First Time Interviewing at Google?

Start Here.

What to Expect from your Google Interview

Over the course of 45-minutes, your interviewer is responsible for gleaning *a lot* of information about your fundamental computer science skill set. This type of environment makes a technical interview feel very different from the day-to-day of any software engineer, which is why we highly recommend that you delve deeply into the resources we provide (see the document “Interviewing at Google?”) and set aside multiple 45-minute blocks for yourself to practice before your interview date.

Interview Topics

Interview topics may cover anything on your resume, but will mostly involve coding questions, building and developing complex algorithms, and analyzing their performance characteristics, logic problems, and core computer science principles. Computer Science fundamentals are prerequisite for all engineering roles at Google, regardless of seniority, due to the complexities and global scale of the projects you would end up participating in. See the list below for examples of technical domains that you should review before interviewing with us as well as our technical communication expectations.

TECHNICAL DOMAINS

- ***Algorithm Complexity***: It's fairly critical that you understand big-O complexity analysis. Again run some practice problems to get this down in application.
- ***Coding***: Come up with solutions quickly! Time yourself during practice problems to make sure you're working at an appropriate pace, as we generally look to have about 20-30 lines of code written over the course of a 45-minute interview. Make sure your code is clean, bug-free and properly handles edge cases. If you need to write pseudo-code to gather your thoughts first, verbally tell the interviewer this is what you're doing.
- ***Sorting:*** Know how to sort. Don't do bubble-sort. You should 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.
- ***Hash tables:*** Arguably the single most important data structure known to mankind. You absolutely should know how they work. Be able to implement one using only arrays in your favorite language, in about the space of one interview.
- ***Trees:*** Know about trees; basic tree construction, traversal and manipulation algorithms. Familiarize yourself with binary trees, n-ary trees, and trie-trees. Be familiar with at least one type of balanced binary tree, whether it's a red/black tree, a splay tree or an AVL tree, and know how it's implemented. Understand tree traversal algorithms: BFS and DFS, and know the difference between inorder, postorder and preorder.
- ***Graphs:*** Graphs are really important at Google. There are 3 basic ways to represent a graph in memory (objects and pointers, matrix, and adjacency list); familiarize 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.

- ***Other data structures:*** You should study up on as many other data structures and algorithms as possible. You should especially know about the most famous classes of NP-complete problems, such as traveling salesman and the knapsack problem, and be able to recognize them when an interviewer asks you them in disguise. Find out what NP-complete means.
- ***Mathematics:*** Some interviewers ask basic discrete math questions. This is more prevalent at Google than at other companies because we are surrounded by counting problems, probability problems, and other Discrete Math 101 situations. Spend some time before the interview refreshing your memory on (or teaching yourself) the essentials of combinatorics and probability. You should be familiar with n-choose-k problems and their ilk – the more the better.
- ***Operating Systems:*** Know about processes, threads and concurrency issues. Know about locks and mutexes and semaphores and monitors and how they work. Know about deadlock and livelock and how to avoid them. Know what resources a processes needs, and a thread needs, and how context switching works, and 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
- ***Test Cases:*** come up with a variety of small test cases, beyond the examples provided by the interviewer. This will show your general fluency rather than simply your understanding of the problem at hand.

TECHNICAL COMMUNICATION

- ***Ask clarifying questions:*** Most questions your interviewer asks are under specified so you may need to ask clarifying questions before you start working on a solution. If you need to make assumptions, state those assumptions to your interviewer. Always let your interviewer know what you are thinking as they are as interested in your thought process and how you approach the problem as the solution itself. Also, if you're stuck, they may provide hints if they know what you're doing. If there is anything you don't understand - it is okay to ask your interviewer for help or clarification.
- ***Stop, define, and frame:*** When asked to provide a solution, first define and frame the problem as you see it.
- ***Stay Verbal:*** If you need to assume something - verbally check that it is a correct assumption. Be sure that you describe how you want to tackle solving each part of the question.

What now?

Now that you have a better understanding of what to expect on the day of your interview, explore our resource “Interviewing at Google?” for articles, texts, videos, and websites to support your interview preparation. Of course, always feel free to contact your recruiter if you have clarifying questions or need support of any kind. Best of luck!