|  |
| --- |
| xxx, |
|  |  |
|  | Thanks again for taking the time to speak with me and for sending me your information. I'm excited to tell you that we would like to move forward in the process! |
|  |  |
|  | One of our coordinators will be emailing you within the next week from an @google.com domain with the date and time of your phone interview. In the meantime, I've included some preparation materials (below.) |
|  |  |
|  | Please note this will be a technical interview that will last for approximately 45 minutes. Google takes an academic approach to the interviewing process. This means that we are interested in your thought process, your approach to problem solving as well as your coding abilities. You may be asked questions that relate to technical knowledge, algorithms, coding, performance, how to test solutions, and perhaps your interest in Google products. The best advice that I can give you is to treat the interview like a conversation, talk through the problems, and please feel free to ask the interviewer if you are not clear with any of the questions. |
|  |  |
|  |  |
|  | IMPORTANT NOTE: You'll need a computer with internet access for the interviews to use Google Docs -- a web-based word processor which lets you share and collaborate your work online. Once you click the link, you'll be able to use the Google Doc along with the interviewer. Please note that you do not need a Gmail account to use these documents - just click on the link provided and you will be able to edit the document. |
|  |  |
|  |  |
|  | Here are a few tips for interviewing with Google as well as some links to check out: |
|  |  |
|  | The interviewer(s) will be interested in the specifics of your past projects, implementations and how you arrived at your conclusions. The name of this game is Interaction!! |
|  |  |
|  | We strongly encourage you to review the following videos. The videos feature Googlers who have been both interviewers and interviewees sharing their unique perspective on how to prepare for Google's technical interviews! The principles are applicable to a technical phone interview as well. |
|  |  |
|  | Preparing for Google's Technical Interview #1: |
|  |  |
|  | Preparing for Google’s Technical Interview #2 |
|  |  |
|  | Review your resume and be prepare to discuss your experiences and past projects! Often times it will help drive the discussion and gives you an opportunity to share situations where you solved a complex problem. |
|  |  |
|  | Your background is important, so utilize it to your advantage by sharing interesting points that can help you and the engineer work through the discussion as efficient as possible. |
|  |  |
|  | Our interviews are very technical!!! It will be based on Computer Science fundamentals. Here are some TIPS DIRECTLY FROM OUR ENGINEERS on the main areas software engineers should prepare to be successful in interviewing with Google: |
|  |  |
|  | 1.) Algorithm Complexity: You need to know Big-O. If you struggle with basic big-O complexity analysis, then you are almost guaranteed not to get hired. For more information on Algorithms you can visit: http://www.topcoder.com/tc?module=Static&d1=tutorials&d2=alg\_index |
|  |  |
|  |  |
|  | 2.) Coding: You should know at least one programming language really well, and it should preferably be C++ or Java. C# is OK too, since it's pretty similar to Java. You will be expected to write some code in at least some of your interviews. You will be expected to know a fair amount of detail about your favorite programming language. |
|  | \*Strongly recommended\* for information on Coding: Programming Interviews Exposed; Secrets to landing your next job by John Monagan and Noah Suojanen (Wiley Computer Publishing) |
|  | http://www.wiley.com/WileyCDA/WileyTitle/productCd-047012167X.html |
|  |  |
|  | 3.) System Design: http://research.google.com/pubs/DistributedSystemsandParallelComputing.html |
|  | Google File System http://research.google.com/archive/gfs.html |
|  | Google Bigtable http://research.google.com/archive/bigtable.html |
|  | Google MapReduce http://research.google.com/archive/mapreduce.html |
|  |  |
|  | 4.) 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. |
|  |  |
|  | 5.) Hashtables: 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. |
|  |  |
|  | 6.) 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. |
|  |  |
|  | 7.) 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 and A\*. |
|  |  |
|  | 8.) 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. |
|  |  |
|  | 9.) 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. |
|  |  |
|  | 10.) 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. |
|  |  |
|  | For information on System Design: |
|  | http://research.google.com/pubs/DistributedSystemsandParallelComputing.html |
|  |  |
|  | 11.) Also, you can review some of our research publications: http://research.google.com/pubs/papers.html |
|  | and paper on Google’s Hybrid Approach to Research: |
|  | http://static.googleusercontent.com/media/research.google.com/en/us/pubs/archive/38149.pdf |
|  |  |
|  |  |
|  | If you've never experienced an interview where you are expected to produce live code and talk through it. Pull out a blank sheet of paper and start writing code, while talking it through to a friend, family member, or to yourself! This will build a high level of comfort as you get on the phone with our engineers and you will feel the conversation go smoother. |
|  |  |
|  | Many of the questions asked in Google interviews are open-ended because our engineers are looking to see how you engage the problem. Be sure to talk out-loud your thought process about the questions you are asked, as well as your approach to problems and solutions. Ask specific questions if you need more clarification. Our engineers are not looking for the right answer, but the right approach and methods of thinking, within relevance, say EVERYTHING you are thinking!! |
|  |  |
|  | As you prepare, build toward thinking about ways to improve the solution you'll present. 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. A brute force explanation will be received less well than taking time to compose a more efficient solution. 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 leap to your mind as the most important piece of the technological puzzle you've been presented. |
|  |  |
|  | Also, one of our senior engineers has recommended candidates read Programming Interviews Exposed: Secrets to Landing Your Next Job by John Mongan and Noah Suojanen to give you an edge. It's a good book to have in general. Two other books that may be helpful are: "Programming Pearls" by Jon Bentley - Programming questions that get you thinking outside of the box; and lastly, "Cormen/Leiserson/Rivest/Stein: Introduction to Algorithms" or the CLR textbook. |
|  |  |
|  | Please take the opportunity to review the following links. They provide additional insights to interviewing/working at Google. Googlers thrive in small, focused teams and high-energy environments, believe in the ability of technology to change the world, and are as passionate about their lives as they are about their work. |
|  |  |
|  | Tips for Interviewing at Google |
|  |  |
|  | Google Products |
|  |  |
|  | Five Essential Phone Screen Questions by Steve Yegge (Google Engineer) |
|  |  |
|  | Types of Algorithm Questions Google Asks: TopCoder Tutorials |
|  |  |
|  | The Official Google Blog: “Baby steps to a new job” by Gretta Cook (Google Engineer) |
|  |  |
|  | “How to Get Hired” by Dan Kegel (Google Engineer) |
|  |  |
|  | Regards, |
|  |  |
|  | yyy |