

Your goals are probably unrealistic...but here's what you can do about it

Posted by *Sam Gavis-Hughson*



Again and again, I hear the same thing. "I want a job at [insert company with a rigorous interview process here]."

That's great! I'm glad that people know where they want to work, but oftentimes they fail to take a step back and ask themselves "Is my goal achievable?"

Occasionally this is an obvious yes or no. Someone with no coding experience getting a job at Google in 2 months? That's a pretty clear no. A brilliant student wanting to work at a small local dev shop? Yes.

But 99.999% of the time, it's not that clear cut.

To answer this, we'll ask ourselves two questions. "Where am I now?" and "How much work will I have to do to get where I want to go?"

Where am I now?

In my experience, this can be a really difficult question to answer.

First of all, knowing what to expect in the interview can be very difficult to judge, especially for those who haven't interviewed before. You can read about interviewing all you want but until you get there, you don't know what it's really going to be like.

The other thing that makes this a really hard question to judge is the Dunning-Kruger effect. This psychological principle says that the less you know about a topic, the more likely you are to think that you are good at it.

Think about that for a second. That means that the more you suck at interviewing, the more you think you're actually prepared.

Talk about counterproductive.

The more you suck at interviewing, the more you think you're actually prepared.

[CLICK TO TWEET](#)

Because of these factors, there is nothing more valuable than soliciting the help of someone who has been there before and knows the system. That's why I recommend mock interviews.

Mock interviews give you a chance to experience what the interview process is going to be like and experience it for yourself. You get feedback from someone impartial who can help you see where you really stand.

In particular, I recommend doing at least 1 mock interview when you're starting your interview prep process. I find that most people want to leave the mock interviews until the end and this often leads to heartbreak.

People come to me a week before their interview and want to do a mock interview. If they do it and it doesn't go well, they now don't have much time to fix the problems before the real interview. All it does is shake their confidence.

However, if you do a mock interview well in advance (ideally 3 months), you will be able to establish a baseline and set goals around it. I still recommend doing more mock interviews later as part of your interview prep process, but getting at least one in early

is critical.

There are a couple of different good resources for mock interviewing:

- Friends. This is an easy and free option. If you have a friend who has been through the process before and is willing to interview you, then great. Just be sure that they are willing and able to give you honest critical feedback. In this case, it **is** important to have an interviewer who knows about the sorts of jobs you're looking for so they can give you useful feedback.
- Pramp. Pramp is another free option that will pair you with fellow interview preppers so that you can mock interview each other. The main limitation is that you're primarily interviewing with those at similar experience levels to you, so they may not be able to give you as in depth feedback. But again, it's free!
- Interviewing.io. This is a free option that pairs you with engineers at different companies and promises to help you get jobs as well. They are currently in private beta, so this option may not be immediately available.
- Gainlo. This is the best choice for when you really want to invest in yourself. It's a paid service, but you get to interview with professional software engineers at companies similar to those you will be interviewing at. This is really the best option for when you want to get serious.

Through the mock interview, your goal is to figure out your starting point. Maybe you nailed the interview. That means you're probably already in pretty good shape. If you really struggled, then you have your work cut out for you.

The key is to get honest feedback from your interviewer about how well you did and combine that with your own experience. How did they think you did? How did you feel?

Once you figure out where you currently are, the next step is to figure out how far you need to go to get to your goal.

How much work will I have to do to get there?

Are you a senior CS major at an elite school? In that case, if you've been keeping up with school work you likely won't have to do so much extra studying. You can just bone up on a few concepts and learn some interview skills.

Did you just complete a bootcamp after having never done any coding? You're definitely going to need to do some serious studying of the core concepts if you didn't heavily cover those during the bootcamp.

Figuring out what you need to do really just comes down to looking at the requirements for you to achieve your goal and figuring out what requirements you have yet to meet.

