Machine Learning

Initial Interview Guide

Meta

What You'll Find in This Guide

Welcome to your prep guide for your machine learning (ML) initial interview at Meta. Our ML engineers and recruiters put together this guide so you know what to expect and how to prepare.

Time

How much time can you expect to prep for and take your initial tech screen?

Prep Time

Start your prep early. Before your interview date, be sure to spend time practicing your coding, algorithmic, and problem-solving skills.

45 Minutes to Interview

Your conversation with an engineer will be divided into the following time blocks:

- **Introductions:** 5 minutes.
  Let's get to know each other. We'll talk about us and you'll talk about who you are, where you've trained and worked, and what your areas of expertise are.
- **Coding:** 35 minutes.
  Solve one or more coding problems focused on computer science fundamentals like algorithms, data structures, recursions, and binary trees. If your tech screen is by phone, the engineer will send you a collaborative editor (such as CoderPad).
- **Your Questions:** 5 minutes.
  Take this brief opportunity to learn more about working at Meta from an engineer's point of view. Think about what you find interesting and challenging about the work you'd be doing.

Interview Overview

How is your initial tech screen structured?

Your interview will be by phone with an engineer, and you'll be coding remotely on CoderPad, so make sure you've practiced using it to code. Also, check your tech so it's solidly operable—fully charged, good internet connection, and hands-free.

The interview is formatted to see how candidates can quickly come up with the most efficient solution to a problem in their head first, and then quickly code it without logic flaws. You'll need to think of corner / edge cases before and while you code, then check your code at the end. The coding problem is worked out by hand on the digital whiteboard so the interviewer is able to see your thought process and collaborate with you.

Content

The content of the screen could cover a variety of topics, but generally interviewers want to assess your ability to develop original software in a short period of time. Interview topics may cover anything on your resume, including:

Critical Topics:

Data Structures:

- Arrays and lists
- Binary trees
- Hash tables
- Stacks and queues
- Graphs

Algorithms:

- Search: iterator, binary, hash
- Sort: merge, quick, bucket
- Graph traversals: BFS, DFS
- Complexity, Big O notation
- Recursion

Nice to Have:

Data Structures:

- Trie
- Heap
- Set
- Red-black trees

Algorithms:

- Randomized quicksort
- Dynamic programming
- Heap sort, radix sort
- Spanning tree, minimum cut

We may pose some of the coding problems as an ambiguous, real-world problem, and we'll ask you to interpret the problem and apply your knowledge to find a solution. You'll need to interpret the coding knowledge that you have for that particular situation. Interviewers are looking for insight into your thought process, creative solutions, ability to work out more than one way to solve a problem, and ability to talk through your rationale for choosing a certain way to approach solving the problem. So, you could perhaps recommend an algorithm, code up a solution using that algorithm, analyze the runtime of your code and then optimize your solution.

Prep Guide for Your Initial Screen

From big picture to the specifics, this is how our engineers suggest you prepare

In general

- **Do as many coding questions as you can.** Go to Program Creek and review the Top 10 algorithms for a coding interview. Try to pick a few of the classic examples to solve by hand and from scratch on a blank sheet of paper. Time yourself so you're completing each question within 15 – 20 minutes. This exercise should best prepare you with what to expect during the interview.
- **Familiarize yourself with key data structures and algorithms.** Really important structures include (but are not limited to) lists, arrays, hash tables, stacks, queues, graphs, various flavors of trees and tries, and heaps. Knowing about balanced trees couldn't hurt. It's really important to know about major types of sorts (mergesort, quicksort, radix sort, etc.), searches, and traversals (BFS, DFS). Knowing how to find a spanning tree or a minimum cut, or knowing about dynamic programming could be nice to have. Be able to discuss the Big O complexity of your approaches. Definitely know when and when not to use recursion.
- **Focus on talking through your logic.** Your reasoning is important. Engineering is all tradeoffs. Be able to discuss those.

More specifically...

To get a deeper sense of what we mean, visit our Meta Engineering Interview Prep pages to see what tips engineers have for initial and onsite interviews.

