Create
Design
Code
Build
for
everyone

Google Technical Internships Interview Preparation

Agenda

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

NOTE: If you think you have been sent the wrong preparation materials, please check with your recruiter!

Recruitment Process

Recruitment Process

2 x 45-minute Technical Interviews with a Google Engineer

During the Technical interviews (usually happening over Hangouts or phone) a Google doc will be used as a virtual whiteboard for coding and algorithmic problems.

Final Evaluation Process

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 as we grow.

Intern Placement Interviews

Your application and candidate questionnaire will be shared with our intern hosts. This will result in a hangout with your potential host to discuss whether your skills and interests are a fit for the team.

Offer

We'll take a look at the additional information gathered and if your application is approved, we will make you an offer, confirm it in writing and talk to you about possible relocation, visas and starting your internship at Google!

Technical Interview Tips

Technical Interview Tips

Practice

Make sure you practice writing code in a Google Doc 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.

Clarify

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 see as the most important piece of the technological puzzle you've been presented. Clarifying questions are encouraged!

Also, **take interviewer tips seriously** because they are trying to give you helpful hints. They may ask some additional questions or share comments when you're solving a problem. Take a minute to think through these prompts -- they can help you land on a better solution!

Technical Interview Tips

Explain

Explain your thought process and decision making throughout the interview. In all of Google's interviews, our engineers are evaluating not only 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 are looking to see how you break down and approach the problem. There isn't always a "one true answer" to them and we recommend starting with a solution even if you know it is partial and sub-optimal, and work from there as it is always important to show good problem solving capabilities. We want to understand how you think. This would include ***explicitly stating and checking any assumptions*** you make in your problem solving to ensure they are reasonable. **Think out loud.**

Technical Interview Tips

Improve

Think about ways to improve the solution. Share the different options or trade-offs you are considering. **Be flexible.** In many cases, the first answer that springs to mind may need some 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 is okay. Let the interviewer know, and write code assuming some similar interface.

Technical Interview Tips

Substantiate
Make sure that you substantiate what your CV/resume says – for instance, if you list Java or Python as your key programming language, questions about this are fair game and may be asked of you.

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 pre-prepared for each interview.

Interviewing at Google Resources

Hiring Process for Students

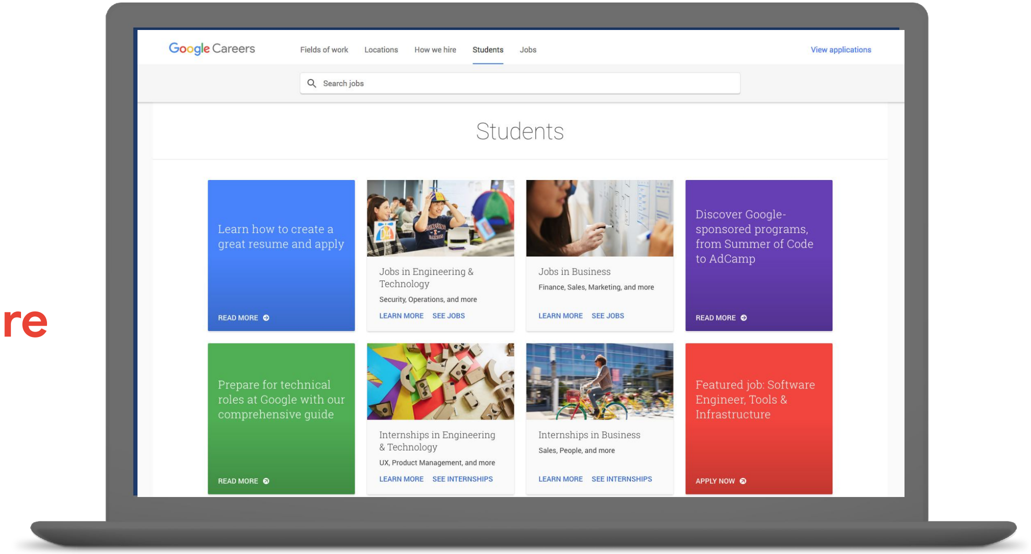
Read more here.

Interviewing at Google

Read more here.

Interview tips from Google Software Engineers

Watch video here.



