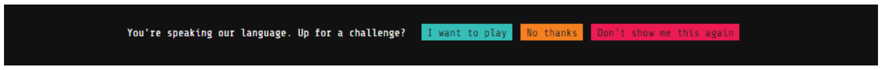
My Experience with Google Foobar: Tips for Tackling Google's Legendary Coding Challenge

Avery Durrant Follow
Nov 10, 2017 · 8 min read



You're speaking our language. Up for a challenge? I want to play No thanks Don't show me this again

Google Foobar Popup

In this article I will be explaining my time with Foobar and giving future Foobar challengers some tips and tricks for succeeding with this rigorous challenge.

What is Foobar?

For those of you who are unfamiliar with Google Foobar, it's a kind of recruitment tool for Google. In order to join Foobar, you must be invited by a friend or have it popup when searching certain CS terms on Google.

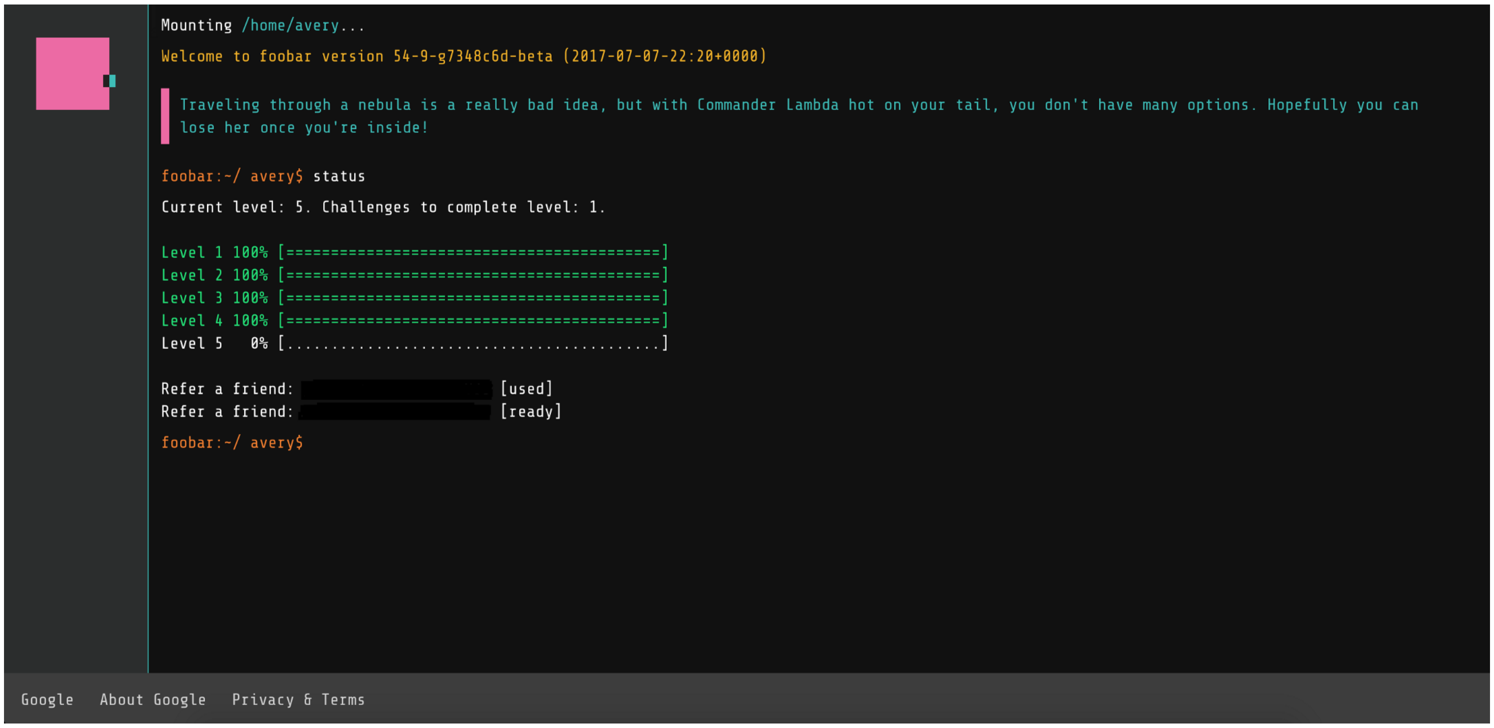
Once you accept the invite, you will be taken to a website in which you are given a question and a time limit and you have to finish the question within the limit or you fail. If you are able to solve the problem, you can request a new one and eventually move on to higher levels. But if you fail to complete a question, then you will no longer be able to request new problems.