I recommend looking in the following places:

- Job descriptions. Here, you can see what companies at least think they are looking for.
- Other's experiences. What happened when other people interviewed? What sort of questions did were they asked? Did they pass or fail the interview? A good resource for doing this is Glassdoor, since they aggregate many interview experiences from different companies. Look at the sorts of interview questions people were asked and work through them on your own. How did you do?
- General knowledge. For this, I recommend *Cracking the Coding Interview* [Affiliate link]. It provides a really good overview of all of the skills that you need to know to get a job at any top company. If there are any topics in the book that you feel you are not strong at, you should consider that something you need to review.

When reviewing individual skills, consider the depth of knowledge that you need in different topics. If a topic is small, it will likely take a short time to get up to speed, whereas more in depth topics will take longer. I recommend reading the overviews of each topic in *Cracking the Coding Interview* [Affiliate link] to assess perceived levels of difficulty.

If you get stuck in this area, this is a place where a coach can provide enormous value. Coaches have worked with many students and know what topics they tend to struggle with. They also know what skills you need and what level you need to get to for

different types of jobs. Based on your current experience, they can make recommendations about how long they expect various areas to take you. I offer 1:1 coaching services here.

Is my goal realistic?

The final thing we have to do is combine our answers to the previous two questions and say whether we can realistically accomplish a given goal.

The question is not just **can** you accomplish the goal but **will** you?

This is the point where honesty is key. "No" is an acceptable answer!

If you're willing to put in the work, that's great. That means your goal is realistic. Maybe you're already close to where you need to be or maybe you are prepared to study full time for 8 months. Whatever it is, then go for it!

If that doesn't describe you, though, your goal may be unrealistic.

Let me give you an example to illustrate why honesty is key. Let's say that I set out to run a marathon next month. I am not a runner by any stretch of the imagination, nor do I particularly enjoy it.

To get adequately prepared, maybe I decide that I need to run 10 miles every day. If I just do that, then I'll be totally prepared.

But the catch is there's no way in hell I'm going to do that. I don't like running and definitely won't wake up early so that I can run for 10 miles. I know myself well enough to know that. Willpower might work for the first week, but after that, it's definitely not going to happen.

Therefore this goal is not realistic, despite how much I try to tell myself to the contrary.

If you've gotten to this point and decided that your goal is in fact realistic, then you can achieve your goal. You know roughly how much work it is going to take to get there and you are willing to put in the effort.

However, as is just as often the case, what if your goal isn't realistic?

My goal isn't realistic. How do I find one that is?

When I see a goal that isn't realistic, it almost always comes down to timeframe.

"I want to get a job at Google by next month."

It's not the getting a job at Google that makes this difficult, although that's no easy feat. The difficulty comes from not having enough time to prepare. You'd have to be studying 24/7, which most people are unwilling to do.

However, what would your studying have to look like if you wanted to get the job next year? Maybe 30 minutes a day. Maybe an hour, depending on where you're starting from. That sounds a lot more doable, doesn't it.

By changing the timeframe of your goal, you can make it much more realistic and achievable.

This is why I recommend a technique I call **Laddering Up**. Often times I talk to people and they want to get their dream job right now. But when we go through the exercise above, that is clearly not realistic.

Laddering up is simply starting at a company that is not your dream company and incrementally working your way up to better companies to ultimately reach where you want to be.

I love this strategy because you learn so much more by doing than anything else. Anyone who has worked as a software engineer knows that almost as soon as you start your job, recruiters start trying to steal you away.

It turns out that having an engineering job is the best way to get an engineering job. This is awesome. You're developing your skills and getting paid and it gets recognized almost immediately.

Therefore, laddering up serves dual purposes. First, it greatly expands the time frame you have to prepare for your dream job, since it takes off any financial pressure. And second, it may actually help you get to your dream job even faster because you're developing skills as you go.

If a job at your dream company is not realistic right now, that doesn't mean you can't get there in the future. The key is recognizing that just because you're not ready to work at Google right now doesn't mean that it can't happen in a few years time.