Students

Learn how to create a great resume and apply

Jobs in Engineering & Technology

Security, Operations, and more

Jobs in Business

Finance, Sales, Marketing, and more

Discover Google-sponsored programs, from Summer of Code to AdCamp

Prepare for technical roles at Google with our comprehensive guide

Internships in Engineering & Technology

UX, Product Management, and more

Internships in Business

Sales, People, and more

Featured job: Software Engineer, Tools & Infrastructure

Technical Preparation & Resources

Technical Preparation

Coding

Google engineers primarily code in C++, Java, Python and Go. We ask that you use one of these languages during your interview. For technical interviews, you will be asked to write 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/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 (say, 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

Study up on as many other structures and algorithms as possible. We recommend you know about the most famous classes of NP-complete problems, such as traveling salesman and the knapsack problem. Be able to recognize them when an interviewer asks you in disguise. 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 about the space of one interview. You'll want 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

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

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 flavor of balanced binary tree, whether it's a red/black tree, a splay tree or an AVL tree. You'll want to know how it's implemented. You should know about tree traversal algorithms: BFS and DFS, and know the difference between inorder, postorder and preorder.

Min/Max Heaps
Heaps are incredibly 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.

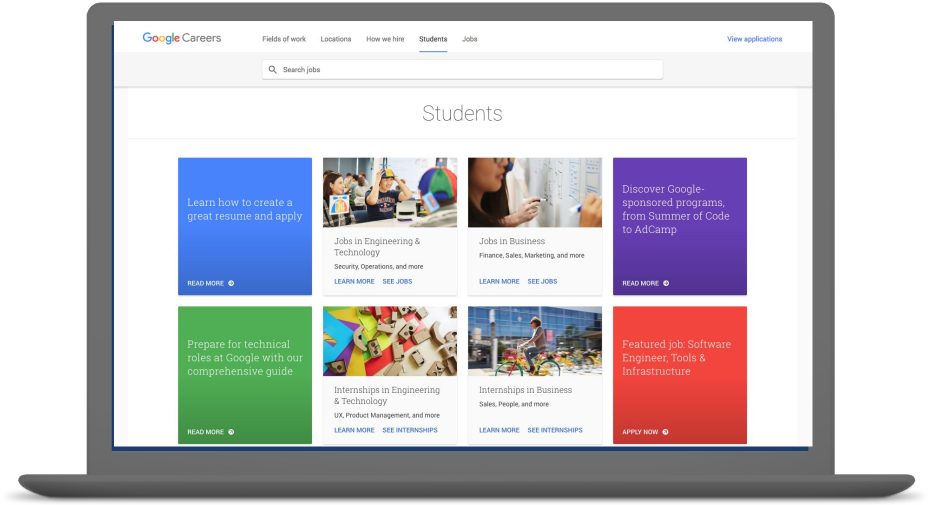
Technical Preparation Resources

Cracking the Coding Interview
Read more here.

Distributed Systems and Parallel Programming
Read more here.

Scalable Web Architecture & Distributed systems
Read more here.

Example Coding/Engineering Interview
Watch video here.



Students

Learn how to create a great resume and apply

Jobs in Engineering & Technology
Security, Operations, and more

Jobs in Business
Finance, Sales, Marketing, and more

Discover Google-sponsored programs, from Summer of Code to AdCamp

Prepare for technical roles at Google with our comprehensive guide

Internships in Engineering & Technology
UX, Product Management, and more

Internships in Business
Sales, People, and more

Featured job: Software Engineer, Tools & Infrastructure

Extra Resources

Resources recommended by Googlers

Useful articles

Dean Jackson's ACM article

Five Essential Phone Screen Questions by Steve Yegge

Books we recommend

Cracking the Coding Interview

Programming Interviews Exposed: Secrets to Landing Your Next Job

Programming Pearls

Introduction to Algorithms

Interview Practice & Resources

Practice makes perfect and interviews are no exception!

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

To practice for your interview, you may also want to try

[Project Euler]

[ACM-ICPC Live Archive]

[UVa Online Judge]

Practice resources recommended by interns

[GeeksforGeeks]

[HackerRank]

[CodeForces]

Get to know Google

Learn about Google products and technologies

About Google

How Google Search works

Google Spanner: Google's Globally-Distributed Database

Google Chubby

Industry News: Search Engine Land

Google File System

Google Bigtable

Google MapReduce

Best of luck with the process!