Entry-Level SWE Interviews

Connor Olson

Last updated 12/13/2025

Purpose

This document summarizes the knowledge I have collected interviewing for internships and entry-level roles in software engineering. You might be reading it if you asked me for advice on this topic. There is no advice here, but the content might help you help yourself. Good luck on your interview.

Types of Interviews

There is different kinds of technical interviews that evaluate you differently. Every company has different evaluation criteria and expects different things in the interview, but there are a few major styles. Some companies may use multiple.

DSA Interview

A data structures and algorithms or LeetCode question style interview consists of the interviewer describing a function and you implementing it in your language of choice, and is the most common kind of technical interview. You will be evaluated on (at least) the following criteria:

- Correctness (does your code produce the result intended by the question)
- Communication (do you explain your ideas and code to the interviewer, do you ask important questions and otherwise communicate effectively)
- Code Efficiency (is your code efficient in runtime, memory usage, and any other relevant objective metrics)
- Code Quality (is your code readable and easy to understand)
- Consideration, Carefulness (do you discuss and consider tradeoffs when writing your solution, do you discuss alternatives and how applications would change your approach)

Depending on the company and interviewer, these metrics will be weighted differently. Companies or interviewers will usually say which ones they care about most, but sometimes they do not, in which case it is important to ask.

LLD Interview

A low level design interview is very similar to a DSA interview, but instead of implementing a single function or algorithm, you are typically implement a class or classes, a set of connected functions, or implement some other relevant program or small system.

This interview is functionally the same as the DSA interview, but focuses more on design patterns and code quality. Interviewers typically do not care if your solution is efficient, and are looking for the solution which would be the easiest to understand, use, and maintain in practice.

Online Assessments

Online Assessments will take the form of a DSA or LLD interview, but there will be no interviewer and you will just be solving the problem by yourself. Certain companies often ask very similar questions, or give questions in the same category. I am sure you have done many of these. I have likely done over 100. They are incredibly depressing and act as some arbitrary filter to thin out the number of candidates. At some companies they can be incredibly easy (e.g. IBM) and at others incredibly hard (e.g. Snowflake). These typically have nothing to do with the job you are applying for or the courses you take in University. I believe this kind of screening is unethical. I have spent hundreds of hours on LeetCode preparing for these and still fail them regularly. Good Luck.

Competency Assessments

Although not a technical interview by definition, another common interview style I have often seen at companies that are not primarily tech companies, or start ups, is a normal conversational interview but with an engineer. The conversation will usually revolve around items on your resume, and the interviewer will probe you to give more details on things you have worked on. They may ask pointed technical questions to gauge how deep your understanding is of things you claim to know, or ask you general knowledge questions related to the particular role you are applying to.

Behavioral Interviews

Typically following a successful technical round, you will have a behavioral interview where the interviewer(s) assess whether they would like to work with you. You will be evaluated on alignment with company values and how easy you would be to work with.

This round is actually incredibly important and is a time for you to expand on your past experience and prove your candidacy. At mid-size to large companies, interviewers will be looking for concrete examples in stories you share that give certain signals (e.g. can this person learn quickly, do they resolve conflicts easily) usually in direct correspondence with company or team values. At the entry-level, most interviewers seem to be trying to assess whether you are interested and capable of learning a lot very quickly, and have good interpersonal skills for resolving conflicts and keeping your ego or opinions out of your work.

Preparation

Good interviews will effectively evaluate if you are good at the job, but entry-level technical interviews tend not to do that. You need to meet specific requirements and send certain signals under

stress in an artificial scenario that might have very little to do with your actual job. Preparation for technical interviews is a massive undertaking, after having spent hundreds of hours preparing, I can confidently say that I am still not very good at them, even though I believe that I am in fact good at my job.

Before beginning the long journey of interview preparation for SWE, one should ask themselves what they want to get out of it. I choose to prepare in ways that will also actually make me better at my job, not just better at interviews.

DSA Prep

Most of my preparation time is spent on LeetCode. These interviews are the most difficult (at some companies) and have the highest bar, even though they also have the least correlation to the job. I prepare every week by doing:

- Targeted practice on LeetCode.

  This consists of learning new data structures and algorithms (the NeetCode list is a good place to start), participating in coding competitions (for me, the LeetCode weekly and biweekly competitions), and practicing questions.

  I *do not* try to memorize questions or solution patterns. Although this technique works up to a certain point, I do not think it is ultimately making anyone a better engineer. I enjoy actually learning and solving very difficult problems, doing better in competitions, and having a deep understanding of the theory and practical implications of problems I am solving. For me, it is easier to be consistent doing this kind of practice in a way that I enjoy and actually feels rewarding.

  There is many conflicting ideas of how to practice LeetCode. I do not think it matters, but I do think that you should not say that you have learned a problem until you can solve it on paper while explaining it out loud with someone watching you and not having looked at the problem in a week. How you reach that point is up to you.

- Mock interviews.

  Being able to solve a problem by yourself in an IDE is completely different than doing it under a time constraint with no syntax highlighting while someone is watching and evaluating you, and explaining yourself at the same time. When I did my first interviews, I was asked easy questions that I knew, but I could not effectively articulate myself while solving them. That did not end well. I find both recording myself and actually doing peer mock interviews to be effective for this purpose.

Behavioral Prep

I attribute multiple offers I have received to targeted practice for behavioral interviews. This video (https://www.youtube.com/watch?v=bBvPQZmPXwQ) is a good starting point to understand how they work and how to prepare.

My checklist for a behavioral interview is something like this:

- Can I explain every word on my resume in great detail?

- Do I have stories ready to convince my interviewer that I am easy to work with, communicate effectively, can learn quickly, am excited about my work, and personally align with company values?
- Do I understand what is expected of me in the job I am interviewing for, and what the company does?

Practical Wisdom

Interviewing is hard. Here are some things I do to boost my chances of succeeding:

- Asking questions in DSA and LLD rounds.

  I ask a lot of questions to interviewers in coding rounds. Some example questions:

  - I am planning on doing XYZ to solve this problem, do you think that is reasonable?

    This question allows the interviewer to steer you in a better direction if you are going on the wrong track, without making it seem like you have no idea what is going on.
  - I'm understanding the problem like ABC, does that sound right?

    Same as above, I want to be sure I am solving the right problem.
  - I'm not sure about QRS, can you explain that?

    It is actually a positive signal to clear up ambiguities in the problem statement or things you didn't understand. I always ask.

- Using strategies to save time and face in DSA and LLD rounds.

  There are some "tricks" you can use to make solving technical questions easier. Here are some that I use:

  - I often say "here I would do X, but I will come back to that later as it is not very interesting, unless you want me to implement it now."

    If you are about to implement sorting, write boilerplate code, or do something that is not super relevant to the problem in order to solve it, you might be able to just hand wave it with a comment and skip over it. This can let you save time.
  - I am always doing one of a few things:
    * Explaining code I am about to write.
    * Writing code I just explained.
    * Asking a question to clear up my understanding.
    * Explaining why I am stuck.

    This ensures I am always making progress in some form or giving a good signal.

Finally, although consuming resources is good to get a basic foundation for how things, you will learn a lot more by actively practicing than listening to others advice. I find this to be true for all kinds of problems, and learning in general. Good luck! We are all in this together.