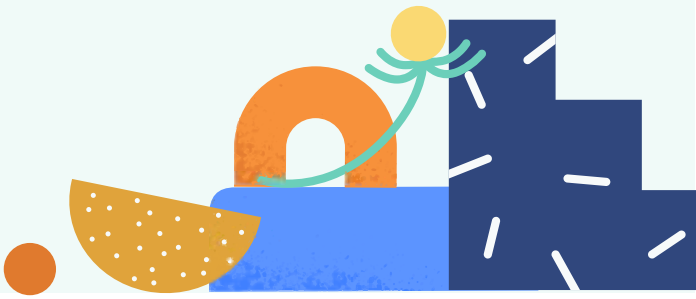
facebook

Technical Screen Interview Guide



What You'll Find in This Guide

Technical skills aren't the same as interview skills, so even the most experienced engineers need to prepare and practice to do well in an interview. For example, it's difficult for interviewers to get a clear signal on coding ability from someone who hasn't practiced solving new problems under time constraints. This can make someone who's simply under-prepared look under-qualified.

This guide can help you plan, practice, and prepare for your initial technical screen at Facebook.

What We Look For

Your interviewer will be thinking about how your skills and experience might help their teams. Help them understand the value you could bring by focusing on these traits and abilities.

- **Communication.** Are you asking for requirements and clarity when necessary, or are you just diving into the code? Your initial tech screen should be a conversation, so don't forget to ask questions.
- **Problem solving.** We're evaluating how you comprehend and explain complex ideas. Are you providing the reasoning behind a particular solution? Developing and comparing multiple solutions? Using appropriate data structures? Speaking about space and time complexity? Optimizing your solution?
- **Coding.** Can you convert solutions to executable code? Is the code organized and does it capture the right logical structure?
- **Verification.** Are you considering a reasonable number of test cases or coming up with a good argument for why your code is correct? If your solution has bugs, are you able to walk through your own logic to find them and explain what the code is doing?

How to Prepare

Interviewers can only assess your skills and abilities based on what you show them during your interview, so it's important to plan and prepare to best showcase your strengths.

1. Before you practice, plan!

Be honest with yourself—only you know how much prep time you'll need. Make the most of your prep time by following these steps to plan your approach before you start practicing.

Schedule time to study and practice. Block out time every day to write code. Target medium and hard problems.

Prioritize breadth over depth. It's much better to practice solving fewer example problems of many problem types than to become very familiar with one type at the expense of the others.

Set aside time to review what you've practiced. As you solve problems, make cheat sheets or flash cards to review later. Revision and repetition will strengthen your understanding of core concepts.

Remember your goal. Aim for confidently solving two questions—while thinking aloud—in about 35 minutes.

2. Use key practice strategies to practice effectively

Reading through sample questions, recognizing concepts, and having a vague understanding of these concepts won't be enough to help you shine. You need to practice! Make sure you're setting your practice sessions up for success by following these tips from engineers who've been through the process.

Practice coding the way you'll code during your tech screen. Use CoderPad.io if your interview is via phone or video call, or use a whiteboard or pen and paper if your interview will be in person. Check with your recruiter if you're not sure which format you'll use.

Set a time constraint when you practice problems. In your tech screen, you'll be asked to solve one or two problems in under 35 minutes. Practice coding solutions to medium and hard problems in less than 15 minutes each to help you be ready for the constraints during the interview.

Code in your strongest language. Provide the most efficient solution, and find and fix the bugs yourself.

Practice talking through the problem space and possible solutions before you dive in, and talk through your decisions out loud as you code. Interviewers will be evaluating your thought process as well as your coding abilities. Explaining your decisions as you code is crucial to helping them understand your choices. The more you practice this, the more natural it will feel during the interview.

3. Understand the types of problems you may encounter

Practice a variety of different problems—and understand why we ask them—so you're prepared to solve them during your interview.

Don't be surprised if the questions sound contrived. Problems may be different than what you're probably tackling in a day-to-day job. We won't ask a "puzzle" question, but questions may be different than real-world questions because they need to be described and solved in 10-20 minutes.

Problems may assess the depth of your knowledge and your versatility. For example, your interviewer might ask you to solve a problem any way you want. Then, they could add constraints on the running or space characteristics and ask you to solve it again.

Problems may focus on edge cases. You might be asked to parse some data format or mini language. Your answers demonstrate your ability to handle multiple states in your head.

Problems may test how well you know how things work under the hood. For example, you might be asked to implement well-known library functions.

4. Decide what resources you'll use to prepare

It's easy to be overwhelmed by the number of online resources or the detail in an entire theoretical algorithms book. Here are some sites that our engineers found helpful when preparing for their coding interviews.

Top sites for practice problems from Facebook:

- Facebook Sample Interview Problems and Solutions
- InterviewBit

Other websites:

- LeetCode
- HackerRank

Video prep guides for tech interviews:

- Cracking the Facebook Coding Interview: The Approach, password: FB_IPS
- Cracking the Facebook Coding Interview: Problem Walk-through, password: FB_IPS
- Example Interview, password: fbprep

Example tech screen study list:

- See page 5 for an example list of exercises from Facebook's engineering team you can use as a starting point to help you prepare. Feel free to tailor it to your specific practice needs.

