Google Criterias and Selection for Engineers +8

What should I expect in a Software Engineer interview at Google and how should I prepare?

Answer Request Follow 1.8k Comments 2+ Downvote

Answer Wiki

Here are a few useful links referred in the answers:

1. Problem Solving in Data Structures & Algorithms Using Java
2. Grokking the System Design Interview
3. LeetCode
4. Coderust 2.0: Faster Coding Interview Preparation using Interactive Visualizations
5. Programming Interview Questions | CareerCup
6. GeeksforGeeks | A computer science portal for geeks
7. HiredInTech's Training Camp for Coding Interviews
8. http://highscalability.com/
9. GeekyPrep.com. Join us, Prepare for Interviews, Get Hired!!
10. Practice anonymous coding interviews with Engineers from Google, Facebook, Palantir, etc. - Refdash
11. A structured, rigorous course: Interview Kickstart (remote or local)

51 Answers

Anonymous

Updated Oct 29, 2014 · Upvoted by Alexandre Duarte, Software Engineer at Google

Great answers here. But I joined Google one month back as Software Engineer and just wanted to share my experience also.

Recruiting Process:

I had two telephonic interviews and five onsite interviews. Each telephonic round is knock out but onsite interviews are not. Each onsite interviewer gives independent feedback about the candidate and these are processed later after all five interviews. So you can start fresh even if you don't perform that well in one interview. And as mentioned in "Cracking the coding interview", it's really true that a packet with scores of 3.6, 3.1, 3.1 and 2.6 is better than all 3.1s out of 5. Google off campus recruitment takes time (mine took 6 months) and good thing about it is you get enough time (2 weeks in avg) before each telephonic interview and the onsite interview day. So be patient and it's really worth the wait :P

Interviewers' Expectation:

1. First of all, for S/w Engineering role in Google, most of the questions are technical and it will be the obvious expectation **that you KNOW algorithms and coding**. And also you should know the core CS subjects like Operating System, Computer Network, Distributed systems (overall basic ideas).

2. Interviewers can give you vague questions and that's intentional. You should always ask relevant questions to make it more clear without making assumptions on your own. Make the problem clear before directly going to solve

Related Questions

How does Google interview candidates for Engineering Director-level positions?

How helpful was the paid program from Interview Cake in your interview preparation?

What are some of the questions asked in Google interviews?

What is the interview process like at Google? What are the people like?

Can an average person pass Google's Software Developer interview?

Which is the best site for preparing for Google interviews?

What are engineering manager interviews at Facebook or Google like?

What topics are covered in the Google phone interview? I hear it's 45 minutes and you have to write code? Can anyone provide more information?

If you had a software engineer interview with Google tomorrow, what would you do today?

How was your interview at Google DeepMind for a software engineering position, and how was it different from regular Google tech interviews?

+ Ask New Question

More Related Questions

Question Stats

1,823 Public Followers

762,777 Views

Last Asked Dec 22

12 Merged Questions

Edits

3. You can always start with the naive solution first but make it brief. Think in different directions and try to give multiple solutions until you reach the most optimal one. **And think out loud**. Otherwise interviewer will have no clue what you are thinking.

4. In Google, scalability is always important. So expect design questions which will be ambiguous real-world problem for large size system. They are looking for process of thought and how you break things down to ultimately get to creative and scalable solutions.

5. And of course after you reach to the good solutions, expect to write the clean code of your algorithm (some parts of it sometimes for the big problems). You can choose any programming language you want but it should be clear and intuitive so that interviewers can understand it easily.

Some tips for preparation:

1. Go through Cormen Algorithm book deeply and make very strong understanding of algorithm and data structure. Make sure you have a good understanding in complexity analysis because you may have to answer the complexity of your solutions pretty much every single time.

2. Make sure you can implement your algorithm no matter what. Keep coding and practice in whiteboard. You have to be fast and clear.

3. Keep solving good problems. Think about the solutions in different approach. Good resources will be GeeksforGeeks - A computer science portal for geeks , Programming Interview Questions | CareerCup , Cracking the coding interview etc.

4. Don't expect known questions in Google interviews. So don't look at the answers immediately when practicing. Try to solve it yourself. It's not about giving right or wrong answers in interviews. It's about your problem solving ability, how you approach to a problem in real time. So try to increase that by practicing.

5. Even if you can solve the problem, it won't be counted until and unless the interviewer understands your logic properly. Always try to express yourself clearly and think out loud. That will clear your progress to the interviewer. Communication with interviewer becomes very important for deliberately ambiguous real-world problems given in Google interviews.

6. Last but not the least, the most important thing is to **keep your calm.** Being relaxed helps you the most solving problems in real time. No matter what the result is, in onsite interviews you will have a great day sharing your thoughts and ideas with great minds which will help you in future.

Good luck! :)

Dinesh Venkata
After that are there challenges in day to day work?

1 more comments from Mansi Jain

Gayle Laakmann McDowell, Author of Cracking the {Coding Interview, Tech Career, PM Interview}
Updated Apr 25, 2012 · Upvoted by Alexandre Duarte, Software Engineer at Google

First of all, don't believe a lot of the hyped up articles - even the ones you see on Business Insider and the Wall Street Journal. They're perpetuating the same silly myths to get eyeballs, but it's all BS.

So now that that's out of the way... here's what to expect:

- There's much less structure behind the interview process than people think. The first interview is equivalent to the last one. If you find that the first one is easier than the last interviews, that's purely coincidence.
- Phone interviews might be slightly easier than onsite interviews (mostly because many interviewers don't want to have you get stuck on a really hard question when you haven't passed any interviews yet), but you shouldn't expect a substantial difference.
- If you pass the 2 (average) phone interviews, you'll be brought in for 4 or 5 onsite interviews.
- Other than the lunch interview, the onsite interviews will be technical. You will be asked a mix of coding and algorithm questions. These might somewhat related to your background, but they usually aren't. Interviewers tend to have a favorite 5 questions or so, and tend to ask them to whoever they interview.

Now for some advice:

- Prepare with actual interview questions that candidates have been asked (that is, don't read "top 10 lists" - they're usually overhyped, and sometimes outright lies). CareerCup.com has thousands of software engineering questions asked at major companies.
- Practice by writing code on paper (NOT on a computer). Make your code perfect - every last semicolon should be there. Test your code by hand as well. Only once you've done everything possible to perfect your code, then type it into a compiler and see how you did.
- Don't worrry too much about fancy algorithms like Dijkstra's. Complex algorithms rarely come up because, frankly, your interviewers probably don't remember them too well. Again, look at actual interview questions and see how often this stuff comes up. If you find that you're missing some knowledge required to solve interview questions, then read up on it. Most people with a strong background in CS will not need to re-learn many things.

you, but you can't *only* write pseudocode). Again, every last semi-colon (although your interviewer will probably overlook missing a few semi-colons :)).

- Once you're done writing code in your interview, test it. You wouldn't submit code at work without testing it, so why do it in your interview?
- Think out loud. Show your interviewer how you're thinking about the problem.
- Don't be afraid to throw out a brute force solution in the beginning, but you should try to improve your solution as well. Think about the time *and* space complexity and see if you can improve those.

You should also check out my book, Cracking the Coding Interview: 150 Programming Questions and Answers (http://www.amazon.com/dp/098478280X). It focused on the strategies for mastering the interview process at the top tech companies.

