Hello candidate,

Congratulations that your application has successfully proceeded to interview! We believe you are expecting to demonstrate your excellence during this interview. To help you better understand Google’s interview and get prepared, we’d like to share some interview tips with you.

Before reading all detailed information below, you should clearly note that Google is not simply looking for engineers to solve the problems they are already well-trained on; we are interested in engineers who have passion and capability to work out the answers to questions they have never come across before. Although we list many specific knowledge and skills for your reference, we don’t suggest treating them with higher priority than those not mentioned here, because the interview may cover but not limited to them. One more thing you should know about Google’s interview is that we will regard your systematic thinking, quick learning ability, verbal communication and personality as the same as or even more important than knowledge itself.

**I. What you will expect:**

45 minute technical interview with a Google software engineer. The interviewers will be interested in your knowledge of computer science principles (data structures, algorithms etc.) and how they can be used in your solutions.

**Interview topics may cover:**

● Brief self-introduction, previous projects, major accomplishments or anything on your CV. especially if you have stated that you are an expert!),

● Coding: During the interview, you’ll need to write codes on whiteboard or paper (or Google docs if phone interview).

● Building and developing complex algorithms and analyzing their performance characteristics.

● Computer Science fundamentals: Object oriented programming and design; common data structures like hash tables, stacks, arrays, etc; system-level knowledge such as operating system, network, compiler basics, etc

● Real-world problem solving: solve fairly new problems with CS knowledge and skills.

● Design questions: Sample topics: features sets, interfaces, class hierarchies, designing a system under certain constraints, simplicity and robustness, tradeoffs.

● Open-ended discussion: biggest challenges faced, best/worst designs seen, performance analysis and optimization, testing, ideas for improving existing products.

**II. What to prepare**

**Coding**

We expect **high quality, efficient, clear and compilable code without typing mistakes.** Because all engineers (at every level) collaborate throughout the Google code base, with an efficient code review process, it’s essential that every engineer works at the same high standard.

You should know at least one programming language really well, and it should preferably be C++, Java, Python or Javascript. Your need to write both logically and syntactically correct code – **COMPILABLE, NOT PSEUDOCODE**. So perhaps to practice, try writing code on paper, then put it through a compiler to make sure that your code compiles (and if it doesn't, then practice some more). Sample topics include: construct / traverse data structures, implement system routines, distill large data sets to single values, transform one data set to another.

**Data structure and algorithm**

Basic big-O complexity of space/time analysis is a must. Due to the size of the products you'll be building, it’s imperative you're comfortable with big O notation. It is important to understand the basic concepts including List, Array, Trees, Hash tables, Graphs, sorting algorithm, NPC problems, etc. In interviews, you need to demonstrate the capability to apply all these knowledge to solve problems. Sample topics include big-O analysis, sorting and hashing, handling obscenely large amounts of data. Also see topics listed under 'Coding'.

**● Sorting**: Know how to sort. Don't do bubble-sort. You should know the details of at least one nlog(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.

**● Hashtable:** 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. If you get a chance, try to study up on fancier algorithms, such as Dijkstra.

**● 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 may ask basic discrete math, linear algebra or some other math questions, so you can spend some time before the interview refreshing your memory on (or teaching yourself) it. The essentials of combinatorics and probability. You should be familiar with n-choose-k problems and their ilk - the more the better.

**Systems Design**

You need to know powers of 2, and be good and back-of- the-envelope calculations. e.g. to estimate the required number of machines for a given design. Know Google's products, and think about how you would design the back-end and also front-end. System design questions are a test of your problem solving. Sample topics include: features sets, interfaces, class hierarchies, distributed systems, designing a system under certain constraints, simplicity and robustness, tradeoffs.

**Operating Systems**

In fact the interview may cover but not limited to operating systems, also there may be other topics like network, parallel computing, compiler theory, etc. Here let’s just take operating systems for example: 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.

**Real-world problems**

Some interviewers may ask some real-world problems, they are looking for process of thought, creative solutions and being able to work out more than one way to solve a problem and 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.

**III. Where you can warm up and practice**

You’re strongly suggested to PRACTICE your coding and algorithm before the interview, below resources may be helpful:

● Coding: Google's APIs, Code Style Guide

● Data Structure and Algorithm: List of Algorithm general topics，List of Algorithms，List of

Data Structures

● Sharing by Interviewer: Video

● Practice: You may want to visit the website www.topcoder.com . If you launch the "Arena" widget and then go to the practice rooms, you can play with the problems in the first/second division as a warm up. Another site recommended highly by engineers is projecteuler.net.

These sites will expose you to programming problems that you would not normally come

across in a standard day.

**IV. How to succeed the interview**

**Speak out and communicate**

Most of the time, you'll need more information from the interviewer to analyze & answer the question to its full extent. The process of your thoughts is as important as final solution, so you are encouraged to ask more questions and speak out your thoughts.

● When asked to provide a solution, first define and frame the problem as you see it.

● If you don't understand the question- ask interviewer for help or clarification.

● If you need to assume something - check it’s a correct assumption.

● Describe how you want to tackle solving each part of the question.

● Always let your interviewer know what you are thinking as he/she will be as interested in your process of thought as your solution.

● Listen carefully - don't miss a hint if your interviewer is trying to assist you!

**Know interviewer’s expectation:**

Interviewers will be looking at the approach to questions as much as the answer:

● Does the candidate listen carefully and comprehend the question?

● Are the correct questions asked before proceeding? (important!)

● Is brute force used to solve a problem? (not good!)

● Are things assumed without first checking? (not good!)

● Are hints heard and heeded?

● Is the candidate slow to comprehend / solve problems? (not good!)

● Does the candidate enjoy finding multiple solutions before choosing the best one?

● Are new ideas and methods of tackling a problem sought?

● Is the candidate inventive and flexible in their solutions and open to new ideas?

● Can questioning move up to more complex problem solving?

**Ask Questions!**

Think about what you care the most and prepare some questions for each interview. Note: please direct to your recruiter if you have questions about the interview process, remuneration or your interview performance. We highly recommend you to do some homework about Google and our products: Google at a Glance, Google's Wikipedia Page, A Google Engineer's Career.

Thanks for your reading. In the end, there is no special requirement for your dressing code, just wear comfortably and confidently. We wish you good luck in the coming interview!