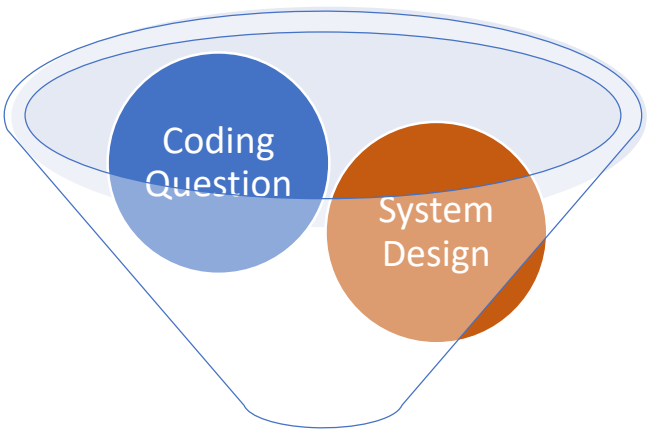
SDE Interview preparation

This document provides you a summary of the key information and tips shared during the preparation session. I includ links as part of the resources to help you to be prepared for this stage.

Preparation is absolutely essential!

SDE II Interview Preparation

At Amazon, our goal is to be the world's most customer-centric company by delivering innovative products, services, and ideas. The SDE interview is designed to identify candidates who have the technical proficiency, behavioral skills, and cultural fit.



Technical Phone Screen TPS

The Technical Phone Screen is a functional interview conducted by a developer from our team, with the aim of evaluating the candidate in the following aspects: Data structures and Algorithms, Logical and Maintainable, and Problem Solving. It is expected that the technical coding questions asked by the interviewer will be resolved with an optimal solution, creating a Production Level code.

Duration: 60 min
Introduction 5 min
System Design Question 15 min
Coding Question 35 min
Wrap up + Q&A: 5 min

Keep in mind always

- Listen carefully
- Clarify questions
- Think out loud
- Be honest and humble

Amazon Coding Sample

Data Structures and Algorithms Preparation :

You must be able to explain the inner workings of common data structures and be able to compare and contrast their usage in various applications.

You must be able to solve problems in Linear time.

You may be asked to solve problems with use combination of two or more data structures.

Absolute Must Haves :

1. Hash Tables – you must be able to explain how internals of hash tables work for example hashing.

2. Linked Lists

3. Big O Notation - you will be asked to determine time/space complexity in almost all your interview questions, and how to optimize your code for better time/space complexity. You are extremely likely to see a question where the solution will involve the use of hash tables.

4. Trees (especially Binary Search Trees) - be ready to explain how a binary search tree works a) if it's balanced and b) if it's not balanced.

5. Algorithms: Breadth First Search/Depth First Search, Binary Search, Merge Sort and Quick Sort, Sorting

6. Arrays, Recursion, Stacks/Queues, Bit Manipulation, Traversals, etc

What's the average and worst case time complexity, e.g. big O notation for

Data Structure	Operation	Average	Worst Case	Difficulty
linked list	insert	O(1) or constant time	O(1) or constant time	Easy
linked list	search	O(n) or linear time	O(n) or linear time	Easy
linked list	delete	O(n) or linear time	O(n) or linear time	Easy
linked list	update	O(n) or linear time	O(n) or linear time	Easy
hash table	search	O(1) or constant time	O(n) or linear time	Easy
hash table	insert	O(1) or constant time	O(n) or linear time	Easy
hash table	delete	O(1) or constant time	O(n) or linear time	Easy
binary search tree*	search	O(logn)	O(n)	Easy
binary search tree*	insert	O(logn)	O(n)	Easy
binary search tree*	delete	O(logn)	O(n)	Easy
binary search tree (balanced)	search	O(logn)	O(logn)	Easy
binary search tree (balanced)	insert	O(logn)	O(logn)	Easy
binary search tree (balanced)	delete	O(logn)	O(logn)	Easy
binary min-heap	insert	O(logn)	O(logn)	Medium
binary min-heap	delete-min	O(logn)	O(logn)	Medium
binary min-heap	find-min	O(1) or constant time	O(1) or constant time	Medium
array	sort - quicksort	O(n logn)	O(n^2)	Medium
array	sort - heapsort	O(n logn)	O(n logn)	Medium

What's the worst case space complexity?

Data Structure	Worst Case Size	Difficulty
linked list	O(n)	Easy
hash table	O(n)	Medium
binary search tree	O(n)	Easy
array quicksort	O(n) or O(logn)*	Medium
heapsort	O(1)	Medium

If candidate is fuzzy, no problem explain time complexity using the example of having to step through an array to find the largest element. Which is O(n) or linear time complexity

Coding

Tips

1. Ask clarifying questions to understand the requirements before you start to code.

2. **Familiarity with the syntax of a prominent language is generally recommended (such as C, C++, C++14, C#, Go, Java7, Java8, JavaScript, Kotlin, Objective-C, PyPy2, PyPy3, Python2, Python3, Ruby, Scala, and Swift).**

3. After you've gathered qualifying requirements, write **maintainable, readable, reusable, and understandable** code with clear functionality and descriptive names.

4. Translate qualifying requirements into **clean written code**, checking edge cases and providing sage examples.

5. Be ready to write **code in real-time** on an online editor.

6. Write syntactically correct code, no pseudo code. Start with a working solution and enhance as you go.

7. Make code **extendable** and avoid single functions that do everything.

8. Use clear and descriptive method, parameter, and variable names, and separate functionality into discrete methods/functions with clear responsibilities.