Additional resources*:

- I Interviewed at Five Top Companies in Silicon Valley in Five Days, and Luckily Got Five Job Offers

**It's not necessary to review these resources when preparing for your initial tech screen, but engineers recommend them to understand the entire technical interview process.*

How to Approach Problems During Your Interview

1. Before you code

- **Ask clarifying questions.** Talk through the problem and ask follow-up questions to make sure you understand the exact problem you're trying to solve before you jump into building the solution.
- **Let us know if you've seen the problem previously.** That will help us understand your context.
- **Present multiple potential solutions, if possible.** Talk through which solution you're choosing and why.

2. While you code

- **Don't forget to talk!** While your tech screen will focus heavily on coding, the engineer you're interviewing with will also be evaluating your thought process. Explaining your decisions and actions as you go will help the interviewer understand your choices.
- **Be flexible.** Some problems have elegant solutions, and some must be brute forced. If you get stuck, just describe your best approach and ask the interviewer if you should go that route. It's much better to have non-optimal but working code than just an idea with nothing written down.
- **Iterate rather than immediately trying to jump to the clever solution.** If you can't explain your concept clearly in five minutes, it's probably too complex.
- **Consider (and be prepared to talk about):**
  - Different algorithms and algorithmic techniques, such as sorting, divide-and-conquer, recursion, etc.
  - Data structures, particularly those used most often (array, stack/queue, hashset/hashmap/hashtable/dictionary, tree/binary tree, heap, graph, etc.)
  - O memory constraints on the complexity of the algorithm you're writing and its running time as expressed by big-O notation.
- **Generally, avoid solutions with lots of edge cases or huge if/else if/else blocks**, in most cases. Deciding between iteration and recursion can be an important step.

3. After you code

- **Expect questions.** The interviewer may tweak the problem a bit to test your knowledge and see if you can come up with another answer and/or further optimize your solution.
- **Take the interviewer's hints to improve your code.** If the interviewer makes a suggestion or asks a question, listen fully so you can incorporate any hints they may provide.
- **Ask yourself if you would approve your solution as part of your codebase.** Explain your answer to your interviewer. Make sure your solution is correct and efficient, that you've taken into account edge cases, and that it clearly reflects the ideas you're trying to express in your code.

What to Practice: An Example Tech Screen Study List

Everyone could use a refresher in at least one core area! Before your initial tech screen, brush up on CS fundamentals— especially algorithms, data structures, object-oriented design, and design patterns in general. Review foundational techniques—recursion, graph theory, tree traversal, combinatorial problems, and so on.

Looking for more detailed guidance on what to review for your tech screen? The exercises below have been helpful for many engineers preparing for a Facebook tech screen and can assist you in solidifying your understanding of data structures and algorithms. Feel free to use this list as a starting point and tailor it to suit your areas of need.

1. Overview

- Each exercise could take you up to one hour.
- These solutions are written in Java, but you will be able to use your language of preference in an interview.
- Remember how to analyze how "good" your solution is: how long does it take for your solution to complete? Watch this video to get familiar with Big O Notation.

2. Exercises

Note: These exercises assume you have knowledge in coding but not necessarily knowledge of binary trees, sorting algorithms, or related concepts.

- **Topic 1 | Arrays & Strings**
  - A Very Big Sum (Warm-up, learning how to use HackerRank)
  - Designer PDF Viewer
  - Left Rotation
- **Topic 2 | Lists**
  - Pre-work: If you need to familiarize yourself with how lists work, watch this video.
  - *Exercises*
    - » Insert a Node at a Position Given in a List
    - » Cycle Detection
- **Topic 3 | Stacks & Queues**
  - Pre-work: If you need a refresher, take a look at this video
  - Exercises
    - » Balanced Brackets
    - » Queue Using Two Stacks
- **Topic 4 | Hash & Maps**
  - Pre-work: If you need a refresher, take a look at this video
  - Exercises
    - » Ice Cream Parlor
    - » Colorful Number (This one might be challenging. Remember, if you get stuck, refer to our proposed solution.)

- **Topic 5 | Sorting Algorithms**
  - Pre-work: If you need a refresher take a look at these videos: Merge Sort
  - Exercises
    - » Insertion Sort part 2
    - » Quicksort part 2

- **Topic 6 | Trees**
  - Theory: If you need a refresher, take a look at this video
  - Exercises
    - » Binary Tree Insertion
    - » Height of a Binary Tree
    - » QHeap1

- **Topic 7 | Graphs (BFS & DFS)**
  - Theory: Watch this video to understand what a graph is and how to traverse it
  - Exercises
    - » Breath First Search
    - » Snakes and Ladders

- **Topic 8 | Recursion**
  - Theory: Watch this video to review concepts on recursion
  - Exercises
  - Fibonacci Numbers

Solutions: All solutions are available in this public repository: https://github.com/lolapriego/coursework