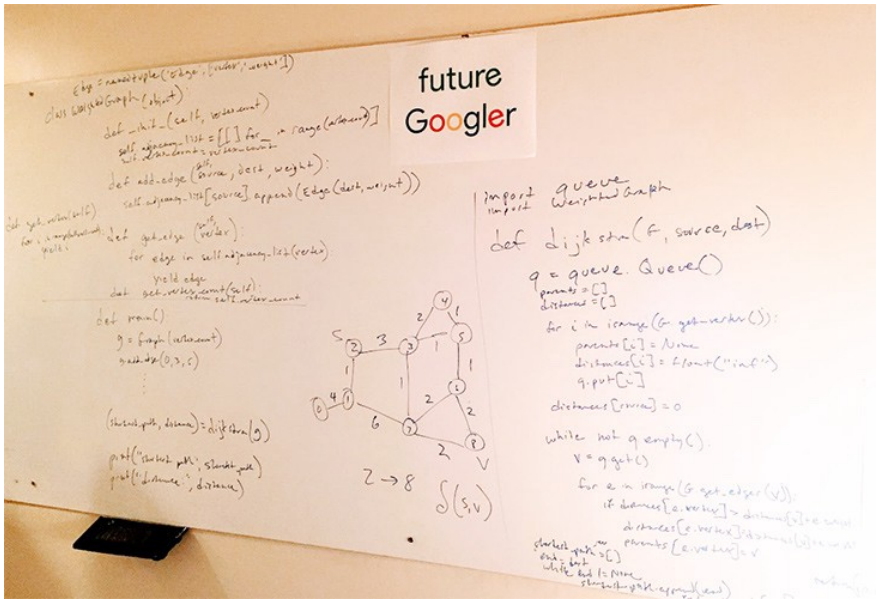
Googley as Heck Follow
Hi, I'm John Washam. Future Googler. Check out my journey on my blog: https://googleyasheck.com
Dec 13 · 10 min read

Why I studied full-time for 8 months for a Google interview



My well-worn whiteboard, adorned with Dijkstra's single-source shortest-paths algorithm.

It's true. I've spent thousands of hours reading books, writing code, and watching computer science lectures, all to prepare for the Google software engineer interview.

If you'd like to prepare for a Google interview, here's my study plan:

jwasham/google-interview-university

google-interview-university - A complete daily plan for studying to become a Google software engineer.

github.com

How I Got Here

I started programming in middle school, but when it came time for college I pursued a degree in Economics. My rationale was that there would be too many programmers looking for jobs by the time I graduated. Boy, I was wrong.

Later, I joined the Army to become a programmer, but the recruiter talked me into a military intelligence position, and I spent the next two years studying the Korean language. I served in South Korea for 2 years afterward.

Before I left the Army, I attempted to get back into programming and was surprised at the difficulty. I had learned BASIC in middle school and kept programming it through high school. But I restarted my programming studies with C++, and the leap was too large. I just couldn't grasp it.

I did enjoy making websites, however, but I used software with a Word-like interface that I used to publish my websites. I didn't know how to make websites from scratch.

After the Army, I decided to stay in Korea for a year and teach English. I used my nights and weekends to study web programming, using Perl, HTML, CSS (which was new at the time), JavaScript, and SQL.

After a year of intense study, I landed a job in the Seattle area, and I've been here ever since.

Doing some work from the balcony in beautiful Bellevue, WA.

I've been a web developer now for 15 years. I've started 3 companies, 2 of which are still running and generating revenue. I've worked at large and small companies, helped startups launch and grow, and recruited and managed teams. I've been a product manager, a CEO, a designer, and a marketer.

I've had a successful career and learned a lot along the way. But I'm not done yet.

Seeking a Career Change

Remember the part where I didn't get a computer science degree? It has made a difference.

A few years ago, I thought I could get hired anywhere. I thought I was hot stuff: the elusive full-stack web developer. But during my job search in 2013, I realized my skills were lacking. I had spent so much time chasing dollars by running startups in my spare time, that I had let my skills atrophy. I hadn't kept up with technology.

For years, I had learned just enough to get by. I had a wide skill set but wasn't an expert in anything.

Don't get me wrong, I could still get hired, but not in the technologies or areas I wanted to work in. I could get hired for areas where the tech stack was somewhat outdated, like me. There's big money in there, but I didn't see exciting prospects.