148.1k Views · View Upvoters

Venkateswara Rao Sanaka
Gayle, you said "No pseudocode". Isn't true that pseudo code helps in organising our t...

2 more comments from Dinesh Kumar, Jayshekar Harkar

Moishe Lettvin, former Software Engineer at Google (2006-2013)
Answered Jan 7, 2012 · Upvoted by Alexandre Duarte, Software Engineer at Google and Bob See, was Principal Recruiter, Google Engineering 2005-2014

I teach interview training classes at Google and I've done over 180 interviews here. I strongly believe that the more candidates know about what to expect, the better it is for the candidates and for Google: it helps remove the "noise" of nervousness and lets us get at candidates' abilities.

Generally you should be prepared to:

- write lots of code
- that demonstrates that you can apply algorithmic knowledge to solve
- problems that may or may not be well-defined

In addition, especially if you're more senior, you should expect to be asked some high-level design questions that will attempt to get at your knowledge of how to scale applications, your sense of what's hard about some arbitrary large-scale problem and/or how to suss out the hard parts. These don't necessarily presume experience in exactly these areas but what they're trying to get at is: can you break things down and choose problems to attack systematically.

described "biggest weakness" -- what I want to know is: does the candidate know how computers *work*, and can they use that knowledge to write code or design systems to solve problems well. Maybe sometimes a candidate doesn't know the "perfect" data structure for a problem, but if he or she can build up that data structure from first principles, that is extremely valuable. Or, maybe the candidate has a wealth of experience but can quickly see what's relevant and discard that which isn't.

As I mentioned above, you'll be writing lots of code on a whiteboard. You'll probably be nervous. Coding on a whiteboard is unnatural and a specific skill that's orthogonal to day-to-day software engineer tasks, so I recommend practicing it. Again, it's about eliminating the "noise" of writing too big or too small or getting your hands messy from erasing with them or whatever so that the interviewer can see the signal of how you're solving the problem.

Get together with some friends, Google "Google Interview Questions," and ask each other questions. Get used to not only coding on the whiteboard, but thinking on the whiteboard -- draw pictures, pseudocode, etc. -- anything that can give your interviewer an idea of what you're thinking. This will let him or her give you hints, maybe nudge you if you're on the right (or wrong) track, and just give them *data,* which is way more valuable than silence, even if you've got a crystalline vision in your head -- if the interviewer doesn't know that, it doesn't help you. And the more you practice the more natural the interview itself will feel.

One note: if you've heard a particular question before, just tell the interviewer. You'll get points for honesty, the interviewer may have you solve it anyway, and you won't run the risk of getting called out for not mentioning it.

There are lots of great resources out there. Gayle's book is great. Also see:
Steve Yegge's excellent advice: http://steve-yegge.blogspot.com/...

MIT's course: http://courses.csail.mit.edu/iap...

The last thing I would say, and it's hard to implement, is have fun. Sometimes you'll get a question and an interviewer that just "click" with you: run with that! Hopefully some of the questions you get asked will just be neat and you'll enjoy solving them. Look forward to that, even if you get a question or interviewer that you don't "click" with. In the course of 90 minutes during my interviews here I went from thinking "oh my god I'm the biggest moran in the world what am I doing here" to "oh wow I didn't know I could have this much fun with a whiteboard and another nerd!"

278.2k Views · View Upvoters

Krish Munot, interviewed at Google
Updated Aug 16, 2015

comparable (if not tougher) to that of Google. Hence whatever is written here is a broad framework of the topics that are required to be known.

Also before you begin to read the below mentioned "formidable" list of topics, I would suggest that you read the following two links:-
ABC: Always Be Coding
Four Steps to Google, Without a Degree

Now moving on to the technical topics to be covered:-

Programming Languages:

No specific programming language is required interviewing for a technical position with these companies, but familiarity with a prominent language is generally a prerequisite for success. Not only should you be familiar with the syntax of a language like Java, Python, C#, C/C++, or Ruby, you should be familiar with some of the languages' nuances, such as how memory management works, or the most commonly used collections or libraries, etc.

Data Structures:

Most of the work that is done in these companies involves storing and providing access to data in efficient ways. This necessitates a very strong background in data structures. Understanding of the inner workings of common data structures and ability to compare and contrast their usage in various applications is expected. You will be expected to know the runtimes for common operations as well as how they use memory.

Algorithms:

Your interview with these companies will not be focused on rote memorization of algorithms; however, having a good understanding of the most common algorithms will likely make solving some of the questions they ask a lot easier. Consider reviewing traversals, divide and conquer, and any other common algorithms you feel might be worth brushing up on. For example, it might be good to know how and when to use a breadth-first search versus a depth-first search, and what the tradeoffs are. Knowing the runtimes, theoretical limitations, and basic implementation strategies of different classes of algorithms is more important than memorizing the specific details of any given algorithm.

Coding

Expect to be asked to write syntactically correct code—no pseudo code. If you feel a bit rusty coding without an IDE or coding in a specific language, it's probably a good idea to dust off the cobwebs and get comfortable coding with a pen and paper. The most important thing a Software Development Engineer does at these companies is write scalable, robust, and well-tested code. These are the main criteria by which your code will be evaluated, so make sure that you check for edge cases and validate that no bad input can slip through. A few missed commas or typos here and there aren't that big of a deal, but the goal is to write code that's as close to production ready as possible. This is your chance to show off your coding ability. Also make sure that your code is highly efficient

yourself this question - "How can I make this code simpler?"

Object-Oriented Design

Good design is paramount to extensible, bug free, long-lived code. It's possible to solve any given software problem in an almost limitless number of ways, but when software needs to be extensible and maintainable, good software design is critical to success. Using Object-oriented design best practices is one way to build lasting software.
You should have a working knowledge of a few common and useful design patterns as well as know how to write software in an object-oriented way, with appropriate use of inheritance and aggregation. You probably won't be asked to describe the details of how specific design patterns work, but expect to have to defend your design choices.

Databases

Most of the software that these companies write is backed by a data store, somewhere. Many of the challenges they face arise when figuring out how to most efficiently retrieve or store data for future use. Many of these companies have been at the forefront of the relational/non-relational DB movement.
The more you know about how relational (SQL) and non-relational (NoSQL) databases work and what tradeoffs exist between them, the better prepared you will be. However, any particular level of expertise is generally not assumed.

Operating Systems

You won't need to know how to build your own operating system from scratch, but you should be familiar with some OS topics that can affect code performance, such as: memory management, processes, threads, synchronization, paging, and multithreading.

Internet Topics

These companies do a lot of business online, and hence expect their engineers to be familiar with at least the basics of how the internet works. You might want to brush up on how browsers work at a high level, from DNS lookups and TCP/IP, to socket connections. A solid understanding of the fundamentals of how the web works is a requirement.

In addition to these, some solid project work (atleast one on each topic) and other co-curricular achievements would be a huge plus in your resume.
As far as finding the right opportunities is concerned, I would recommend the reading the book written by Gayle Laakmann McDowell - 'Cracking the Coding Interview' as they are the best source to understand how the recruitment program at these companies work and how to make it work in your favor along with several practice problems that are of a similar difficulty level of which that they ask in such companies.

Also, one thing that you might have noticed after reading the two articles is that these companies respect impact. So, once you have understood and mastered the above said topics, pick up a field that you love and try to create a software that make other people's lives a little better.

