There are 3 types of interviews for a SDE Role. They are:

1. Coding Interview
2. Design/Architectural Interview
3. Behavioural Interview

**Coding Interview Tips**

**ABC (Always Be Coding)**

The more code you write, the more prepared you’ll be for your technical interviews — it really is that simple. The key here is to make sure you’ve spent at least 4x as much time preparing for a coding interview as you think you’ll spend coding. In other words, if you expect a single 60-minute interview writing code, then you should spend 4 hours practicing.

**Be comfortable with at least one dynamic programming language**

Dynamic programming languages like Python and JavaScript are both well-known and conducive to writing short, powerful programs with less boilerplate code than found in other languages like Java or C++. Coding problems don’t typically involve writing a lot of code, and it’s most important that your solution is correct, clean, and performant (often in that order). Using a language that allows you to focus on solving the problem rather than wrestling with the compiler or abstractions is often the best way to go.

**Know thy complexities (Focus on Time and Space Complexities of Algorithms)**

Cheat Sheet - <https://www.bigocheatsheet.com/>

Target – 80% of the above cheat sheet

“Maybe you haven’t ever had to implement a **red-black tree**, but you should understand when and how it’s useful in various applications. When presenting a solution, interviewers will often follow-up with: “**What’s the time complexity of your solution**?” Be prepared to speak to it.”

**Reinvent the wheel**

Practice by **implementing the most common data structures in your language of choice**. Do not rely on common libraries. Implement the following and write tests for them: **vector (dynamic array), linked list, stack, queue, circular queue, hash map, set, priority queue, binary search tree**, etc. Interviewers won’t typically ask you to do this directly. They will usually take the form of a word problem or some other abstraction (though I was once asked to pair program a circular queue with my interviewer).

**Solve word problems**

This is one of my favourites and one of the most important activities you can do. Interviewers will often present problems that seem complex but can be reduced down to simple algorithmic solutions. First, read about [competitive programming](https://www.topcoder.com/community/competitive-programming/tutorials/). Make sure you cover your bases by solving problems that use recursion, pattern-matching, greedy/dynamic programming, and graph-based problems. Solve a few in each category using your dynamic programming language of choice. You can find sample problems on sites like [HackerRank](https://www.hackerrank.com/dashboard) and [TopCoder](https://www.topcoder.com/tc?module=ProblemArchive&sc=2&sd=desc).

**Additional Tips**

During the interview, make sure you are **clear on the problem** you’re being asked to solve. Sometimes interviewers won’t give you all of the information required for the optimal solution and expect you to **ask clarifying questions**. While that’s more common in design/architecture interviews, you should be prepared to make sure you’re **approaching the problem correctly** before attempting to solve it. When you do start to work on it, focus on solving it first and optimizing it later. **Correctness counts the most, and getting to a suboptimal solution is better than failing to produce a performant one**.

After you solve the problem, **be prepared for follow-up questions** that may require you to write more code. When a problem seems straightforward, you might anticipate there will be follow-up questions that constrain or complicate the problem somehow.

**Design/Architecture Interviews (Not important as of now)**

**Behavioural/Experience Interview Tips**

* Generally, less technical
* Used to assess culture fit, work experience, background, and how you handle specific situations

When answering behavioral questions, consider first explaining how you think about it from **a conceptual level**. This helps to frame your answer and demonstrate to the interviewer that you can approach handling situations in a **systematic way**. Interviewers will ask questions that draw on your experiences, such as “**Tell me about a time when you had to deal with a conflict. How did you resolve it?**” Before jumping into a specific answer, you might start with something like, “**Well, I believe that most conflict in the workplace can be avoided with good communication, and so it’s important that people spend time getting in sync often.**” Then go on to explain a situation where a conflict arose because there was a lack of communication between two people and how it was resolved. In this case, taking this approach shows that **you think systematically and can both perceive and handle problems like this in a repeatable manner**. When describing the situation, you might find the [STAR method](https://www.thebalancecareers.com/what-is-the-star-interview-response-technique-2061629) useful.

**Write the Script**

Think about the types of questions that you expect in the interview (and ask the recruiter). Then write them down and answer them, **ideally with anecdotal examples**. You should **never have to think hard** when an interviewer says something like, “Tell me about an interesting technical challenge you’ve solved and how you did it.” Here are some questions that I would make sure to pre-emptively answer:

* What are you looking for in your next role? Why this company?
* Tell me about a time when you had to work with a cross-functional team to solve a problem.
* Tell me about a time when you demonstrated a bias for action.
* Tell me about a time when you took ownership of a project or process and improved it.
* Tell me about a time when you had to deal with ambiguity.
* Tell me about a time when you disagreed with someone or had to resolve a disagreement.
* Tell me about a time when you led a project that failed. Why did it fail and what did you learn?
* What is your management style?
* How do you know that your team is working optimally?
* How do you handle low-performers?
* How do you approach solving complex problems?
* How do you evaluate buying vs. building technology?
* What is your approach to recruiting talent?

**Write down your principles.**

You should try to enumerate some of **your principles or virtues** for dealing with common workplace situations. It’s a good exercise to think about **how you’ve approached problems and reduce them down to first-order concepts**. You can then use these to frame your answers. Here are some examples of my principles:

* Perceive problems early and often, but don’t tolerate them. Don’t just call out problems without owning them.
* Be approachable and encourage others to probe by actively soliciting questions and challenges.
* Escalate when necessary to avoid getting stuck or to resolve a dispute or problem quickly.
* Act like an owner by putting the best interest of the organization and its people first.
* Be assertive and open-minded at the same time in controversial discussions or debates.
* Understand the disagreement in a situation before attempting to find agreement; this often reveals important information worth digging into.
* Get to the root cause to assess how a process is working, and/or to learn more about the people involved.

A good interview is a collaboration between the interviewer and the candidate. You should play an active role to help the interviewer get the information they are looking for.