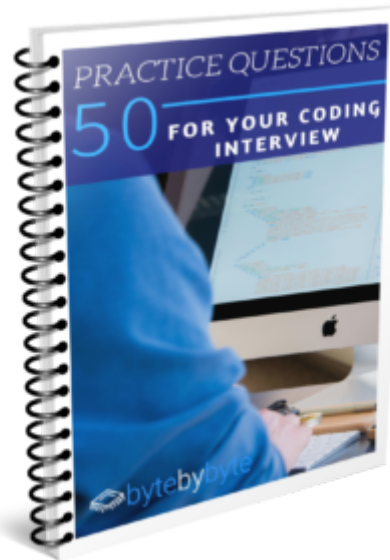
If you follow the steps I outline in this post, you will find yourself much more able to assess whether your career goals are realistic. Sometimes it's hard to admit that you're not where you want to be, but accepting that now can save a ton of frustration down the road.

I encourage you to follow this process and clearly assess your goals. In the comments, tell me about one of your career goals and whether it's realistic or not. If not, how are you going to ladder up to it? I look forward to hearing your experiences.

Don't do another coding interview...

...Until you've mastered these 50 questions!

GET YOUR FREE GUIDE



Sam Gavis-Hughson

Sam, founder of Byte by Byte, helps software engineers successfully interview for jobs at top tech companies. Sam has helped thousands of students through his blog and free content -- as well as 400+ paying students -- land jobs at companies such as Google, Amazon, Microsoft, Bloomberg, Uber, and more.

← [The only 6 types of questions you need to know to ace any coding interview](#)

[Coding interview roadmap: How to develop a foolproof interview study plan](#) →

0 Comments

Byte By Byte

1 Login ▾

 Recommend Tweet Share

Sort by Best ▾



Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name

Be the first to comment.

ALSO ON BYTE BY BYTE

Coding Interview Question: Consecutive Array

7 comments • 3 years ago

**Hitesh** — #include "QuickSort.hpp"#include "sortingArray.hpp"#include <stdio.h>#include <iostream>void**The reason why you should always find a brute force solution first**



2 comments • 3 years ago

**Sam Gavis-Hughson** — That's certainly true that it doesn't always help you find the optimal solution, but I think that in that case,**Private: How to use Cracking the Coding Interview effectively**

2 comments • a year ago

**Sam Gavis-Hughson** — 6.**Coding Interview Question: Find Duplicates**

4 comments • 2 years ago

**AASHISH KUMAR** — nice reasoning Subscribe  Add Disqus to your site Add Disqus Add

Search ...

DYNAMIC PROGRAMMING CRASH COURSE FOR NON-GENIUSES

Download my free guide to learn:

How to finally “get” what Dynamic Programming really is – no Ph.D required

The not-so-obvious way you can solve any dynamic programming problem fast – and not freeze up during your interview

The only 10% of information you need to know to ace your interview – forget all the useless fluff

Enter your email below and get instant access to your free Dynamic Programming guide.

GET THE FREE GUIDE

RECENT POSTS

How To Nail the Amazon Interview: A Practical Guide

The Ultimate Guide to Dynamic Programming

Behavioral Interviews for Software Engineers

Acing the Google Interview: The Ultimate Guide

INTERVIEW CAKE



Interview Cake is an awesome resource for more practice interview questions. Get 50% off for a limited time.

CRACKING THE CODING INTERVIEW



Check out my hands down favorite resource for coding interview prep here.



Made in NYC



sam@byte-by-byte.com



[Youtube](#)



© Byte by Byte 2016-2019

[Privacy Policy](#)
[Terms and Conditions](#)

Sam Gavis-Hughson is a software engineer based in New York City. Through Byte by Byte, he publishes regular coding interview question videos, demonstrating proper interview techniques. He also helps many students by offering practice coding interviews to help them get jobs at Google, Facebook, and other exciting tech companies.