Mark Ali, Interned at Google, Facebook and got my full-time offer
Updated Aug 14, 2015

I believe that there are already tons of advice on how to prepare software engineer interview at Google, but I'd like to share my experience how did I crack it **IN 2 MONTHS** since definitely there are certain ways that can make your life easier.

- **Be familiar with basic data structure and algorithm**. I can't emphasize more about this point since it is the most fundamental thing for a software engineer interview. If you fail to get a good grasp of those basic data structures you learnt at school, you just failed the whole interview. I'm not exaggerating, once you've been thru several technical interviews, you'll realize how important it is. Books about data structure and algorithm are everywhere, do make sure you are very clear about basic stuffs like binary tree, queue, stack, linked list and also be fast at estimating efficiency of your code because most of the interviewers will ask time/space complexity of your code.
- Practice writing code on whiteboard. This is what most people ignores. It looks quite simple at first glance, right? But it isn't once you try it. You'll miss so much about those fancy shortcuts on your favorite text editors and IDEs and what's more, it's so inconvenient to modify the code like inserting another piece of codes in between. But you have to get over it as most of the real interviews will ask you to write SOLID code on whiteboard. It doesn't need to be compiled, but it should be almost there. No pseudo code! **This is even true for Google interview** as during the onsite interviews you'll be asked to write a bunch of solid code on whiteboard and discuss with interviewers about its efficiency.
- Practice with real interview questions for Google. Since you already have a target, it isn't hard for you to get some real questions from past Google interviews. In fact, there are tons of them online and it's almost impossible for you to finish all. The point here is not to expect having the same question in your interview (though it's possible), but to get an idea about what kind of questions Google likes to ask, how difficult it is in general, and find out your weak point. For instance, if you keep failing on binary tree questions, you should go check your textbook and do some google search about it.
- **Practice with MOCK INTERVIEWS**. It's a great experience for you to practice in a way where you can't fail. You can do this with your friends and interview them back. I also got my mock interview fromhttp://www.gainlo.co whose interviewers are working at Google, Facebook etc. and gave me tons of feedbacks.

Personally mock interview is the most effective approach I've ever had because you will have totally different feeling when thinking and solving problems in front of a person. You'll be nervous, and you may fail even at the simplest question.

However a mock interviewer will help you improve in every way especially he/she is experienced.

All in all, practice makes perfect. It's never too late to start preparing for your interview and it's always worthwhile to spend time on it.

43.9k Views · View Upvoters

Upvote 183 Downvote

Add a comment... Recommended All

Lokesh Agarwal, I have taken many and given even more interviews! I know a wee bit there..

Answered Aug 30, 2015

A2A.

There is nothing more in terms of specifics that I can tell you which has already not been mentioned in all the answers already written. And I mean that, I went through a lot of answers already posted and they incredibly good answers. There is one thing I can tell you though..

Don't be afraid to work hard. Don't give up. Don't be afraid of putting in the time you need to for getting everything right. If u don't understand something read it again, trace your code on whiteboard and debug it. Think of your brain as the computer executing the code and go through it. It will take time. Be patient. It will be frustrating when u can't answer something. Be patient and try again, and then try some more. Stay dedicated and focussed until it becomes 2nd nature to you. When you are that familiar, then you have reached a point where your interviews will turn into brainstorming sessions and you will have fun, and do well.

4k Views · View Upvoters · Answer requested by Ayush Shrimali

Upvote 15 Downvote

Add a comment... Recommended All

Erik Corry

Answered Dec 10, 2014

Lots of good tips here. I interview someone every week at Google. My advice:

If you put a language on your resume you may be asked to code in it. Especially if you say you are proficient and it's C++, Java or Python.

It's often a good idea to start with a simple, slow algorithm that works, and refine it on the whiteboard before you start coding. The interviewer isn't looking for you to just magically produce the perfect algorithm, they are interested in seeing you think.

Here is what I like to see in an interviewee:

* Asks clarifying questions
* Thinks aloud and listens when I try to give hints
* Firm grip on programming language fundamentals like pointers, recursion/stacks, scope, memory allocation.
* Firm grip on data structures like which operations are O(1) vs. O(n) for

The lunch interview isn't a real interview. As far as I know the lunch partner never writes a report. It's your chance to learn about what it's like to work at the office or live in the city.

The first phone interview is normally with a recruiter. The second is with a software engineer and will probably be a lot harder. He/she is trying to avoid bringing you on site if you have no chance of getting the job. That would just waste a lot of people's time, including your own.

Good luck!

Ash Murthy, HuffPo blogger, Ex-Googler, Interview coach at www.ProgrammingInterviewPrep.com

Updated May 28, 2015

Please note that I am answering a slightly different question, "what to expect in a software engineer interview at top software companies?"

I am asked this question very often, and I am not surprised. 100's of websites (careercup, geekforgeek, leetcode to name a few), dozens of books (CLR, algorithm design manual, cracking a programming interview) and thousands of questions. It certainly does get intimidating and confusing.

The crux of programming interviews is essentially fundamental data structures and algorithms. From an earliest post, Hacking the Programming Interview - 1 by Ash Murthy on Random Rants, I re-quote my checklist of programming interview topics. Any of this is fair game :

- **Proficiency in programming language(s) of your choice**
- **Programming questions** :
- **General computer science concepts**

operating systems (thread vs process, synchronization, deadlock etc), computer networks (IP packet, IP addressing, routing, TCP packet, protocol - sliding window, three way handshake, congestion control, man in the middle attack, TCP vs UDP, HTTP - pipelining, GET vs POST)

- ***object oriented design* and design patterns**

Factory method, Lazy initiation, Object pool, Singleton, Decorator, Publish/Subscribe, State, Monitor

- ***Big-O notations*** (Unable to guess the complexity of your solution is as bad as not having a solution)
- **Sorting** (heap, merge, quick. Don't use bubble sort.)
- **Data -Structures**

Hash
LinkedList
Trees - complexity and use of balanced search tree
Tries

- **Design questions**
- **Recursion**
- **Fundamental algorithms** such as divide and conquer, greedy and ***dynamic programming*** and common problems such as Knapsack, Travelling salesman and so on. Here is a specific list of Important Algorithms to master to solve programming puzzles.
- **Other Helpful to know topics**

Dijkstra, A*
Heuristic/approximate algorithms for NP hard problems
basics of randomized algorithms (randomized quick sort, local maxima and related AI concepts)
Map Reduce
Distributed and parallel computing (and distributed problem solving for large scale problems)

Once you have a strong grasp of the above fundamentals, start practicing interview questions, depending on the level of company that you want to crack.

If you don't know where to get started or would like an organized preparatory course, reach out to either of the following :
1)**Interview Prep (prepinterview@yahoo.com)**
2) Interview Kickstart

8.6k Views · View Upvoters · Answer requested by Praveen Karkhile

Upvote 33 Downvote

Add a comment... Recommended All

Murtaza Aliakbar, Been there, done that.
Updated Nov 4, 2010

I'm going to paste here the best advice I received, from a friend who works at Google as a Level 5 Engineer on the Google Search Team.

I followed the advice, and it worked. :)

> Heard google is coming to campus, so I decided to give some idea about our interview process. I am sure most of you will be knowing about the interview process already ..
>
> Google Interview Process :
>
> - 4-5 rounds. First 2 are almost similar. Later rounds are more difficult.
> - Most questions are basic data structures, algorithms, coding your solution, discussion about complexities (time, memory) and basic computer fundamentals.
> - Interviewers like clarifying questions e.g. if working on some string do we consider all unicode strings [Trie will go crazy in such cases].
> - Come to a working solution and iterate over there, rather than taking much time and coming with perfect solution.

