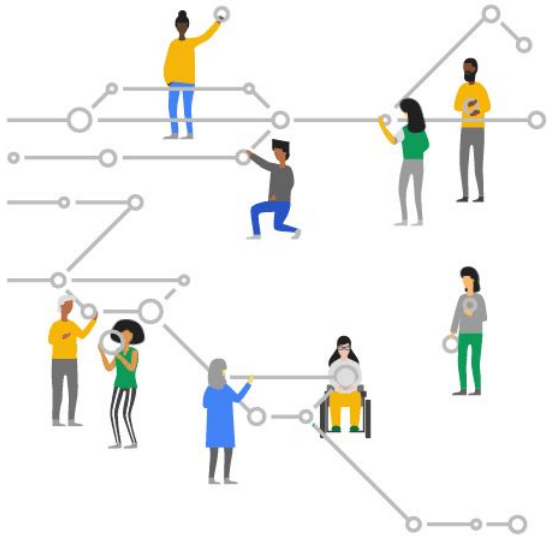
Google Technical Internships

Interview preparation

EMEA

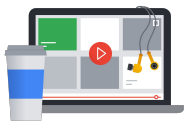


What you'll find in this deck...

This guide is intended to help you prepare for interviews as part of the application process for **Software Engineering and Site Reliability Engineering** and **STEP** internships at Google. If you think you have been sent the wrong preparation materials or have any additional questions, please don't hesitate to get in touch with your recruiter.

1. The recruitment process
2. Technical interview best practices
3. Areas of assessment and tips
4. Resources

Click the dots to skip ahead



Click to watch a video of a Google Engineer and Intern Team Recruiter talking through this slide deck and extra tips.

The recruitment process

The recruitment process

Technical interview

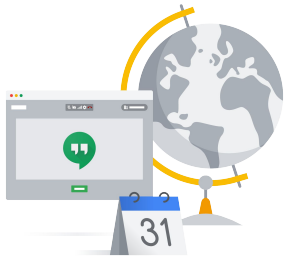
Feedback review

Intern Placement Interview

Offer

You're at this stage!

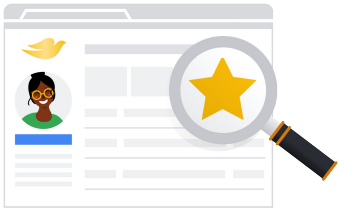
The recruitment process



1. Technical Interviews

You will have two technical interviews with a Google Engineer via Google Meet.

A shared Google Doc will be used as a virtual whiteboard for coding and algorithmic problems.



2. Feedback Review

Your interview feedback will be reviewed. We ensure our hiring process is fair and that we're holding true to our "good for Google" standards.

Your recruiter will let you know as soon as there's an update.

The recruitment process

3. Intern Placement Interview (IPI)

Your candidate questionnaire and application will be shared with Intern Hosts. You will meet with them via Google Meet to discuss if your skills and interests are in line with those of the team.

Your recruiter will share more information about IPIs at this stage.

4. Offer Stage(!)

Your application will be reviewed for a final time. If approved, you will receive a written offer.

Your recruiter will talk you through the offer details, including immigration, relocation requirements, and starting your Google internship!

Things to do before your interview

Check your internet connection and access to the shared Google Doc

You will need a computer with internet access to use the shared interview coding Google Doc.

Click on the link provided in your interview confirmation, and you'll be able to edit the document. Please check that you are able to access this document **at least 1 day** before your interview.

Use a headset

Additionally, we recommend that you have a headset available for the interview. Be sure to test the quality of your device ahead of time to avoid any disruptions during the call.

Accommodations

If you require any special accommodations for your interviews, please let your recruiter know as soon as possible.

Alternatively, you may contact our accommodations partner **EmployAbilityUK**, if you require any disability accommodations to be made. You can contact them confidentially at +44 7852 764 684 or email them at info@employ-ability.org.uk.

On the day of your interview

Join ahead of time

On the day of your interviews, we recommend joining the Google Meet link before the start time. It's a good idea to be ready at least 5 minutes before your calls.