Prep exercises

Take these actions to test your knowledge:

- **Review** this Diagram to break down interview questions.
- **Study** the Data Structures. This guide will give you an overview of how to implement each one. Your goal should be to know how to code these out.
- **Read** the book Cracking the Coding Interview. You'll start to notice the solution rotates around recursion and a data structure.
- **Practice** real questions online like LeetCode Regular Problem Sets. Attempt to solve medium and hard problems. You should be able to consistently get through medium
  to hard problems within 40 minutes and at an optimal complexity.
- **There is no need to learn or know how to do dynamic programming or memorization.**
- Get in the habit of communicating your thought process out loud.

What Success Looks Like

Considerations that lead to a positive outcome

Interviewers will weigh the success of an interview based on the approach as much as the answer. They'll funnel your performance based on the following considerations:

- Do you listen carefully and comprehend the question?
- Are you asking the correct questions before proceeding?
- Do you hear and heed hints?
- Are you quick to comprehend / solve problems?
- Do you enjoy finding multiple solutions before selecting the best one?
- Are you looking for new ideas and methods of tackling a problem?
- Are you inventive and flexible in your solutions and open to new ideas?
- Can questioning move up to more complex problem solving?

11 Tips for Your Initial Screen

What to keep in mind while you're interviewing

1. Avoid misunderstanding.

When we ask you to provide a solution, first define and develop a framework of the problem as you see it. Ask for help or clarification and spend two to five minutes asking the interviewer about corner cases on the problem. This will ensure that you've understood the problem correctly.

2. Think out loud.

It helps your interviewer follow along, learn about your problem-solving skills, and provide hints if needed.

3. Write a working solution and iterate.

It's better to have a non-optimal but working solution than random fragments of an optimal but unfinished solution.

4. Hints.

If your interviewer gives you hints to improve your code, please run with them.

5. Don't worry about memorizing tables of runtimes or API calls.

It's always good to know how to figure out approximate runtimes on the fly but the code you write is more important.

6. If your solution is getting messy, step back.

Most coding interview questions are designed to have reasonably elegant solutions. If you have if-else blocks and special cases everywhere, you might be taking the wrong approach. Look for patterns and try to generalize.

7. Plan your approach.

Ensure that you spend time planning your approach but remember you can always go brute force and then optimize from there.

8. Clarifying questions.

Make sure you're asking clarifying questions as you go along (there won't be tricks but ensure you have all the info you need).

9. Think of cool things that you've done in engineering.

You'll likely talk about things you've made.

10. Questions.

You'll most likely have some time at the end for questions for your interviewer. Some people find it easier to come up with a few questions in advance rather than think of them on the spot.

11. Cancelling.

Don't hesitate to cancel if something comes up. If you won't have a quiet space, a good internet connection, and a good phone connection, or you had to stay up all night with a sick child or broken system, please reschedule. We want you at your best and will be happy to move your interview to a better time.

Appendix / Resources

Links to exercises, information, and guides to help you prepare

For your initial tech screen, our engineers collected some helpful resources with content and activities. Take a look through the list as you prepare.

Coding questions

- CareerCup: Archive of interview questions by the author of Cracking the Coding Interview.
- Glassdoor: Archive of interview questions and reviews of interviews.
- Project Euler: Coding problems, but not interview questions.
- GeeksforGeeks Interview Questions
- Cracking the Coding Interview
- CodeChef
- Use CodeRunner App, an editor to practice your algorithms.
- To study runtime complexity, use Big-O Cheat Sheet.

Resources from Meta

- How to Crush Your Coding Interview: A video about how to prepare for programming interviews at Meta.
- Open Source at Meta

Background Information about Meta

- Meta Quarterly Earnings
- Engineering Blog: The Meta Engineering blog. Certainly not required reading, but highlights neat stuff we've built or open-sourced.
- Meta News Room: Meta's global newsroom, highlighting recent product launches.

Helpful books

- Cracking the Coding Interview by Gayle McDowell. Best-known and best-selling book about software interviews.
- Introduction to Algorithms by Cormen, et al. The classic textbook on algorithms and data structures.
- Elements of Programming Interviews: The Insiders' Guide by Adnan Aziz.
- Programming Pearls by Jon Bentley. Not about interviews, but a fun book about programming tricks.

Thank you for taking the time to review this guide!