The realization reached its peak last year at a career fair. I was interested in perhaps working for one of the local companies that were startup labs run by venture capital firms. However, the fact that I lacked a computer science degree, and the skills and knowledge that accompany such a degree, meant I didn't have a chance.

I was working full-time on my businesses at the time, and still am today.

At the beginning of 2016, I decided it was time to make a career change from web developer to software engineer. I would need to study hard and practice in order to compress a computer science degree into a few months, but once I did, I could start a new career.

You may not see web development and software engineering as different positions. Both involve programming and craftsmanship, but software engineering adds to it knowledge of data structures and algorithms, compiled languages, memory considerations, and understanding the impact of coding and architecture decisions on the machines where they reside.

Large companies that hire for software engineering positions expect candidates to have this knowledge.

I reached out to an acquaintance who works at Google and asked him questions about his experience at the company. I had been reading *How Google Works* and was pretty familiar with Google already.

Through another contact, I received a copy of Google's coaching notes that are provided to interview candidates. This became the basis of my study plan.

Google is a pretty awesome place to work, but before I even knew that, Google was my goal.

Why Google?

Google sets a very high bar for hiring. They want to hire only the best. So if I set my sights high (getting hired at Google), I'll still be quite hireable elsewhere even if I'm not selected.

The more I learn about Google, the more I want to work there.

In brief, Google is a company that hires smart, creative people, and treats them well. Google rewards merit, encourages big ideas, and gives employees the freedom to make good decisions for the user.

The hiring process is calibrated to bring in smart, passionate people. Google has honed the recruitment and interview process over the years. The brain teaser questions are long gone. Nowadays candidates are chosen based on coding ability, technical knowledge, and Googleyness. There's a lot going on in that one word.

On a road trip in 2015, I visited the Mountain View HQ. That planted the seed.

Management is different. Managers don't micro-manage. They trust engineers to make the right decisions. Trusting employees changes the role of managers at Google from what most folks envision when they think of management. In addition, managers can't unilaterally, hire, fire, or promote. Many of the important management decisions that could be perceived as office politics are handled by a committee to remove that danger.

Google's people operations (HR) has learned what works over time, and they use data and employee feedback to improve evaluation systems, the hiring process, promotions, compensation, benefits, and more. Read *Work Rules!* by Laszlo Bock (SVP, People Operations) for more.

Yes, the benefits are amazing. I went on a tour of the Google office in Kirkland, WA, and it surpassed my expectations. And my expectations were already high.