- Test your camera and audio
- Check the shared Google Doc

Technical interview best practices

Technical interview tips

Check your CV and prepare examples

Familiarise yourself with your CV, and make sure that you can back up what your CV/ resume says with examples. If your resume says that Python is your key programming language, you can assume that the interviewers will ask you questions about it.

Practice

Make sure you practice writing code in a Google Doc to prepare for your technical interviews. Be sure to test your own code and ensure it's easily readable without bugs. Keep in mind, you can choose one of the following programming languages you're most comfortable coding in: C++, Java, Python and Go.

Technical interview tips

Clarify

Clarifying questions are encouraged! Ask clarifying questions if you do not understand the problem or need more information. Many of the questions asked in Google interviews are deliberately underspecified because our engineers are looking to see how you engage the problem. In particular, they are looking to see which areas you consider to be the most important piece of the technological puzzle you've been presented.

If your interviewer shares tips with you, you should consider them seriously because they are trying to help you. They may ask some additional questions or share comments when you're solving a problem. Take a minute to think through these prompts, they might help you land on a better solution!

Technical interview tips

Explain and think out loud

Explain your thought process and decision making throughout the interview. The interviewers are evaluating not just your technical abilities, but also how you approach problems and how you try to solve them. Many of the questions asked in Google interviews are open-ended because our engineers want to see how you break down and approach the problem.

There isn't always a "one true answer" to the questions. We recommend starting with a solution (even if you know it is partial and sub-optimal), and working from there as it is always important to show good problem solving capabilities. We want to understand how you think. This includes explicitly stating and checking any assumptions you make in your problem solving to ensure they are reasonable.

Technical interview tips

Think about ways to improve your solution

Share the different options or trade-offs you are considering. **Be flexible**. In many cases, the first answer you think of may need refining. It is worthwhile to talk about your initial thoughts to a question.

Here are some guidelines:

- You want to get to the simplest solution as quickly as possible. You can start with a brute force solution, but then...
- **Discuss trade-offs**. How would you improve your solution? How do you make it faster? Use less memory? Know the time and space trade-offs of the solution/data structures you pick.
- Code. We generally don't want pseudocode. It might be acceptable in some limited cases. If you are not sure, ask the interviewer who will be able to guide you through.
- If you don't remember the exact interface to a library class or method, that's okay. Let the interviewer know, and write code assuming some similar interface.

Technical interview tips

Ask questions

At the end of the interview, most interviewers will ask you if you have any questions about the company, work environment, their experience etc. It’s always good to have some questions prepared for your interviewers.

Areas of assessment and tips

Technical preparation

Coding

Google engineers primarily code in C++, Java, Python and Go. We ask that you choose one of these languages during your interview. If you would like to change your interview language, please let your recruiter know.

In your interview, you will be asked to write your code in real time in Google Docs.

You may be asked to:

- Construct/traverse data structures
- Implement system routines
- Distill large data sets to single values
- Transform one data set to another

Technical preparation

Algorithms

You will be expected to know the complexity of an algorithm and how you can improve and change it. Some examples of algorithmic challenges you may be asked about include:

- Big-O analysis: understanding this is particularly important
- Sorting and hashing
- Handling very large amounts of data

Sorting

We recommend that you know the details of at least one n*log(n) sorting algorithm, preferably two (e.g. quicksort and merge sort). Merge sort can be highly useful in situations where quicksort is impractical, so take a look at it.

What common sorting functions are there? On what kind of input data are they efficient, when are they not? What does efficiency mean in these cases in terms of runtime and space used? E.G. in exceptional cases, insertion-sort or radix-sort are much better than the generic QuickSort/MergeSort/HeapSort answers.

Technical preparation

Data structures

Refresh your knowledge on as many other structures and algorithms as possible. We suggest that you know about the most famous classes of NP-complete problems, such as travelling salesman and the knapsack problem. Be ready to recognise them if they come up.

You will also need to know about trees, basic tree construction, traversal and manipulation algorithms, hash tables, stacks, arrays, linked lists and priority queues.

