Selected Soft Skills Advice

1. Proactive Communication

- Don’t assume the question is easy
    - Repeat the question and rephrase it in your own words; the interviewer will clarify if necessary
    - Assume all information that is given is necessary to solve the problem

- Ask A LOT of questions
    - Make sure you fully understand what you’re being asked to return
    - Resolve any areas of ambiguity; clarify the scope and intention of the problem
    - Validate or state assumptions
    - If there are edge cases (inputs that could break your solution), how should they be handled?

2. Design your Algorithm

- Approach 1 - Pattern Matching

Consider what problems the algorithm is similar to, and figure out if you can modify the solution to develop an algorithm for this problem.

- Approach 2 - Simplify & Generalize

Change a constraint (data type, size, etc.) to simplify the problem. Solve it. Once you have an algorithm for the simplified problem, generalize again.

- Approach 3 - Base Case & Build

Solve the algorithm first for a base case (e.g. just one element). Then, try to solve it for elements one and two. Then, try to solve it for elements one, two, and three.

- Approach 4 - Data Structure Brainstorm

This is a bit hacky, but it often works. Simply run through a list of data structures and try to apply each one.

3. Discussing your Solution

- Think out loud

Communicate verbally to demonstrate how you tackle hard problems, consider and communicate engineering tradeoffs

- See interviewers as collaborators

Look up from the whiteboard and talk to the interviewer
Make eye contact
You might even catch some non-verbal cues about whether or not you’re on the right track

- Ask questions,don’t ask for hints

Questions aren’t just for the first 5 minutes of the interview
It’s okay to admit you’re stumped; ask a question that helps identify the path

forward

- Evaluate the time and space complexity

Summarize the big-o of your solution and why it’s less than ideal (if necessary)

4. Handling Mistakes

- Trace the code you’ve written with an example. Pretend it’s someone else’s
  - By tracing the code you wrote, not what you intended to write, you’ll catch mistakes
- Acknowledge the mistake, explain how to fix it, and implement the correction

- If your interviewer spots an error and you can’t find it, it’s okay to talk through assumptions or ask for a hint. Listen carefully!

5. Testing your Code

- Failing to test your code is quick way to undermine yourself in your technical interview.
  - It’s very common for code to fail in some specific test cases
  - After all, would you ever write code without running or testing it?
- It’s your job to read your interviewer’s mind.
  - Don’t make them do the intellectual heavy lifting
  - It’s a huge plus if you write tests for your code without being prompted