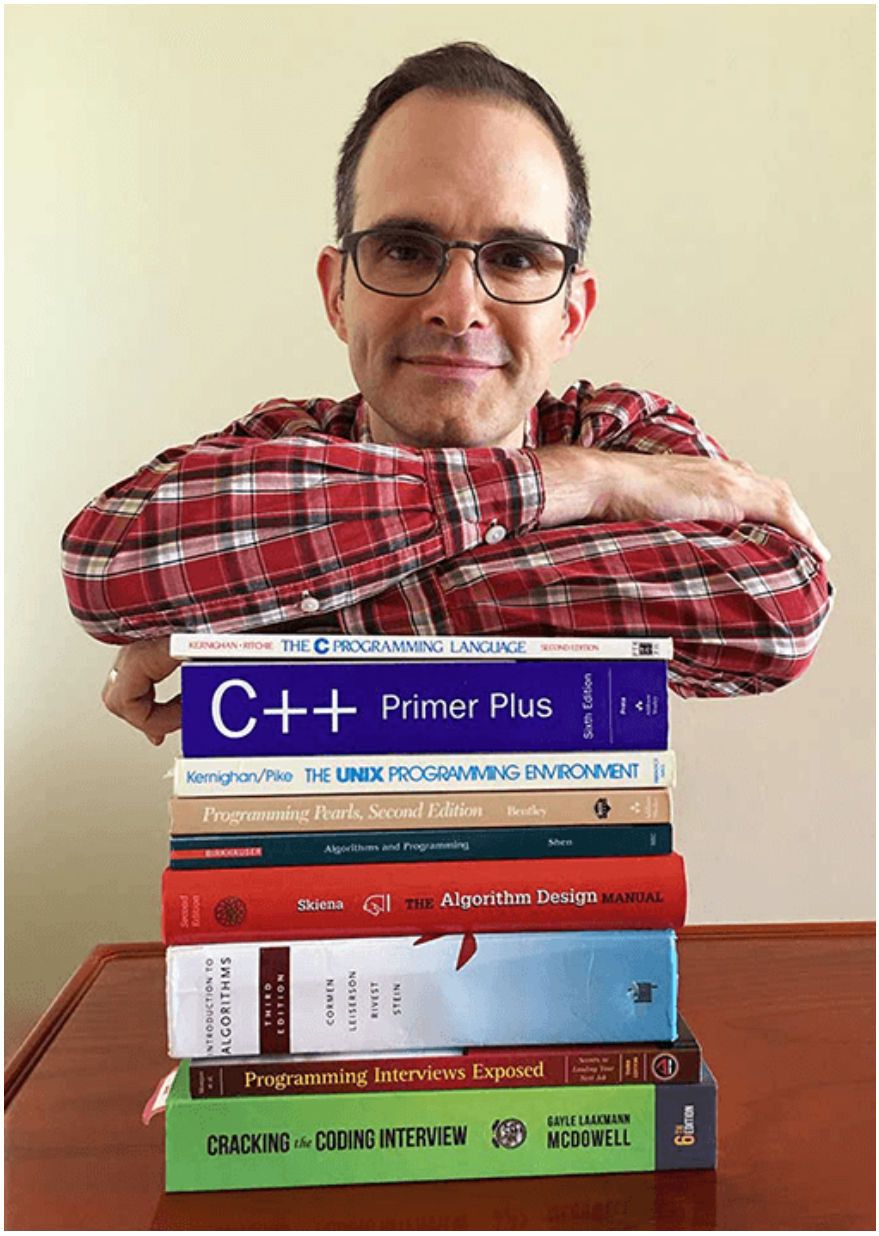
Google Interview University

Remember the coaching notes I received telling me what to study? The list of topics seemed manageable, even though I didn't know anything on the list.

I turned the topics on the notes into an outline and started filling in the topics with YouTube videos of lectures from MIT and UC Berkeley. A video on linked lists in one place, a video about queues in another. The list started to grow.

I published the list on Github because my Github account was pretty empty. Since all the code I wrote for my businesses and work was private, my Github account made it look like I didn't code at all. I needed to build up a portfolio. I originally called the project "Project 9894". Google launched on Sept 4, 1998. Hence the name. I later renamed it to "Google Interview University".

Over time I added some optional topics that I discovered along the way.



My summer reading list. A little over the top.

I was pretty amazed I had gotten so far in my career without even knowing how a CPU processed a program, how memory worked, or any of it. I had known “just enough” to be a success.

My little Github project started getting a few stars, and I published a blog post celebrating 20 stars.

One morning, I awoke to find it had grown to 120 stars. Someone famous had tweeted about it during the night, and that led to it ending up on the Github daily trending report. I was #1 trending on Github for a few days.

Many kind people reached out to thank and encourage me. It turns out there are thousands of people who want to not only work at Google but want to work as a software engineer, and this list was just the to-do list they needed.

It's now at over **21,000** stars.

I still can't believe it.

What If I Don't Get the Job?

It won't be the end of the world.

I've put the time and dedication into my studies for the goal of getting hired as a Google software engineer, but even if I fail, I'll still be armed with the skills and knowledge required to work as a software engineer at any company.

Wherever I end up, I'm going in as an entry-level software engineer. I'm not going in with 15 years of software engineering experience because I simply don't have it. When it comes to this stuff, I'm the equivalent of a fresh CS grad.

But I have the enthusiasm of a new grad, too. This is a new world for me. I'm just getting started. I'm not afraid to make mistakes. I know I will. I also want to learn everything I can and be an excellent addition to any team.