When you request a question, you are given an explanation of the problem, one or two examples, and a few test cases. You then edit either a Java or Python file to solve the problem, and once you think you solved it you can verify your file and check if it passed all the test cases.

Not only do you have to pass the test cases given with the question, but also hidden test cases that are not shown. Once you pass all the test cases, you then can submit your solution and, when you are ready, request a new problem.

Foobar presents its questions through a cometic narrative, which makes doing the questions fun and silly. You are rewarded when submitted questions with the next chapter in the story, and the new problem that arises within the narrative. The story really helps the flow of Foobar because instead of just solving problems, you now have a task to complete in order to progress the story. It makes Foobar even more enjoyable.

The final goal in Foobar is to reach and complete level 5. But once you finish level 3, you are able to request a Google recruiter to review your code and hopefully get in contact with you.



```
Mounting /home/avery...
Welcome to foobar version 54-9-g7348c6d-beta (2017-07-07-22:20+0000)

Traveling through a nebula is a really bad idea, but with Commander Lambda hot on your tail, you don't have many options. Hopefully you can
lose her once you're inside!

foobar:~/ avery$ status
Current level: 5. Challenges to complete level: 1.

Level 1 100% [==========================================]
Level 2 100% [==========================================]
Level 3 100% [==========================================]
Level 4 100% [==========================================]
Level 5   0% [..........................................]

Refer a friend: [redacted] [used]
Refer a friend: [redacted] [ready]
foobar:~/ avery$
```

Google Foobar Website

My Foobar Journey

To get to understand my experience better, you should get to know my programming background. I have been coding for over 6 years in many languages including Java, Python, Javascript, C++, C#, etc. Within those 6 years, I was developing professionally off and on.

When I got prompted with Foobar at the end of August, I immediately requested my first challenge. I was given 48 hours to complete this problem, and it was quite simple. It was basically something given in any college level computer science course, and it took little to no time.

Once I finished the first problem I was promoted to **level 2**, and I was give two more problems with I believe around 72 hours to complete each on. These level 2 problems were a little harder then the first problem, but still not that hard. They were reskins of some harder problems you can find floating around.

Once I progressed past those problems, I was on **level 3** and I was given three problems. These three problems were where I started to slow down, and really think about the problems. **I fully understood the problems**, but the hard part now was implementation. On level 3, your code has to be extremely optimized in order for your solution to be correct. Your problem could pass the three test cases given, but once you verify it you could exceed the time limit or just fail hidden test cases. I was able to complete the three problems on Level 3, but they did take a lot out of me.

Level 4 is where Foobar really started to get intense. Level 4 contained two problems with around two weeks to complete each problem. *These problems were by far the hardest problems I have ever solved*. They took multiple advanced math and programming concepts in which you had to first understand the problem then implement it. This level was incredibly hard because they gave you less test cases, and their explanation of the problems were intentionally vague. You really had to understand the problem and have a lot of prior math and CS knowledge to solve the problems. I was able to complete these problems, but they took me until the last minute to complete and fully optimize.

Level 5 was where I was stumped by Foobar. I was given a little over three weeks to solve one problem, and I was regretfully unable too. A good part of that was improper time management and accepting the problem at a bad time. But it was mainly because how incredibly hard the problem was. I won't give the problem away, but they gave one test case, and the explanation was so incredibly vague that I had no clue where to even start the problem. I spent days on end drawing things on a whiteboard, but I just could not **understand the problem**.

I was not ashamed to be stumped by Foobar; in fact, I was actually quite proud. Foobar taught me so much, and the improvement from the start to end was incredible. It was truly one of the best programming experiences in my 6+ year of coding. And like I said before, I am happy to have done it.