- If you don't do well in one question, don't panic try better for next question. But don't say to the interviewer 'I don't know this question, ask something else' :)
- While writing code, mix algorithm style writing. e.g. // int a = max element in heap. [Complete it when interviewers ask for it]
- In General no puzzles will be asked. So don't waste time on those.
- For software test positions, questions will be based from testing too-- think-- how to break something. e.g. to test gmail login you may put whole page in user name or password, leave empty fields, just put password, different case, extra space etc.

If you are not allowed for written test based on your GPA score, ask a senior staff to recommend. Google always respects staff recommendations.

Some common areas :
1. Hash, BST, Linked List, Array, Sorting algos, Searching, String search etc.
2. Linux, deadlocks, compilation, networking may be asked. But for new grads it will be mainly data structure, coding, and algorithms.
3. For software test positions, read about unit test, regression test, load test etc.

Google for 'Google interview questions' you will get plenty of them online.

These are just guideline and don't scold me if they ask something else and nothing out of these :) So prepare well, hope to see of you at Google.

20.8k Views · View Upvoters

Upvote 68 Downvote

Gayle Laakmann McDowell, Author of Cracking the {Coding Interview, Tech Career, PM Interview}
Most of this is correct and good advice. A few corrections:...

Sanket Dialani, Co-founder at GeekyPrep.com,Ex-SDE at Amazon.com, BITS-Pilani CSE 08-12
Answered Oct 26

Dynamic Programming questions are the favourites to be asked by Google Interviewers. Some other common problems including DP (both easy and moderate) whose variations are asked at **Google** interviews :

- Remove Alternate Duplicate characters from a char array you have to do it in Place.Like keeping only the odd occurences of each character.

```
Example: Input: “you got beautiful eyes”
Output: ”you gtbeaiful es”
Allowed Time Complexity was O(n) and Space Complexity was O(1)
```

- In a file there are 1 million words . Find 10 most frequent words in that file.
- Clone a linked list with next and random pointer
- Serialise and Deserialise a linked list with next and random pointer.
- How will you implement linked list with 1 million nodes? How will you access 999999 th node? Give some optimal design strategy and implementation.

- Remove duplicates from string in place in O(n).
- Given a matrix of characters and a word.
  you have to count the number of occurrences of that word in that matrix. you can move to any of the eight valid directions from current position.
- You are given an string as input which represents a path. You have to normalize that path inplace(NO EXTRA SPACE).

```
e.g.         input : "\a\b\c\..\..\file.txt"
output: "\a\file.txt"
```

- Given two sorted arrays (with repetitive elements) find the kth minimum number from both arrays.
- Given two linked lists both represent a number. Create a linked list that contains its sum.
- Given a binary search tree , print the path which has the sum equal to k and has minimum hops. i.e if there are multiple paths with the sum equal to k then print the path with minimum number of nodes.
- Find the nth number that contains the digit k or is divisible by k. (2 <= k <= 9)
- Given a directed graph. Construct another graph from given graph such that if path exists from vertices A to vertices B and from B to C, then path from A to C and from C to A also should exists.
- Given an array, arrange the elements such that the number formed by concatenating the elements is highest.
  `E.g.: input = [9, 93, 24, 6], the output should be: [9,93,6,24]. This is because if you concatenate all the numbers, 993624 is the highest number that can be formed.`
- Given a string, find the longest substring which is palindrome.
- Given that integers are read from a data stream. Find median of elements read so for in efficient way. For simplicity assume there are no duplicates.
- Design a stack which holds an integer value such that getMinimum() function should return the minimum element in the stack. Implement popMin() function which would pop minimum element from the original stack.
- Given a set of intervals like 5-10, 15-20, 25-40, 30-45, 50-100. Find the ith smallest number in these intervals. Assume there are no duplicate numbers.

```
e.g:  1st smallest number = 5     6th smallest number = 10    7th smallest numb
```

- Given a string example : shoppingwithflipkartiseasy, Now we are given this string and a dictionary containing valid words , now we need to break the sentence into words separated by space. Output : shopping with flipkart is easy
- Given a series 2,3,4,5,6,8,9,10,....., here in this series all the numbers are present which have factors only and only either 2,3 or 5. Need to write a node to generate nth number for the series . With best approach and complexity

sum of edges.

- Given a sentence and a set of characters. Find the minimum window within which the set of characters can be found in the sentence in any order.
- You are given a string of 0's and 1's you have to find the number of substrings in the string which starts and end with a 1.

```
eg : input : 0010110010
output : 6
```

- You are given a mapping like a -> 1, b-> 2... z-> 26. You have to print all possible combinations of a given number using the above information.

```
eg : input : 121
output : aba,la,au
```

- Given a dictionary of 50,000 words. Given a phrase without spaces, add spaces to make it a proper sentence.

```
e.g:input:  thequickbrownfoxjumpoverlazydog
output: the quick brown fox jump over lazy dog
```

- Get the next bigger number using the same digits of a number.
  `Eg, For 123456, next number would be 123465`
- Whether a number is a power of 2 without using a loop.

For more interview experiences and problems asked at Google interview, visit GeekyPrep.com. Join us, Prepare for Interviews, Get Hired!!

Good luck!

6.6k Views · View Upvoters

Upvote 33 Downvote

Add a comment... Recommended All

Soham Mehta, Founder, InterviewKickstart.com.
Answered May 16, 2017

7 year old question, but still as relevant! Thanks for A2A Vincenzo Vicidomini.

Google SE interview will depend on the level you are being interviewed at.

Level-3 (or SE-II, there is no SE-I):

This is the level where most entry level engineers are hired at, from BS or MS.

There are 4 or 5 onsite rounds at this level. They are nearly all about coding of DS/Algos. For people on the cusp of L3 and L4 (below), they may throw in a design question, but usually not.

Level-4 (or SE-III):

This is for candidates with BS + 8 years, MS + 5 years, PhD + 2 years. Approximately.

At this level, expect 4 or 5 onsite rounds also, but certainly at least one Systems Design question.

Level-5 (or Senior SE)

At this level, expect the same 4 or 5 onsite rounds, and two Systems Design questions.

Level-6 (Staff Engineer)

Level-7 (Senior Staff Engineer)

Level-8 (Principal Engineer)

Direct Hiring at levels 6, 7 and 8 is much rarer/harder than hiring 3 through 5. Here, the interview process, besides coding and design, also has a component of the domain you are coming in from.

It is rare to get hired at Staff and Principal directly at many other companies, not just Google. Several reasons for that, which are out of scope of discussion here.

Level-9 (Distinguished Engineer)

Level-10 (Google Fellow)

Nearly impossible to be hired directly here.

Jeff Dean (Senior Google Fellow, added 2013)

Enough said.

About preparation

The bar for clearing SE interviews these days, has gone up considerably compared to when this question was asked. Preparation these days takes at least 3 months for most candidates, especially if you have never had a rigorous course on DS/Algos in your college or elsewhere.

There are many ways to prepare e.g. There are websites, there are books and there are courses.

If you are a CS student in college, then the best way is to form a study group in school, and do online prep, like Leetcode. You're targeting Level3, so you won't need Systems Design stuff.