Don't Study As Much As I Did

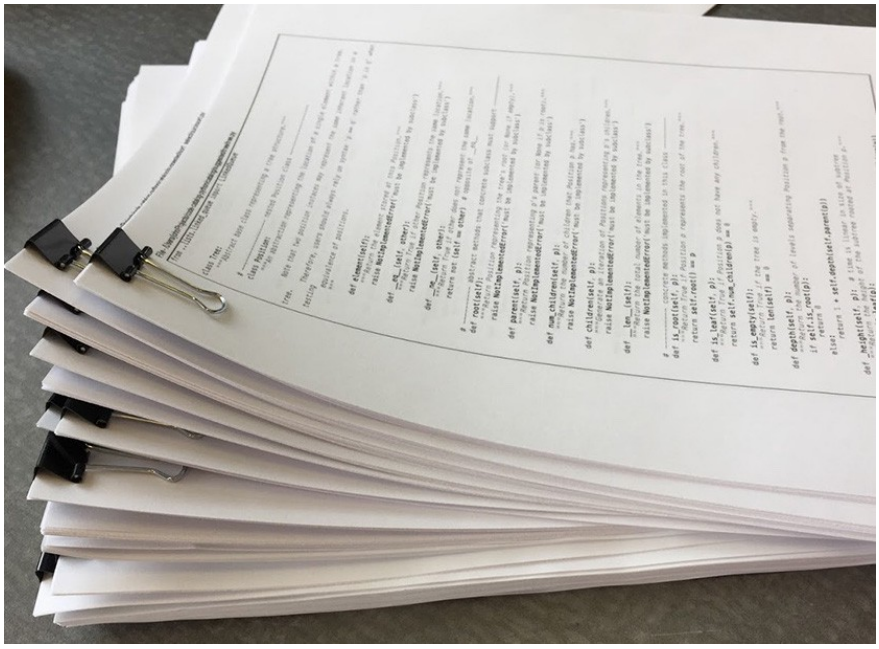
Yes, I took 8 months. But I could have abbreviated the process. Like any startup with a big goal, you make mistakes and do things that waste time. **There are many things I wish I go back and do differently.**

I studied topics I didn't need to, some because I thought I would need them for the interview, and some because I wanted to have the knowledge on hand for when I started working. I didn't want to be a burden on the team I'm assigned to. It turns out I simply over-prepared.

I spent 3 weeks reading a 1,000-page book on C++. I don't remember 1,000 pages worth, but I know a good bit about C++ now. As it turns out, I'm using Python for the interview, not C++. I had assumed I needed C++, C, or Java, but I was wrong. **It's good to ask, not assume.**

I read way more books than I needed to. There are only 3 or 4 books I should have read.

I have a code catalog of dozens of algorithms that I review, most of which I wouldn't expect in an interview. **You don't need to do that.**



A stack of algorithms, printed out for review.

I watched many hours of YouTube videos but could have watched far less, and spread out topics over time.

I should have stopped reading books and watching videos earlier and started on coding problems sooner. I would have been able to spend more time applying the topics I learned.

Spaced repetition is the key to memorization. Once you learn something, review it again later, and again even later. At each repetition, you reinforce your learning. Spending hours and hours at one time on priority queues won't make you an expert. You become

an expert by revisiting and reviewing over time. If you do so, you'll get to the point where can't forget details.

To help review, I made 1,792 flashcards (digital flashcards). **This is way too many.** I review them on my phone or tablet whenever I get a spare moment (such as during Christmas shopping). Flash cards and spaced repetition go hand-in-hand. Once I get an answer on a flashcard right, I don't mark it as known. I keep it in the deck and once I've seen it and answered it correctly many times, then I mark it as known.

My sense of fear ("What if they ask me a question about red-black trees?") led me to study far more topics than I needed to.

But I didn't want to just prepare for the interview, I wanted to prepare for a career at Google, solving large-scale problems. That means knowing algorithms that will save computing resources of time, space, and I/O.

I may never need to know a maximum flow algorithm (Ford-Fulkerson), but it's nice to know I have that tool available if the situation arises (without memorizing the implementation), and can recognize its application to a problem space.

Conclusion

Early on, I wished I could skip all this learning, and just hurry up and get hired so I could instead spend my time learning the languages and tools for the team I join. But along the way, I realized how important this knowledge is, and even though most of it may not be applicable on a daily basis, I'm glad I put in the effort. I have a new appreciation of the history of computing, the greats in the field, data structures and algorithms (and how they complement each other), and how computer systems work at low-level.

I'll be putting in my application soon. It's been a long journey getting to this point—almost an entire year. It began back in January, but I wasn't able to commit to full-time study until April.

I'm about as prepared as I can be. I can't keep studying and putting off the application forever. At some point, I have to take the leap.

I see a bright future ahead.

Thanks for taking the time to read my story.

Article also available in Arabic.

Where to Find Me

Googley as Heck: *https://googleyasheck.com*

Google Interview University on Github:

jwasham/google-interview-university

google-interview-university - A complete daily plan for studying to become a Google software engineer.

github.com