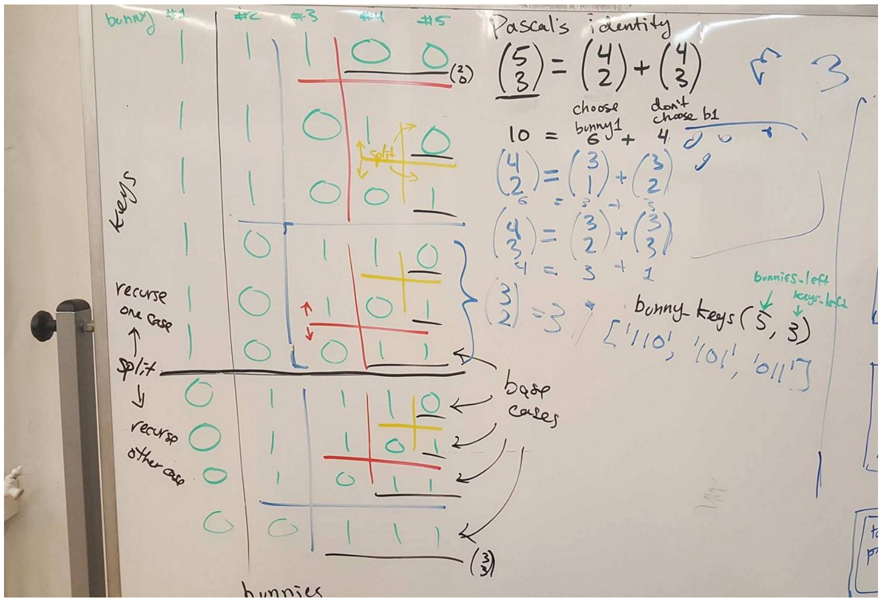
Tips and Tricks

Now that you have heard my story, I will be sharing some tips and tricks to future Foobar challengers.

Problem Solving

To excel at Foobar, you have to be very good at problem solving. In this section, I will explain my steps for solving a problem not just for Foobar but any problem you might come against (*whiteboard interview*).

First step to solving a problem is **understanding the problem**. This might sound kinda crazy, but it happens. You should not even touch any code until you fully understand the problem given, and you understand the problem enough that you can explain to someone else. The best way to understand a problem is to **draw a picture** that helps define a solution and be more aware of the problem. Below is an image I drew for one of the problems in level 4. Without this picture, I am sure I would have never solved the problem.



My picture for understanding a level 4 problem

Once you understand the problem, then you can **start to solve the first few test cases**. There is no reason to try to make a fully optimized algorithm; a solution that works is always going to be better then a optimized solution that doesn't work. This step should be the easiest, but can be argued as the most important.

Now that we solved all the test cases, **let's revise the problem and find all edge cases**. Foobar loves giving you a few test cases then when verifying the solution you fail 6 hidden test cases. To get around these hidden test cases, you should be testing your code with lots of your own test cases and finding all the cases that your algorithm doesn't do what's intended (edge cases). Then once you find all the failing edge cases, fix them.

Now that our problem passes all given and hidden test cases, **optimize and clean your code**. Make sure your code is clean with good variable names and comment, and make sure the code is optimized to be as fast and most efficient as possible with the least amount of space required. Once we finish this step, we can finally submit our problem!

After we submit our code, we should **reflect on the problem**. Think about what you did right, what you did wrong, and what you could have improved on. This greatly improves the quality of your problem solving technique in the future, and shows you know what you could work on if you had more time.

Lightning Tips

Now that you know how to problem solve, here are a few tips that are simple to do, but are very impactful.

1. One thing I failed to do during the last few problems on Foobar, is **making sure you have time to dedicate to your problem before you request it**. Foobar has a great feature that after you solve one problem, you don't get a new problem unless you request it. If you are like me and have multiple prior commitments, **don't request a new problem until you are ready**. The number one reason I couldn't finish the last problem is because I had so many things to do, and hastily requested a problem before one of the busiest weeks of my year. Please just wait until your calendar is open, and mark down time for each question. It will save you so much stress.

2. **Use all the time you are given on each question.** During the first few stages you can finish a problem in a matter of hours. That's great, but there are always ways to optimize your code. If you are submitting a problem early, you sure as hell must have great variable names, comments, and the best algorithm the world has ever seen. You can always play around with making your code more efficient and cleaner, and there really is no reason you should be submitting a problem early. Make more test cases, simplify your answer, do anything. Just don't submit a questions days before it's due.

3. A lot of people who do Foobar do it in order to get a job at Google. While this is what it was created for, I don't think that should be your mindset going into it. Foobar is one of the best tools you have to improve your coding skills in my opinion. **Go into Foobar to learn, not to get a job**. Foobar is great for recruiting top talents, but going into Foobar only thinking of a job at Google could taint your mindset and once you hit a wall, it will feel like life or death.

4. The number one advice I give to people about Foobar is: **don't start Foobar until you are ready**. I know it's cool to do a coding challenge for Google, but Foobar is cold and unforgiving. If you fail on the first few levels, there are no lives. If you fail you're done, and you can't come back. In my opinion don't start Foobar until you are almost as skilled as a **Google Software Engineer**. It's plain and simple: you have one shot at Foobar. Don't start it until you are ready!

Take Away

When I look back on my months doing Foobar, I learned a lot. Personally I thought Foobar was incredibly fun, and if I could do it again I would.

Since Foobar is over and I still have the taste for challenging problems, I have moved to the land of machine learning to quench my thirst. If you want to learn more, follow my Medium for an updates on my new journey or check out the projects I have been working on. You can also check out my LinkedIn.