How to Ace Hopper's Engineering Interviews

Introduction

We know how stressful interviews can be so we want to see you at your best! This guide will give you some insight on what to expect during our interviews. We hope that knowing our process beforehand will make you more comfortable and simplify your preparation.

Our interview process has both technical challenges that you will need to solve, as well as behavioural questions tailored towards our leadership principles that help us learn more about your motivations, previous achievements, and values.

Technical Challenges

CODING

Both the screen and the onsite require you to write code without the help of an integrated development environment (IDE). It's not expected that you know a particular programming language prior to your interview, but you should be fluent in at least one. You are expected to be familiar with some of the nuances of your preferred language, such as how memory management works and commonly used libraries. Also, it is important to think of edge cases and remember that you are not only allowed but even encouraged to ask some clarifying questions instead of making assumptions. Don't stress about small syntactical errors or if your code compiles, but your code is expected to be correct (no pseudocode!).

DATA STRUCTURES

You must be capable of showing the different applications of common data structures by highlighting their fundamental principles, such as runtime and memory complexity for common operations. You should be familiar with most frequently used data structures, such as arrays, linked lists, stacks, queues, hash-sets, hash-maps, hash-tables, dictionaries, trees and binary trees, heaps and graphs.

EXTENSIBLE & MAINTAINABLE CODE

You should be able to explain how you would test your code. Ensure that your code is easily readable, verify any edge cases, and check for incorrect inputs. As you design your code, think about how you would modify it when new constraints or business requirements are added later. Many coding problems involve thinking recursively and potentially coding a recursive solution. Use recursion to find more elegant solutions to problems that can be solved iteratively.

DOMAIN KNOWLEDGE

During any round of the interview, we might poke around technologies/ tools/ languages/ frameworks you've included in your resume/ linkedIn profile so we expect you to be comfortable having an in depth discussion on any of these topics at any given time during the interview process.

ALGORITHMS

For this part of the interview, we want to see a solid understanding of the most typical algorithms which will help you solve our questions that much easier. While we don't expect you to memorize algorithms, you should be familiar with the most common types. You'll be expected to calculate the time and space complexity of your algorithm and know if/how you can improve it. We recommend working through requirements and discussing the algorithm you have in mind before writing code. You should also review the tradeoffs for common algorithms such as divide and conquer, dynamic programming/memoization, graph traversals, and breadth-first search vs. depth first search.

SYSTEM DESIGN

System design questions are used to assess your ability to combine knowledge, theory, experience and judgement towards solving a real-world engineering problem. The questions asked are deliberately underspecified because our interviewers want to see how you engage with the problem. For each question, talk through your thought process and ask clarifying questions before you dive into a solution. It's important to have an understanding of fundamental distributed computing concepts such as concurrency, locking, caching, load balancing, and service-oriented/microservices architecture.

Experience questions

In addition to technical challenges, we'll also ask you about past situations or challenges you've faced and how you handled them. Specifics are key: avoid generalizations. Give a detailed account of one situation for each question you answer and use data or metrics to support your example as you need to remember that we're all about data! Refer to recent situations whenever possible, and don't embellish or omit parts of the story. Ultimately, we're looking for examples that showcase your experience and demonstrate that you've taken risks, succeeded, failed, and grown over the course of your career.