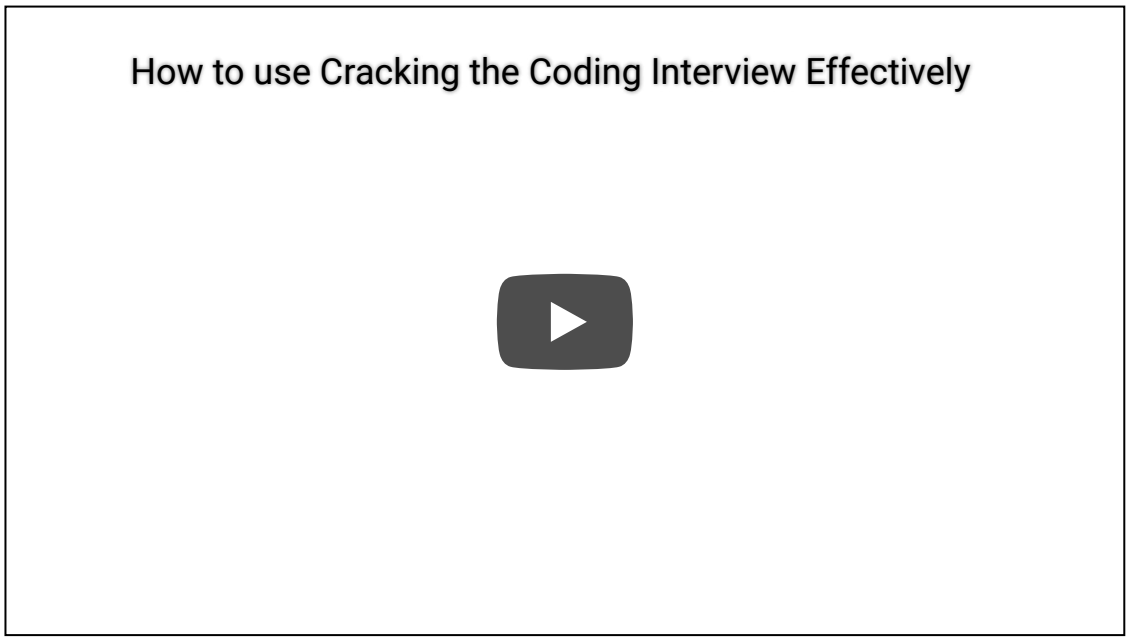
# How to use Cracking the Coding Interview effectively

Posted by *Sam Gavis-Hughson*




What's up everyone. Sam here from Byte-by-Byte.com and today I'm going to show you how to make the most out of this book. So how do we use Cracking the Coding Interview most effectively? And maybe a good question to ask now is, why do we really need a video on how to use Cracking the Coding Interview effectively in the first place? I realize that it seems a little weird that we would have a video on how to use a book. But I consider Cracking the Coding Interview to be much more of an encyclopedia than it is like an easy read where you would sit down in bed and read the book. And in the same way as with an encyclopedia you might want to have a guide for how to use the encyclopedia effectively for research. I think that it's really important that we understand how to use Cracking the Coding Interview properly for preparing for our interviews.

So there are three things that I want to talk about today. The first thing is what chapters of this book do you actually need to know. When you look in the index of this or in the table of contents for this book you'll see that if you have the most recent edition there are actually 17 chapters. And that's not including the solutions and not including all of the other you know valuable stuff at the end. So if there are 17 chapters do you need to go through all these 17 chapters? Well no you don't. And this is I think a really important point that people don't realize. There are chapters in here that are not

going to be relevant to you. If there are topics that you know really really well you don't necessarily need to go through the chapter on that just because it's included in the book. And in the same way there are chapters that I believe probably don't need it to be in the book in the first place. For example chapter 6 in the most recent edition is math and logic puzzles. And this is just something that companies haven't asked in years. I think that it's something that a lot of people are scared of which is why it's included in the book because questions like how many ping-pong balls fit on a 747, or how many windows are there in New York City seem like difficult questions to ask to answer. But they're really just so irrelevant at this point that there's no need to even go through that chapter.

Then there are also chapters at the end of the book which hopefully are going to be apparent. That they are on very specific topics. And a lot of these topics you don't really need to go in depth into, unless that's something that you are already an expert in. So for example if you were doing your interviews in Java you wouldn't need to study the chapter on C and C++. And that leads me to my sort of core set of chapters that you should focus on. So the core chapters of this book that you really need to know are chapters 1 through 5 and 7 through 10. And this is in the most recent edition.

So chapters 1 through 5 and 7 through 10 basically cover all of the fundamentals that you would need to know for your coding interview. And just to go through them quickly chapter 1 is stacks and or sorry chapter 1 is arrays and strings, chapter 2 is linked lists, we have stacks and queues, trees and graphs, bit manipulation which a lot of people think you can just skip over, but is a really valuable thing. You don't have to go super deep in it but it is a valuable thing. Object-oriented design, another thing that may or may not come up but is actually really valuable to have a good grasp on. Recursion and dynamic programming and with dynamic programming you can also check out my ebook on dynamic programming if you haven't already. System design and scalability which is going to be more relevant if you are further on in your career but it's still something that's worth covering. Sorting and searching and that's the first ten chapters, skipping the math and logic puzzles. And then other than that there are a couple other chapters that you could go into. You could go into a language specific chapter, if you're gonna do a lot of stuff with databases. Or if you do a lot of stuff with databases. Covering the database chapter would be helpful and there's also the medium and hard problems at the very end that are gonna be useful if those if you get

through everything else. But focusing for the starters on those first ten chapters and skipping chapter six is a really good way to get started. The second thing I want to talk about is how you actually approach this book.

