hours 1-35:** *Grokking the Coding Interview: Patterns for Coding Questions* (omit DP section, we will do that later). I recommend doing every single problem except for the challenges (not necessary). Devote at least 20 min to each problem before you look at the answer. After every problem, try to redo them from the ground up. Focus on a systematic approach to the problems. It might seem like a lot of initial investment of time in the first few problems but since the problems test the same technique, if you know how to practice right, you will be very fast at applying them over and over again so invest your time to doing the first problem RIGHT. If you do the first problem in the right way and think in the right way, the rest of the problems become a 5 min exercise.

**Hours 35-50:***Grokking Dynamic Programming Patterns for Coding Interviews* (separate course). This was the hardest for me to start but once I got going, it was fine. There is a systematic way to approach all these problems except for the substring/subsequence ones. I just memorized those.

**Hours 50-85:** *Elements of Programming Interview* (EPI). Do the problem set for the "project" turn around time (3 weeks). Personally, I did pseudo code for every question; I didn’t even write out the code because it was just far too time consuming and there wasn’t a way to check other than to actually write the test cases which took away time from actually studying.

For the recommended questions, I spent an average of 20 minutes trying to figure it out. For the non-recommended questions, I spent an average of 10. You can omit dynamic programming chapter. You really only need to cover chapter 5-16 for technical material. First 4 chapters are introductory.

**Hours 85-115:** System Design study (omit if you’re aiming for L3). This one is particularly murky for me since I’m a mobile developer and most of this time was me re-reading Android documentation, system design patterns, what is new with new Android versions, new components available, etc. I think *Grokking System Design* is good for most people who are going to be working backend or are full stack engineers. I bought the course but it wasn’t particularly relevant to me.

**Hours 115-130:** Leetcode. Fill in the gap. Do mostly medium level problems but also try random hard level problems. These will tell you where you're missing knowledge. I recommend sticking with the questions from the section for company interviews. This is your “ramp down” period.

**Note:**

You will feel like absolute shit for the first 50 hours and you should. But you'll get over it. Briefly re-skim through the problems and see if you remember how to solve them before moving on to the next chapter.

Also remember to spend a good day or 2 before your onsite interview to wind down and not focus so much on studying. Otherwise, you will go crazy and get paranoid about what information you may or may not have forgot.

If you don’t have as much time as I did, you can just do the *Grokking* series before your phone screen (without the DP portion) and finish up the rest prior to your onsite. If you find there is not enough time between now and your onsite, you can cut back on some of the EPI topics. Focus on arrays, lists, sorting, trees, graphs in *Grokking* and *EPI*. The bitwise section can be put for later (it is rare but not unheard of to ask bitwise operation problems). Reduce the problems you look at in EPI to just the essentials based on your timeline. They have a “schedule” of recommended problems depending on when your timeline is so adjust accordingly.