Hash tables and maps

Hash tables are arguably the single most important data structure known to mankind. You should be able to implement one using only arrays in your favorite language, in the space of one interview. You'll need to know the O() characteristics of the standard library implementation for Hash tables and Maps in the language you choose to write in.

Technical preparation

Trees

We recommend you know about basic tree construction, traversal and manipulation algorithms. It could be useful to have some familiarity with binary trees, n-ary trees and trie-trees. You should be familiar with at least one flavour of balanced binary tree, whether it's a red/black tree, a splay tree or an AVL tree. You should know how it's implemented. You should know about tree traversal algorithms: BFS and DFS, and know the difference between inorder, postorder and preorder.

Graphs

To consider a problem as a graph is often a very good abstraction to apply, since well-known graph algorithms for distance, search, connectivity, cycle-detection etc. will then yield a solution to the original problem. There are 3 basic ways to represent a graph in memory (objects and pointers, matrix, and adjacency list); familiarise yourself with each representation and its pros/cons. You should know the basic graph traversal algorithms, breadth-first search and depth-first search. Know their computational complexity, their tradeoffs and how to implement them in real code.

Technical preparation

Min/max heaps

Heaps are useful. Understand their application and O() characteristics. We probably won't ask you to implement one during an interview, but you should know when it makes sense to use one.

Recursion

Many coding problems involve thinking recursively and potentially coding a recursive solution. Prepare for recursion, which can sometimes be tricky if not approached properly. Practice some problems that can be solved iteratively, but a more elegant solution is recursion.

Technical preparation

Operating systems

You should understand processes, threads, concurrency issues, locks, mutexes, semaphores, monitors and how they all work. Understand deadlock, livelock and how to avoid them. Know what resources a process needs and a thread needs. Understand how context switching works, how it's initiated by the operating system and underlying hardware. Know a little about scheduling. The world is rapidly moving towards multi-core, so know the fundamentals of "modern" concurrency constructs.

Mathematics

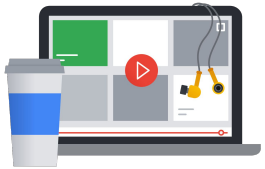
Some interviewers ask basic discrete math questions. This is more prevalent at Google than at other companies because counting problems, probability problems and other Discrete Math 101 situations surrounds us. Spend some time before the interview refreshing your memory on (or teaching yourself) the essentials of elementary probability theory and combinatorics. You should be familiar with n-choose-k problems and their ilk – the more the better.

Resources

Engineering interviews at Google

[Example coding/ engineering Interview]

(video)



[How to prepare for a Google Engineering interview]

(video)

[Interview tips from Google Software Engineers]

(video)

Click for the links

Extra reading

[The Google Technical Interview How to Get Your Dream Job]

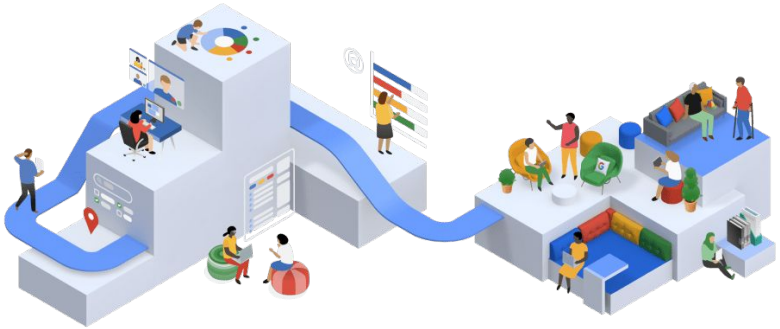
(article)

[Distributed systems and parallel programming]

(publications)

Click for the links

Overview of interviewing at Google



[The hiring process for students]

[Overview of how we hire at Google]

Click for the links

Get to know Google better

Learn about Google's products and technologies

- About Google
- How Google Search works
- Google Spanner: Google's Globally-Distributed Database
- Google Chubby
- Google File System
- Google Bigtable
- Google MapReduce

When asked what was the most helpful preparation technique, a large number of our interns said it was doing practice interviews with a friend. So call a friend, a relative or anyone who can act as your ‘interviewer’ and start coding!

You've got this.
We're rooting for you!