If you are a working professional, then your biggest problem is motivation and carving out the time to prepare. I'd highly recommend taking a course for it. Much like going to a gym.

A course has many other benefits e.g. being taught by engineers who have done hundreds of interviews at G, or getting continuous feedback, or having a motivated group of peers, or guidance on offer negotiations etc.

Click here to get to know us better: Coding Interview Preparation Bootcamp, Large Scale Systems Design Interview Preparation Bootcamp

Sanket Agarwal, Googler
Answered Jan 10, 2012

The first thing to understand it that, you might be a great match to their profile -- you might know more about their products and technology, but with my experience I have come to know that all that matters is that you are `smart`. To judge that, companies have tactically used rigorous Algorithm tests as interviews. The jist is -- you might be awesome, but you need to clear a rather stiff algorithm test to get through. If you are a fresher then it matters all the more.

I religiously followed Algorithms by Cormen, perhaps the best treatise on introductory techniques. I would suggest you to go through the core topics:

- Sorting
- Graph Algo.
- Dynamic Programming
- Red-Black Trees and Application
- Hashing

Note that, just reading them is not enough -- you should thorough yourself with all questions in between and most importantly at the back. Once you are done with the book, you'll find that about 50% of interview questions are just sub parts (typically the tough subparts :P) of very standard Cormen problem sets.

It helped me clear the interviews, hope it helps you :)!

Ahmad Malik, Conducted 200+ interviews, software engineer for 15 years
Answered Apr 19, 2017

There is usually a phone screen followed by a full interview loop onsite at Google. I've seen people got their phone screen waived off with a good internal reference. Basically, there are three parts to a software engineering interview at Google, try to focus on all of them for your preparation:

1. **Coding interview:** Either on a whiteboard or on an actual computer. This has become a prerequisite for getting an offer. If you cannot code/solve the problem (or have some serious bugs in your code), it's quite difficult to get an offer. Good resources to prepare for this kind of interview: Coderust 2.0 . You should expect at least two coding interviews in a loop.
2. **System Design Interview:** This tests your design skills and your ability to work with complex and scalable services. Your performance in these interviews determines what position and salary you will be offered. If you have commendable design skills, you will get a higher offer. A good resource to prepare for this interview: Grokking the System Design Interview .
3. **Cultural fit interview**. During this interview, companies see if a candidate would be a good fit for their culture. Candidate should not show any red flag.

when you are interviewing at the top companies like Google or Facebook. In these companies, if a candidate doesn't perform above average, they have a limited chance to get an offer. Also, a good performance in such interviews always results in a better offer (financially), since your performance reflects upon your ability to work with complex systems.

Also, all these big companies like candidates who are familiar with distributed systems concepts like Consistent Hashing, Data Partitioning , Load Balancing , Redundancy and Replication , CAP Theorem , etc.

A few sample design problems are:

- Designing a URL Shortening service like TinyURL
- Designing Instagram
- Designing Twitter
- Designing Youtube
- Designing Facebook Messenger
- Designing Dropbox
- Designing Typeahead Suggestion
- Designing Facebook's Newsfeed
- Designing Yelp
- Designing Uber

I can put more details about coding interviews, but I felt that other answers have some good references.

Hope this help!

Jake Cook, ex-googler, years experiences in interview, co-founder of gainlo.co

Answered Sep 23, 2015

Software engineer interview at Google is quite standard. You will expect to have general technical interviews that mainly cover data structure, algorithm, system design and testing. And this process is quite similar to other big companies like Facebook, Amazon etc.. Usually the on-site interview has 4-5 rounds and you will be expected to write a lot of codes on whiteboard. So it's quite intense. You can check How we hire - Google Careers for more info from its official page.

How to prepare for it? Certain ways will definitely make the preparation easier as the interview is very standard and general and the key is really spending enough time on practicing.

1. Internship is a shortcut.

A lot of people are talking about how to prepare for the interview, but before that I'd like to mention if you were able to get an internship at Google, you're gonna get the full time offer much easier.

Google provide return offers for last year interns and the ratio is quite high. So

well. Although internship may not be an option for a lot of people, it's good to know that this can be a "shortcut".

2. Prepare well for data structure and algorithms. You'd better spend a lot of time getting familiar with these basic knowledge you learned at school, as they are the basic of your interview. I would describe these as your tools to solve interview questions. In an interview, you may encounter different data structures, you may be asked to analyze time and space complexity, and all of them are covered in this topic. Books like Introduction to Algorithms are great options and you can also check technical interview cheat sheet and Big-O Cheat Sheet . Here are things you should spend most of your time on:

- Concepts of each data structures. You shouldn't be confused about stack and queue.
- Pros and cons of each data structure and when to use each of them. For example, you should be clear when to use tree instead of linked list.
- Understand basic algorithms like BFS, DFS, sorting algorithms. It's better you can code them without hesitation. It's said that only 10% of programmers can code binary search without bug.
- Pros and cons of each algorithm and when to use each of them.
- Be extremely proficient in time/space complexity analysis. It's almost for sure they will be covered in an interview.

3. Be familiar with coding questions. The idea is to be familiar with how to use what you learned from those books to solve a real question and know about what kind of questions are asked in a general interview. Just delve into those questions and practice as much as you can. Resources like leetcode.com , glassdoor.com and Cracking the Coding Interview are very popular and you can find tons of interview questions online to practice. Here are several thing to keep in mind:

- It's always good to practice past interview questions from the company. In your case, you can get a lot of Google interview questions online. It's unlikely to be asked the same question in real interview, but you can get an idea about the style and focus of that company.
- Try to write down your solution on white board or at least on paper. Many candidate could jump to the write solution quickly, but failed to write down the code. It's so different between describe the solution and code the solution perfectly.
- Try to talk while thinking. This is quite important in a real interview since you need to keep communicating with the interviewer. You may feel hard to focus initially and this requires some practice.
- Use a timer to track how much time you spend on each questions. It's quite common to have 2 questions at 45min interview at Google, so you won't be allowed to solve a simple question with one hour.

4. Have mock interview. Technical interview doesn't only evaluate your coding ability, but a variety of skills and abilities like communication skills, analysis ability etc.. It's very important to communicate with your interviewer while solving the problem and your reaction to interview's hint is also evaluated. So it's a two-way process rather than you are taking an exam by yourself.

over his shoulder. That's why people may fail with problems that can be solved easily at home. The key point is to practice with a real person instead of yourself.

A lot of people also want to get good quality feedbacks from experienced interviewers. With that in mind, we worked on building http://www.gainlo.co/ , which allows candidate **have mock interview with experienced interviewers** from top companies like Google, Amazon etc. and will get real feedback to help them improve.

Try to make a detailed preparation timeline and stick to it. It's also important to allocate enough time every day for your preparation. Many people choose to spend less than an hour a day, which will never work. Just practice as much as you can and the interview won't be hard thing for you eventually.

252 Views

Upvote Downvote

Add a comment... Recommended All

Brian Samson, worked at Google
Answered Feb 28, 2012

I worked on the Google engineering recruiting team in 2011. It's a much faster process than in the past and they are working hard to improve the experience. Expect everything to take roughly 2-3 months, but you'll only have 3 interviews in that process.

