Mock Interview Report

Coding & Algorithms | February 21, 2022

Topics Covered: Arrays, Strings, Sorting, Map

Interviewed by Austin Wentz

—

John Riselvato

Your Results

Needs Improvement
HIRE DECISION

2
TECHNICAL

2+
BEHAVIORAL

Interpretations

Hire Decision

Your interviewer believes one or more areas need improvement before a hire offer could be made in a real-world interview. This decision was based on a combination of your technical and behavioral performance. Check-out the score and overall interpretations for additional insight.

Technical Score

A score of 2 indicates struggle in some areas but not others. Some assistance was given. Solutions/code may have been tenable but suboptimal or incomplete. Interviewer lacks enough technical evidence to justify an offer.

Behavioral Score

A score of 2+ means you were nearly there behaviorally, but just fell short of inspiring an offer. A higher score is likely just a matter of tweaking your delivery on one or more criteria.

Everything Considered Together

Your behavioral performance was on-the-fence, and your technical performance needs some healthy improvement, so it's no surprise your interviewer would encourage you to improve before making you an offer. Your top priorities are getting both of these scores above the threshold of 3, which is needed to secure offers at most companies. Then you can focus on going from good to great.

Action Plan

High-Level Goals

Based on your technical and behavioral scores, your top three high-level goals for the near future are as follows:

1. ### [Technical] Increase score from 2 to 3

   You've demonstrated a mix of strengths and challenges on the topics tested in this mock. Now it's time to acknowledge those strengths and isolate your areas of improvement. Deepen your understanding, refine your delivery, and be more strategic with your time. Put your interviewer in a position to comfortably justify a hire.

2. ### [Behavioral] Increase score from 2+ to 3

   If you got a 2+, your interviewer likely noticed some concrete positives, but was on the fence nonetheless. Getting to a 3 from here is about tying up any remaining loose ends to get you unambiguously into the positive realm.

3. ### [Overall] Validate performance in additional areas

   Since you were solid on topics tested in this mock, you should confirm whether the same performance is possible when tested on other coding topics, system design, or behavioral questions. When all bases are covered, you can graduate to even more sophisticated levels of refinement.

Next Steps to Improve

Based on topic-specific scores, these are the most immediate concrete next steps you can take to improve your performance on future interviews:

1. ### [Behavioral] Tackle interview anxiety

   To quote the sci-fi classic Dune, 'fear is the mind killer.' Interview anxiety impacts every aspect of your performance, behaviorally and technically. But it's also way more changeable than you might think.

2. ### [Technical] Brush-up on specific technical topics

   Sometimes performance is simply a matter of better understanding specific technical topics. Your interviewer mentioned one or more topics they think you'd benefit from exploring in the notes at the end of this guide.

3. ### [Technical] Get to a working solution

   It's easy to miss problem requirements or go down a rabbit hole that ultimately doesn't work for a problem given. Here's how to get better at quickly arriving at a working solution, and what to do if you get stuck.

Detailed Scores

The following is a complete listing of scores for all criteria measured. All scores are based on a 1 through 4 scale. For simplicity, it is not possible to receive a score of 2+ for the criteria below. For further information about scoring, check out our scoring guide.

Problems Completed

Problem Text	Problem Difficulty
[P1] Given an array nums with n objects colored red, white, or blue, sort them in-place so that objects of the same color are adjacent, with the colors in the order red, white, and blue. We will use the integers 0, 1, and 2 to represent the color red, white, and blue, respectively. You must solve this problem without using the library's sort function. Example 1: Input: nums = [2,0,2,1,1,0] Output: [0,0,1,1,2,2] Example 2: Input: nums = [2,0,1] Output: [0,1,2]	[P1] Easy
[P2] Given an array of strings strs, group the anagrams together. You can return the answer in any order.	[P2] Medium

An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

Example 1:

Input: strs = ["eat","tea","tan","ate","nat","bat"]

Output: [["bat"],["nat","tan"],["ate","eat","tea"]]

Example 2:

Input: strs = [""]

Output: [[""]]

Example 3:

Input: strs = ["a"]

Output: [["a"]]

Understanding

[P1] 4 — The candidate quickly and clearly understood the problem, with little to no resistance in processing what was being asked.

[P2] 4 — The candidate quickly and clearly understood the problem, with little to no resistance in processing what was being asked.

Working Solution Proposed

[P1] 1 — The candidate did not arrive at a working solution or the solution worked to a very limited degree.

[P2] 4 — The candidate arrived at a working solution that addressed all problem requirements and covered all possible edge cases. Well done.

Optimized Solution Proposed

3 — The solution was solidly interview-appropriate. It was both sufficiently performant and algorithmically/logically optimized.

Solution Implemented

3 — The solution was implemented in code to a degree acceptable for a real-world interview.

Problem-Solving Details

Hints Needed

2 — The candidate relied on a fair amount of hinting, but also made not-insignificant progress on their own.

Hints Effectively Utilized

2 — The candidate listened to some hints, but not others. The degree to which they helped was limited.

Verbal Presentation

1 — The candidate either explained very little or was unclear in communicating their thinking around their solutions.

Bugs

Amount & Severity

3 — A small number of bugs were detected and none were substantial enough to raise a red flag. Well done.

Discovered & Fixed

2 — The candidate could identify and fix some bugs on their own, but needed significant help from the interviewer. On the other hand, they struggled to fix important bugs or a sufficient amount of little bugs regardless of help.

Time & Space Complexity

2 — The candidate struggled to effectively communicate time/space complexity, but was able to demonstrate some understanding.

Code

Readability

3 — The code was sufficiently readable.

Programming Language Comfort

3 — The candidate seemed sufficiently comfortable with their programming language. Any quirks noticed are within the realm of acceptable. They are fluent enough to use this language in a real interview.

Behavioral

Perceived Humility

4 — The candidate demonstrated exceptional humility. At no point could it even be possible to mistake this person as arrogant. It was a pleasure to engage with them.

Perceived Energy

3 — The candidate's energy was sufficient to high in one or more areas. Enthusiasm was expressed on a handful of topics and they give enough of an impression of embracing forward momentum.

Communication

2 — Communication was mixed. Sometimes they were clear enough, other times some significant probing needed to be employed.

Perceived Interview Anxiety

1 — The candidate appeared to be very nervous to such a degree that it likely impacted their performance substantially.

Interviewer Notes

This section covers nuances your interviewer would like you to know.

Topics to Brush-Up On

Candidate could brush up on:

1. Pointers--when to use them vs when not to.

2. How to implement in-place swapping algorithms

3. Iterating through a map structure.

4. Space complexity notation

Strengths Noticed

Once the candidate was given a structure for interviewing, the candidate did very well.

1. The candidate was able to understand the questions and come up with edge cases effectively.

2. The candidate proposed an optimal solution in the second problem.

Additional Feedback

Good structure for interviewing:

1. Understand the problem completely and convey this to the interviewer by walking through example inputs and outputs.

2. Look for edge cases and gotchas

3. Come up with brute force and optimal solutions and discuss the trade offs between time and space complexity between them (if you cant come up with the optimal approach, mention this to the interviewer)

4. Write some really high level sudo code

5. Code

6. Walk through test cases, be sure to include edge cases from step 1!

What I am looking for as an interviewer:

-Capturing edge cases, asking questions

-Understanding solution before coding

-Time complexity, space complexity

-Communication skills

-Debugging skills

-Code organization

-Variable naming, code structure

-Pacing, speed