My Guide To Preparing for the Google Technical Interview

Anthony D. Mays | Follow
Software Engineer at Google

*TODO(you): Upvote **my original post** and other good (better?) answers on **Quora**. The following is my answer to the question "How should I prepare for my Google interview if I have 1 month left?"*

With over ten years of programming experience and a CS degree, it took me about a month and a half of daily practice to get ready for the interview. “Ready,” for me, is *ambitiously* defined as the ability to tackle almost any technical interview question in 30

Ingredients:

About two years of solid coding experience

Pencil and paper

Cracking the Coding Interview (*CTCI*)

Your favorite algorithms book—**Introduction to Algorithms**, **The Algorithm Design Manual**, and **Programming Pearls** are a few good choices. Might I suggest you go low tech and buy physical copies?

Two or three hours a day

Highly recommended: Whiteboard

Highly recommended: CS degree

Optional: **MIT OpenCourseware** or another learning site

Directions:

1) Learn as much as you can about the Google interview process (days 1–2)

Just like an incomplete understanding of a technical question will ruin you in the actual interview, misinformation will derail your preparation leading up to it. **I learned this the hard way** when I failed at my first attempt after emphasizing brain teasers over studying algorithms and data structures.

Start with the source—check out **google.com/careers** for info on how Google hires. Then watch **this video** from Google about what interviewers look for in the interview, and finally check out **an example interview** featuring real Google engineers.

Once you have a solid foundation, I'd recommend following up by reading *CTCI*. Particularly focus on chapters 5 and 6 entitled *Behavioral Questions* and *Technical Questions*.

By the time you finally read my short article about the **six things you absolutely need to do** during the actual interview, you should have a good grasp of what a well prepared candidate looks like.

2) Benchmark yourself (days 3–5)

Now that you know how prepared you need to be, figure out where you are right now.

Use *CTCI* for this. Take a couple of questions from each section and solve them using the **six steps I mentioned earlier**. Keep track of how long it takes you to reach an optimal solution for each problem you solve.

If and only if you've solved the problem yourself, take a look at the accompanying solution to assess how you did. Did you reach the optimal solution or at least progress beyond the naive/brute force answer? How long did it take you? Was your code written in the fewest lines possible?

Do this for every section. When done, you can prioritize the sections that you didn't do so well on up front in your practice regimen and leave the other sections for later. You should repeat this exercise just before your interview so that you know your weak spots going into the day of the interview.

to study. This list should include:
Memorizing two good sorting algorithms and **their Big-O**
Memorizing binary search
Memorizing how to implement basic data structures such as hashmap, linked list, stack, queue, and trees (n-ary, trie, heap) and **their Big-O complexities**
Memorizing graph traversal algorithms (BFS, DFS, and a shortest path algorithm like Dijkstra's)
Memorizing powers of 2
Practice bit manipulation exercises (working with bit maps, bit shifting)
Object-Oriented Programming terminology (abstraction, inheritance, cohesion, coupling)
Know the collections and math APIs for your given programming language
Recursion, backtracking, and memoization
Review principles of basic discrete mathematics and statistics
This is all covered in *CTCI* and your favorite algorithms book. **Note:** *the point of the memorization is understanding! You will probably never be asked to write out an algorithm verbatim. Rather, you'll be expected to know each well enough that you can use them creatively to solve a problem you've never seen.*

4) Practice algorithms and data structures daily (days 6–30)

Pick a two or three items from your list and commit about two or three hours each day working on these things (e.g. 1 hr before work, lunch break, 1 hr after work). As you memorize things, test yourself by writing out an algorithm or data structure on paper or on a whiteboard. Write down the worst case Big-O time and space complexities for the algorithm when you're done.
Always check your work, always!
Now copy what you've written to your favorite IDE and compile. Take note of any compilation errors so that you can avoid them when you repeat the exercise again. You can and should also create unit tests to verify the correctness of your code.
Keep doing this until you can transcribe and compile your code without logical or syntactical errors.

5) Tackle as many programming questions as you can (days 16–30)

By now, you should have a pretty good handle on the skills you need to succeed on an interview question. Starting with CTCI, tackle every single programming problem you can, again using **the six steps**. Devote about half your study time to this while you spend the other half reviewing items from your study list.
If you're doing well, you'll probably start to run out of questions in the book. You can find tons more of real samples online from sites like **CareerCup** or **Interview Cake**. Or, just use your favorite search engine. I know a **pretty good one** you can use ;).
Practice a few times with another person, both with someone technical *and* someone non-technical. Ask them if:
You looked and sounded relaxed

6) Relax and get some good sleep (day 30)

Congratulations! You've worked really hard. There's nothing else you can do. Relax and get into your good place. You've made it this far. That means you either really enjoy coding or that you will stop at nothing to get that job at Google. I think you will genuinely enjoy your interview experience. Make sure to have fun! I look forward to seeing you in the meeting room.

Anthony D. Mays
Software Engineer at Google

---

16 comments

Daniel Richardson
Student - Software Development, B.S. @ Western Governors University
4mo

Valuable experience's shared. Thanks for the post!

Like Reply | 1 Like

Narendra Koli
Associate Software Developer at SAP
8mo

Excellent.... Thank you..

Like Reply

Narendra Koli
Associate Software Developer at SAP
8mo

Excellent.... Thank you..

Like Reply

Annette Hudson Nguyen
Digital Pioneer | Google Product Marketing
1y

Alex Dinh Kevin Tran John Huynh Cass Tao

Like Reply | 2 Likes

Great post Anthony, all is possible with God. Thanks for sharing.

Like Reply | 2 Likes

Samhita S.

Samhita S.
Software Test Engineer at Microsoft -Roberthalf

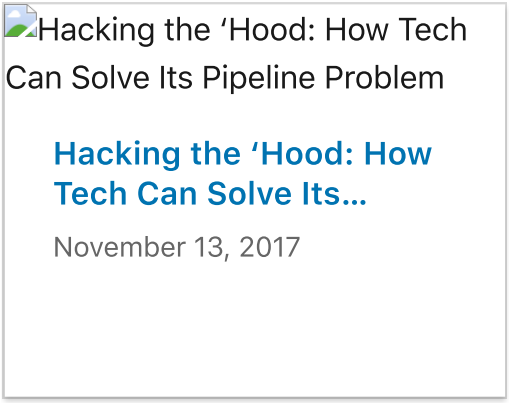
1y

Very good article!! Thanks for sharing!!

Like Reply | 1 Like

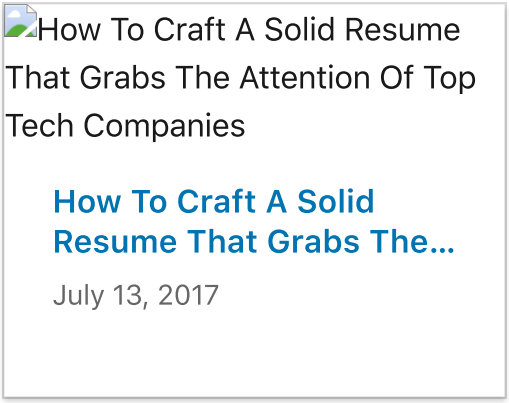
Show more comments.

More from Anthony D. Mays 6 articles



Hacking the 'Hood: How Tech Can Solve Its...

November 13, 2017



How To Craft A Solid Resume That Grabs The...

July 13, 2017

Google Help The Father I

May 6, 2017