#1 Phone interview with a sourcer or recruiter
#2 Phone interview with an engineer. Expect a live coding exercise in a Google doc. Know your Java, Python, and C++
#3 The onsite interview. This is an intense 4+ hour event, where you'll meet with several engineers and be grilled on your coding and algorithm abilities. Being able to think on your feet is critical. The wild questions of how many marbles can you fit into a school bus have mostly stopped. Very few get selected to come onsite, so consider this an honor in itself. An even smaller number do well enough on the onsite to get to the offer stage, so don't feel bad if it doesn't work out.

If you do well on the interview, Google will make it happen to bring you on-board, the salaries and bonuses are strong, although they aren't hung up on titles. Your education certainly helps you navigate the interview process, so if you're a MIT, Cal, Stanford, or CMU grad, you're a step ahead of the game!

11.9k Views · View Upvoters

Upvote 10 Downvote

Add a comment... Recommended All

Jonathan Sobieski, Software Engineer at Google
Answered Mar 6, 2016

I interviewed at Google in November and December of 2015 (about four months ago), got an offer, and will be starting working at Google later this month. For me this was after spending three years working as a software engineer after finishing my CS Professional Master's degree. So I was interviewing for a level 4 position. New college hires are usually brought in at level 3 while new PhD grads

interviews since it's not a particularly senior level.

Initially I was contacted by recruiters periodically (4 or 5 times over a 2-3 year period - I assumed they saw my profile on LinkedIn) until I decided to interview. The recruiters explained very clearly what to expect in the process, and they provided resources to prepare for the interview including a PDF copy of Cracking the Coding Interview, a PDF document explaining what CS topics to review before the interview as well as links to many of the excellent articles already listed in other answers to this question in addition to the general guidance just to try to find example Google interview questions online and try them out. The recruiters were very helpful, and it was clear that they wanted to make sure that I was as prepared as possible for the interview. They explained that it is in both Google and the candidate's best interest for candidates to be as prepared as possible especially since Google's interview process is deliberately designed with an emphasis on preventing false positives even at the cost of passing on people who otherwise would be good. They even asked if I would like to put off the interviews to spend some time preparing, so I decided to spend a few extra weeks preparing. One thing they didn't mention until the topic of compensation came up later in the process was that your performance in the interviews can impact your initial offer which is another compelling reason to prepare. I'm not sure it's that much of an impact and of course your performance on the job will be much more important in the long run in deciding your bonus, equity refreshes, and promotions.

I had one phone screen then a day of on site interviews in Mountain View with five technical interviews (3 algorithmic coding interviews and 2 design discussion interviews) and a lunch interview which they made clear was just to allow you to ask questions about Google and did not factor into the hiring decision. Google was very nice about flying me to Mountain View the day before the interviews and covering the hotel the night of the interviews allowing me to fly home the next day.

All interviews were 45 minutes. All interviews required me to write code, syntactically correct code in the algorithmic interviews and pseudo code in the design discussion interviews. I coded in Java. All coding questions were followed up with two questions: how would you test your code, and what's its runtime? All interviews ended with a few minutes for me the ask the interviewer questions.

The initial phone interview required me to write code in a Google doc (strongly recommend you practice this before the interview, it's not too bad) and consisted of 3 coding questions that were clearly asked in increasing order of difficulty. I assume this was because many people fail the phone screen and hence they do not want to give them a hard initial question, and also answering easy questions initially helps build up your confidence and ease your nerves. The first coding question actually came from Cracking the Coding Interview, and I had a solution to it written up in about 5 minutes. The 2nd question I spent about 10-15 minutes on and was also fairly straight forward. The 3rd question was clearly the hardest, and I produced an initial solution that was correct but not maximally effici...(more)

Originally Answered: How should one prepare for an upcoming google internship phone interview for a software engineering position?

Steve Yegge's blog has a number of posts that give a good overview of the interview process, including topics to study.
http://sites.google.com/site/ste...
http://steve-yegge.blogspot.com/...
http://steve-yegge.blogspot.com/...

Robin Thomas, Interviewed with Google
Answered Feb 4, 2017

Coding

Pseudo code? Be expected to write syntactically correct and well formatted code. You shall be coding in Google Docs. So get well versed in it. No more whiteboard!

You can pick your programming language of choice. But I hear it's recommended to choose one from C++, Java or Python. And make sure you know that one well, for you are expected to know it well.

Data Structures

Make sure you are well versed in *linked lists*, *trees*, and *hash tables* at the very least. Know all their basic operations and make sure you can code them well. Most interviewers would stay away from graphs, but knowing it can come handy. Try learning *tries* and one *self-balancing BST*.

Algorithms

Know atleast one *O(n log n)* sorting algorithm. Forget Bubble Sort. Learn *BFS*, *DFS*, *Topological sorting*, and *Dijkstra's algorithm* (maybe A* too). And know your Big O's well. Learn about NP-Complete class. Check out the Travelling Salesman, knapsack, SAT and halting problems too.

Try out books like *Cracking The Coding Interview*, *Programming Interviews Exposed*. Read through Glassdoor . Try competitive programming.

Operating Systems

The probability of you being asked to code a multi-threaded program is slim. But nonetheless, be expected to get drilled on OS theory (think *process*, *thread*, *semaphore*, *mutex*, *deadlock*, *concurrency*, *scheduling*, *system calls* and so on) and Linux commands. I have found Robert Love's books to be a great primer.

Databases

This is a bit of grey area. Unless your role demands it, you might get away with no database questions. But mind the *Design Round*. And since Scalability is such a big word at Google, better get yourself to speed on NoSQL, sharding, scaling, and so on.

And one last thing. Read Get that job at Google from Steve Yegge, which even Facebook was so keen on recommending!

Good luck!


Add a comment... Recommended All


Miguel Andres, Appeared and cleared Google interview.
Answered Mar 21, 2012

I recently interviewed for a Software Engineer position at Google (I am assuming that you are also applying for an engineering position). Below some of the things I learned from the process.

(Note: I did not have any interviewing experience for software engineer positions -- have done mostly research for the last few years -- so I spent quite some time on finding out what to expect from this kind of interviewing process)

1. The first step is to **get really familiar with the interviewing process**. Few things you can do for this purpose:

- Check out websites like glassdoor.com and careercup.com -- both offer plenty of information about what to expect from the interviews. I even took a mock interview with careercup (which helped a lot). Also bought several interview videos from careercup (this is also highly recommended -- this way you can learn not only what to expect from the interviews but also what the interviewer will expect from you).
- You should also ask your recruiter as much information as possible about the interviews (for instance, you can find out in advance which teams will be interviewing you and use this info to "tune" your preparation).
- Ask friends if they know somebody that has gone through this kind of interviewing process. Talking to people that have already has this experience can be very fruitful (it was for me).
- Something else that would help A LOT (but probably is not an option for you anymore -- as you will be interviewing very soon) is to have interviews with other companies. In this way, you will get a good feeling of what to expect from the Google interviews (most of the big tech companies - like Google, Facebook, Microsoft, Amazon, etc - have a very similar interviewing process). I am convinced that having (at least some) experience with this kind of interviewing system is very helpful to improve your interviewing skills (I did not do this myself but should have done it).

2. **Prepare for the technical questions**. Probably/hopefully you have already invested a good amount of time on this. Anyway, these are my advices:

- Again, check out places like glassdoor and careercup (there are several other sites like this). This will give you a pretty good idea of the type of questions that you can expect.
- You can also find several interviewing questions in books like "Programming Interviews Exposed..." and "Cracking the Coding Interview". I those books you will find detailed information about how to solve those problems in addition to the solutions.