So as you can see this is a really thick book. There's a lot of stuff in here. Thankfully maybe most of this is solutions. So this entire portion of the book is the solutions to the problems. But what that means is that there's a lot here that you could go through. So how do you organize your time? And how do you decide where to focus your energy? Well there are two different approaches that you can take to this. And I think that they're both very valuable depending on how you like to learn. So these are basically a breadth-first and a depth-first approach to the book. In the depth-first version, that's probably what you would initially think of doing.

You start with the book at the beginning you start with chapter 1, and you've read the whole chapter. You study that material and then you do the practice problems. And once you feel really good about arrays and strings you move on to linked lists. And then you move on to stacks and queues and etc etc. So that's one way that you can approach the problem the book. But what that fails at is that it may be very difficult to portion out your time. Especially because some of the later chapters are harder. If you break up your time evenly for each chapter you may find yourself running out of time as you're getting closer and closer to your interview. So that's something to be very aware of if you're going to take this approach.

You probably want to assign less time to the earlier chapters and then leave additional time for things like, dynamic programming and recursion as you get later on. The other approach which I like is the breadth-first approach. And this basically means what you would expect it to mean. You're gonna go through all the chapters a little bit, and then you're gonna circle around and you're gonna keep doing this over and over again. So you'll spend let's say you spend one day on chapter one. You can spend half an hour studying the material. You could go in a little bit of depth. You might go look at some other materials on arrays and strings, and then you do one or two practice problems. Depending on how much time you have. If you have an hour, you spend half an hour studying, half an hour doing practice problems. If you have two hours you can spend an hour on each. You could also spend 30 minutes one day studying and then thirty minutes the next day of doing practice problems.

Basically the point is that you're just going to spend a little bit of time and then you're gonna move on to the next topic. And what you're gonna do is you're gonna continue rotating through topics day after day, until if there are topics that you feel really good about you can remove those from the rotation and focus more on the other topics. Or until you get to your interview. And the goal here is to make sure that you have a good breadth of knowledge of all of these topics. It's much more important that you have that breadth of knowledge than you are really strong on graphs, but you don't know anything about recursion or dynamic programming. Because if a question on that comes up then you're gonna have no idea what to do. It's better, especially if you have good interview skills, to have a moderate amount of knowledge on everything and then be able to apply those skills so that you can work through the problem in your interview, and not get totally stuck. Whereas if you didn't have any knowledge on that topic you might get totally stuck and not know what to do.

So I really like this breadth-first approach to Cracking the Coding Interview because I think that it makes sure that you are covering everything you need to cover for your interview, at least in a little bit of depth. And the final thing that I want to talk about with Cracking the Coding Interview, is just the fact that this book is meant for review. This book is not the full encyclopedia. It's not your computer science textbook. It is a textbook for interviewing but that means that you are going to have to go outside of Cracking the Coding Interview to get all of the knowledge you need. If you just study exactly what's in the book, you're not going to go deep enough. And yes looking at yes the book does include a pretty good list of topics, but it doesn't really cover those topics in great detail. So as you're studying I highly highly recommend that you go reach out beyond the book and use the book just as a guideline for what topics you need to study in more depth.

For example with dynamic programming, just go back to this example, you'll see that there are some stuff in the book. There's an example or two. And it shows you tells you a little bit about what dynamic programming is. But that's not gonna help you really get good at dynamic programming. If you really want to get that good at dynamic programming you should check out my ebook, or you should check out other resources that are going to be more helpful for you and go into more depth. So on the day that you were working on dynamic programming you might reach out beyond the book and

go look at those other resources. And that's just a really important point because I think that some people view this book as being the entire sum of knowledge that they're gonna need for their interviews.

And unfortunately just because that's not plausible in a book. Like this book would be ten times as thick if that was true. And so it's just not possible to do that. And so that's a good way to make sure that you're getting all of the knowledge in there. So just to recap real quick. With Cracking the Coding Interview, I think it is the best book for prepping for interviews and I highly highly recommend that you get a copy if you haven't already. And it doesn't even matter you don't even need the most recent edition any edition will do. It's just a really good starting point.

When you're studying the first thing I recommend is do chapters one through five and seven through ten. Skip chapter six, skip the remaining chapters unless there is specific knowledge in there that you need for your interview. Secondly, decide if you're going to take a depth first or a breadth first approach. I like the breadth first approach where you cover all the topics and then continue to cycle through them until you get to your interview. Because that makes sure that you actually cover all the topics that you need to.

And third, make sure that you go outside of Cracking the Coding Interview and look at the other resources that are out there. Use it as a guideline for what topics might come up in your interview, but don't let it be the end-all, be-all in terms of the actual stuff you need to know.

So I hope those tips were helpful for you. If you haven't already please subscribe to this YouTube channel so that you can get notifications for all of these videos. And also if you haven't go over to DynamicProgrammingBook.com and you'll be able to download my free ebook on dynamic programming which will take you a lot deeper on that topic. And I other than that I look forward to seeing you guys again in the coming weeks.

# Don't do another coding interview...

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

## 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.

← How to be productive while studying for interviews

Don't Be a Google Software Engineer →

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.

sam@byte-by-byte.com

Youtube

Made in NYC



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.