Common Mistakes in Coding Interviews

- Attempt to implement the perfect and complex solution with**out having a naïve working solution to the problem. Having a solution that is not optimal but solves the problem is better than no solution.**
- **Not mentioning or considering the edge cases. (It's okay to initially mention them and come back later to write the code that handles them).**
- **Not communicating your thought process with the interviewer.**
- **Not asking your interviewer for help if you get completely stuck.**

What if I make a mistake?

You might start writing the code and then realize your approach won't work or you might run out of time when your code is ¾ done. All is not lost!

If this happens, you still have a good opportunity to demonstrate honesty and understanding. Try to recover using these statements:

- ❑ "I see now this approach won't work" – **Be honest.**
- ❑ "The reason why is not working is because..."- **show you understand what is wrong.**
- ❑ "I don't have time to code it, but I could do _____" – **show that you think ahead of what you code.**
- ❑ "This works in most cases but not when _____" – **show that you found the edge cases.**

The important point is that you communicate clearly your work and your thought process.

Good engineers don't get the perfect solution right away, but they communicate clearly about the status and limitations of their work so that others can help them.

Isy Garcia
Technical Sourcer Recruiter II

System Design

Amazon System Design Preparation

- ❑ Ask clarifying questions (specially in the beginning)
- ❑ Describe the software components and how they interact, starting at a high level and proceeding to more detail
- ❑ State **trade-offs**
- ❑ **Scaling is a critical part of software design at Amazon.**

Common Mistakes in System Design

- Focus on the technology rather than focus on the design.
- Avoid rabbit holing.
- Do not forget to discuss every tradeoff decision.
- Do not forget to ask questions to solve the right problem.

Approaching System Design Interview

Discussion Order

- ❑ Interviewer describe the customer experience.
- ❑ You ask **clarifying questions** (especially at the beginning, but also throughout)
- ❑ You state **simplifying assumptions** you want to make (beginning, and throughout)
- ❑ You describe the **software components and how they interact, starting at a high level and proceeding to more detail** (writing code, if the interviewer asks) until we run out of time

** This is an intentionally **open-ended** question, so drive the conversation however you see fit. The interviewer will guide the conversation if they feel the need.*

> “You own a **large fleet** of servers that run a diverse set of workloads. Monitoring these servers is critical to ensuring these workloads complete successful. Design a system to **periodically** run a suite of **diagnostics** gather health data from these servers.

- ❑ What’s a “large fleet”? (1000s)
- ❑ What’s a “diagnostic”? (a small self-contained script)
- ❑ What is “periodic”? (hours)
- ❑ Who are our customers?
- ❑ What inputs/outputs do we want?

Important Areas

Load (a.k.a. scale)

- ❑ How is the system scaled?
- ❑ What breaks when volumes increase?
- ❑ How is the system measured and monitored?

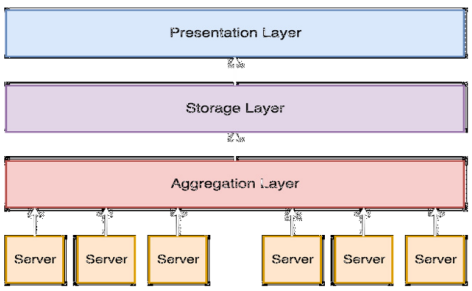
Correctness (and Availability)

- ❑ Does it satisfy the requirements?
- ❑ How can the system fail?
- ❑ Are there single points of failure?

Integration

- ❑ How are the components connected?
- ❑ What messages are interchanged?
- ❑ How are messages interchanged?
- ❑ What happens if a dependency breaks?

The simplest possible thing that might work is...



Prepare yourself and study	Practice your *english* skills	Be *specific*
Ask clarifying questions	Be honest and *vocal*	Take *ownership*

Isy Garcia
Technical Sourcer Recruiter II

Additional resources

Data Structure & Algorithms

HackerRank Interview Prep Kit

Coding Interview Questions

Clone a linked list with next and random pointer

Trees-and-graphs-interview-questions

OOD concepts

Object Oriented Design

Component Design interview

Programming_foundations

Binary Search Tree

Coding Technical Phone Interview
(start 13:43 to 43:53)

Coding Interview Prep Data Structures

Data Structures and Algorithms practice

Find the Running Median

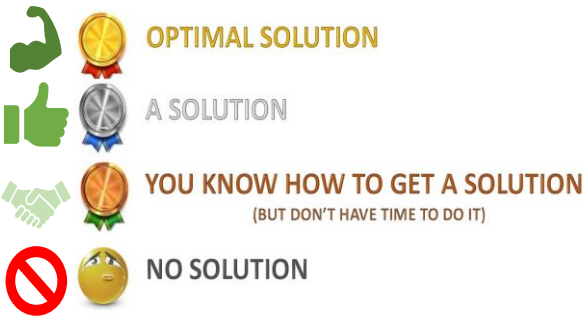
Data Structure Trees

Binary Search Tree Practice

Amazon Sample Test

DS_Trees Swap Nodes

Learn Code



System Design Primer Netflix System design

System Design Interview Questions

Amazon has a game platform for learning to use AWS solutions

System Design Introduction

System Design Interview Question

SD explanation and diagrams

System Design Mock Interview

Amazon System Design Preparation

System Design interview questions and Architectural concepts

System Design and Algorithms

System Design Interview videos

System Design in a hurry introduction

Mock Interviews

HIGHLY RECOMMEND!

Tech Talk Q&A session

Online Course: Interview Prep for Software Development Engineer