Cup, and Top Coder. This is much more fun than just solving (simpe/ish) interviewing problems. Some of the lessons that I took from solving those problems turned out to be very useful during my online interviews.

(Anyway, whatever the source of problems you want to use, the real key is to really**solve the problems yourself**, i.e., **do NOT just read through the answers**).

- Practice interviewing with your friends! I know it is boring and annoying for your friends, but it is really important (it will help you a lot to have a good feeling of how it is interviewing for real).
- Something else (VERY IMPORTANT): Do practice in a blackboard (preferably) or paper - bottom line: *NOT on a computer*. The kind of experience you will find interviewing at Google will be completely different than writing code in your computer.

3. I know that this is a difficult one, but it is an important one: **be relaxed** during the interviews.

- Something that will help with this (but is certainly not enough) is to be as prepared as possible for the interview.
- Having an offer from another company might also help a lot.This might take away some of the pressure (actually this may also help you to get an offer - being wanted by other tech companies says something good about you - and also in the negotiation stage).
- Something else that you should keep in mind is that you WILL BE asked tough and completely unfamiliar/unexpected questions. Being aware of this should help you not to freak out WHEN this happens.
- If one interview does not go well try to get over it as soon as possible (this is very important to avoid ruining subsequent interviews). This happens very often (and it is part of the game), just remember that screwing up one (or maybe even two) interview(s) does not mean that you are not gonna get an offer (in spite of what some people say in forums online). I have heard plenty of stories of people having 1 or 2 bad interviews and still getting a job offer. The key is to have some of the interviewers to really like you. I believe that it is better to have 1 or 2 very good feedback reports (from the interviewers) + some negative feedback rather than all "just ok" (not really enthusiastic) feedback reports.

4. A final advice: **do NOT interview if you are not properly prepared**. If you really want to work for Google as much as most people interviewing with them do, then spending few extra weeks (or even 1-2 extra months) preparing for the interviews makes a lot of sense (this extra time might be the difference between landing your dream-job and having to go with a plan B).

Thats it! hope you find some of this information useful. Good luck with your interviews.

102.6k Views · View Upvoters

Tony Zeng
I have similar background as yours. I have been in research area for a few years. Do yo...

Sachin Gupta, Created Learn.HackerEarth.com a portal to discuss Job Interview Questions
Answered Jul 15, 2012

First thing while preparing for Google Interview is that don't get scared of the hype surrounding the Google interviews. If you prepare well then you can easily crack it. Make sure you have your concepts clear. Some of the topics you must absolutely brush through are

1. Dynamic Programming - 4 out of 8 questions are on DP
2. Trees - Trees are a must in any technical interview.
3. Data Structures - Specifically Google is fond of asking questions based on relatively less heard data structures like Interval Tree or Balanced Binary Trees
4. Puzzles - Almost any Google interview will have puzzles, mostly related to some programming concept.

Check out http://www.mycareerstack.com/que...
It has got lots of resources.

10.8k Views · View Upvoters

Upvote 28 Downvote

Add a comment... Recommended All

Taki Chowdhury, studied at Mahindra United World College of India
Answered Dec 15, 2015

Hi,

I recently interviewed there and I can definitely share my experience with you:

I started preparing by solving each problem in the standard textbooks for interviews:

1. cracking the coding interview 6e

2. techincal interviews exposed

My preparations started 1 month before my first interview of the season (with Microsoft). I finished solving all the problems in the books within the first 3 weeks at ~15 problems per day over 3 hours per day. Over the final week, i used the following service to do some mock interviews: Mock Interview With Real IT Professionals It was recommended to me by a friend and I chose it over others like Gainlo mainly because of its competitive prices (~$50 per interview). I can definitely say that doing mock interviews helped me a lot as I tend to be somebody who has trouble clearly expressing my ideas verbally. I would suggest doing 3-5 (I did 5: first 3 were bad, next 2 were amazing) to gain confidence before your interviews. It is important to remember that doing poorly in these interviews is kind of the point (they are mock interviews after all!); what you should be focused on is improving over a series of interviews. In my case, I realized I was on an upward trajectory when I made fewer mistakes (personal + technical) on the second interview as compared to the first and so on.

8.7k Views · View Upvoters

Upvote 31 Downvote

Add a comment... Recommended All

Randy Knaflic, Director of Recruiting and Staffing Programs
Answered Jan 10, 2012

Moishe pretty much sums it up.

Two additional points that could be helpful. Candidates may want to get a sense of how Google likes to set code (Java, C++, etc). The easiest way to do so is to read our style guide! http://code.google.com/p/google-...

Google also loves opensource - many of our Engineers actively participate and are on projects. Why not join an opensource project with a Google engineer on it and get a sense of how they like to write code.

8.6k Views · View Upvoters

Upvote 8 Downvote

Add a comment... Recommended All

Nemanja Cerovac, Making Interviews Great at Refdash (2017-present)
Answered Feb 7

What you can expect in a Google interview may vary based on your experience and the exact position you are applying for. Still, these three things will certainly be tested:

- **Relevant tech knowledge** (coding practices, language specifics...)
- **Your ability to solve problems** (using algorithms and data structures)
- **Your ability to design systems** (especially if you're more senior)

Most technical questions that you will be asked will be about **algorithms and data structures or system design**.

Data structures & Algorithms

When it comes the algorithms, questions usually cover areas such as searching and sorting, geometric algorithms, dynamic programming, divide and conquer, and analysis of algorithms. They also cover most of the common data structures.

First, brush up your theoretical knowledge. Picking up a book or a course may be a good idea here.

CLRS book is a classic and it covers most of algorithm topics that you need for a Google interview.

There are also online courses. These are especially great for self taught engineers who never had formal education in algorithms and data structures.

On websites such as MIT OCW and edx.org you can find many great courses available for free. Here are some courses to get you started:

- Introduction to Computer Science and Programming
- Structure and Interpretation of Computer Programs
- Introduction to Algorithms

After you go through all the basics of algorithms, start doing a lot of practice problems on sites such as TopCoder and CodeChef .

System design

Other questions will be related to system design.

This is what sets experienced engineers and not-so-experienced apart. Key here is to understand concepts and main ideas of system design and then you'll be able to tackle almost any problem interviewer could give you.

Keep in mind that these questions rarely have one correct or optimal answer. They are meant to give insight into your way of thinking about design and scaling. Knowledge of distributed systems can prove to be useful here.

Many engineers claim that Grokking The System Design Interview is the best resource for preparing for this part of the interview.

System design primer is also worth checking out.

Practice interviews

Now **here's where a lot of people go wrong:** they fail to acknowledge how you will be tested.

You may have a round or two of phone screens — practice talking about tech topics and your experience.

You will be asked whiteboard questions — practice coding on paper or a whiteboard if you have one.

You will be expected to share your thought process — practice speaking while solving practice problems.

Once you've got the basics down, **practice interviewing**. Find someone who can act as your interviewer and try to simulate an actual interview.

Let your partner come up with some questions and try to have realistic time constraints. Use the same or similar media as you will use in the actual interview (whiteboard, paper).

You can check Glassdoor for actual interview experiences people had with Google.

After each practice interview, have your partner give you feedback on how you did. This will be easy if he or she is an experienced engineer, but you often won't have access to those while you practice.

This is why we've built Refdash — a platform where you can do free interviews with senior engineers from companies like Google, Palantir, and Facebook, just to name a few. They will give you detailed verbal and written feedback that you can use to work on your skills and increase your chances of getting hired. If you do well, they can even help you skip some rounds of interviews at some big tech companies.

Good luck in your job hunt!

Also, check out Answer to Programming Interviews: What are the top sites for software engineer interview preparation?

3.3k Views · View Upvoters · Answer requested by Nitin Asokan


Add a comment... Recommended All


Sharad Dutta
Answered Jun 13, 2017

The syllabus for the interviews is very clear and simple:

1) Dynamic Programming

2) Super recursion (permutation, combination,...2^n, m^n, n!...etc. type of program. (NP hard, NP programs)

3) Probability related programs

4) Graphs: BFS/DFS are usually enough

5) All basic data structures from Arrays/Lists to circular queues, BSTs, Hash tables, B-Trees, and Red-Black trees, and all basic algorithms like sorting, binary search, median,...

6) Problem solving ability at a level similar to TopCoder Division 1, 250 points. If you can consistently solve these, then you are almost sure to get in with 2-weeks brush up.

7) Review all old interview questions in Glassdoor to get a feel. If you can solve 95% of them at home (including coding them up quickly and testing them out in a debugger + editor setup), you are in good shape.

8) Practice coding--write often and write a lot. If you can think of a solution, you should be able to code it easily...without much thought.

9) Very good to have for design interview: distributed systems knowledge and practical experience.

10) Good understanding of basic discrete math, computer architecture, basic math.

11) Coursera courses and assignments give a lot of what you need to know.

12) Note that all the above except the first 2 are useful in "real life" programming too!

Interview 1:

Graph related question and super recursion

Interview 2:

Design discussion involving a distributed system with writes/reads going on at different sites in parallel.

Interview 3:

Array and Tree related questions

Interview 4:

Designing a simple class to do something. Not hard, but not easy either. You need to know basic data structures very well to consider different designs and trade-offs.

Interview 5:

Computer architecture and low level perf. enhancement question which requires knowledge of Trees, binary search, etc.

Conclusion***: "It's not the best who win the race; it's the best prepared who win it."***

This is to the best of my knowledge. :D

```
System.out.println("All the best");
Print "Have a badass day"
```

582 Views · View Upvoters

Upvote 2 Downvote

Add a comment... Recommended All

Maryna Cherniavska, Software Developer, OLX Berlin
Updated Dec 20, 2015

Originally Answered: Should I prepare for Google's software engineer interview?

I have about 10+ years of experience, and I failed a Google Software Engineer interview very recently. Not even an on-site interview; I failed a phone interview. So, basically, it's a fail of rather epic proportions.

When you've been on the job for some years, you sometimes tend to fall into a routine. And let's face it; our routine tasks aren't usually close to solving algorithmic problems - well, for most of us at least. There's usually lots and lots of business logic to deal with, some architectural decisions, some design patterns, and lots of routine - bug fixing, unit tests etc. You don't do a lot of algorithms that way. But gaining experience, you sometimes are inclined to mistake it for what it's not. And it's not, well, the kind of experience that helps in an interview. You learn to deal with clients, with frameworks, with people inside the team. You sometimes learn new programming languages too. But this is all very far from the kind of academic learning that Google requires. I'd say that freshly graduated college/uni students will probably beat you nine times out of ten. Their experience is just so much closer to what an interview requires.

So, you don't just need to prepare. You need to study a lot and refresh a lot. You need to get back to the basics you have long forgotten. You need to beat that college student hands down.

Having said all that, I will add that I studied and prepared for 3 or 4 weeks. And it wasn't enough. But that's not what made me fail. When I was on a call I just froze and couldn't think clearly. So I hope that you also have the nerve to not be intimidated by the Great Googler Calling!

3.7k Views · View Upvoters

Upvote 9 Downvote

Add a comment... Recommended All

Amit Agarwal, Software Developer at Amazon
Answered Jan 12

The standard interview process for product development companies like Amazon,Facebook ,Google ,Microsoft is same. You can fairly prepare in 2–3 months . I got a offer preparing using the right strategy.

Complete Process

telephonic or you will be asked to complete a coding challenge .These round consists of basic to medium level data structures and algorithms questions.

Onsite Interview : After you passed the screening round you are asked to fly down for the onsite interview at the company office.The onsite round usually consists of 3-4 more rounds of interview. This will consist of at least 2 technical rounds, 1 system design round and 1 behavioral round which is usually taken by the hiring manger of the team .

How to Prepare??

Data Structures and Algorithms : This is the heart of every software development company interview . This comprises of the 50-70% of the whole interview questions. Ability to convey your algorithmic knowledge along with the neat code is must.

System Design Interview : This is very important round of interview . This will be make or break for your position and will decide the compensation u will be getting in the company. There is usually one round of it.

Behavioral Round : This is usually for general behavioral questions mostly taken by the hiring manager . This is equally important to pass

What to Prepare??

Data Structures And Algorithms:

Array, Queue, Linked List, Tress , BST : Try to understand the basic data structures and the running time complexity and space complexity of inserting , deletion and update operation. Try to cover all standard questions .

Greedy , Dynamic Programming ,Divide and Conquer ,Graphs Algorithms are very important.

Source : GeeksforGeeks | A computer science portal for geeks, leetcode ,Cracking the Coding Interview book ,Hackerrank,Hackerearth,Interviewbit(for practice in real time).

System Design Interview

Usually the question asked will be design a messenger, uber ,bookmyshow or social media app.

The candidate should be able to first point out the functionality along with the technical design.

Imporatnt points : DB design , Scalability, Cache design, server choice, sql vs no sql db, distributed infrastructure and cdn.

Source : http://Scalability.com, Success in Tech (youtube channel), Tushor Roy(youtube channel) , https://github.com/checkcheckzz/... (github link)

Complete video link:

How to crack interview of Amazon,Facebook,Google,Microsoft

Questions generally belong to one of these three categories:

1. **Passion Questions:**

These questions are designed to demonstrate your passion for technology.

Example:

“I’m really interested in learning about scalability, and I know [Company] has a lot of interesting scalability challenges. What sorts of opportunities are there to learn about scalability?”

2. **Insightful Questions:**

These questions are intended to show how you think about technical problems.

Example: “I noticed that [Company] based its software off an open source ad server, but I’ve heard that it had a lot of issues with multiple connections. How do you handle that?”

3. **Genuine Questions:**

These questions are about what you really want to know. That could be anything, of course, but here’s an example:

Example: “How much of your day do you actually spend coding? How many meetings do you have on average per week?”

Technical Skills Required

For each data structure, you should know how to implement the data structure, when to use it (pros and cons) and how to deal with space and time complexity.

Some of the most important data structures are:

[math]Linked lists [/math]

[math]Stacks [/math]

[math]Queues [/math]

[math]Trees [/math]

[math]Trees [/math]

[math]Graphs [/math]

[math]Vectors [/math]

[math]Heaps[/math]

Algorithms that you must know

For each algorithm, you should know how to implement the algorithm and you should know its space and time complexity.

Some of the most important algorithms are:

[math]Breadth first search[/math]

[math]Depth first search[/math]

[math]Tree Insert / Find[/math]

[math]Merge sort[/math]

[math]Quick sort[/math]

[math]Binary search[/math]

```
1  SOURCE : http://docplayer.net/17959013-Google-software-engineer-interview-guide
```