<https://leetcode.com/discuss/general-discussion/494279>

Trying to make a short, complete and realistic DSA study guide for coding interviews as well as competitive programming. Studying for competitive programming is quite vast, so we will try to focus more on coding interviews as well as some overlapping study materials for competitive programming. Study materials depend mostly on the position you are looking for, say for example, i have seen people been asked on rope data structure, hopcroft-karp algorithm, max-flow algorithm, hungarian algorithm etc. others non trivial stuffs. Though those are for L4 or L5 positions and DSA matters less there(System design does). So it depends. In this study materials guide we will be writing for mostly entry level or at most 1 or 2 level up from entry (FAANG mostly).

I won't give a massive list of video links for each topic rather it will be short and concise with some tips, so that one can finish this guide within a reasonable amount of time. And i will try to categorize topics so that one can find his/her weakness easily and use this guide to work on it. And remember **Only Practice** can give you a ***sense of completeness*** for a specific topic. And **Don't Rush. Enjoy the journey. Try to be a natural problem solver not a interview acing problem solver. And you are not alone! Keep moving mate, never ever dare to give up. Hard work is going to pay you off very soon.**

***N.B:****CI = Coding Interview, CP = Competitive Programming, DSA = Data Structure and Algorithm, LC = LeetCode, CLRS = Cormen, Leiserson, Rivest, and Stein, BFS/DFS= Breadth/Depth First Search, DP = Dynamic Programming.*

**Here is a Straightforward Study Guide PDF if you don't have time to read whole article. PDF contains only links to study materials. But i will highly recommned you to study the article first and then use the PDF.**

**Link for PDF:** [Comprehensive Data Structure and Algorithm Study Guide PDF Format](https://raw.githubusercontent.com/aminPial/zPixel/master/CI_DSA_study_guide.pdf)  
**Preview of the Study Guide:**

**Time Complexity**

Video no. 1-16[Abdul Bari's Algorithm Playlist](https://www.youtube.com/playlist?list=PLDN4rrl48XKpZkf03iYFl-O29szjTrs_O)  
  
**Comment:** After watching this 16 videos i can guarantee that you will gain mastery on Time Complexity for sure.

**Data Structure**

[Data Structures Easy to Advanced Course - Full Tutorial from a Google Engineer and ACM ICPC World Finalist.](https://www.youtube.com/playlist?list=PLDV1Zeh2NRsB6SWUrDFW2RmDotAfPbeHu)  
  
**Comment:** This is one of the best materials to study on data structure topic. William implemented each on Java. But it doesn't really matter which language you use, i did this course in both in c++ and python. And try to code yourself after watching a data structure topic and do some leetcode question on that. And this is all you need to ace DS questions.

**Algorithms**

The **Major Five topics** are:-

1. Ad hoc/ Implementation Problems
2. Programming Paradigm(Greedy, backtracking, branch and bound, DP, Divide and Conquer, Brute force etc.)
3. Graph Theory(directed, undirected, weighted, rooted(IN & OUT) and unrooted tree, DAG etc.)
4. Math(Number theory, Computational Geometry, Combinatorics, Linear Algebra etc.)
5. Others(String Processing, Bit Manipulation etc.)

Implementation Problems

Practice, Practice and Practice! Besides LC try to do some problems on other platforms for this. I will recommend doing **Implementation tag section on hackerrank**. After solving from [Implementation Category of hackerrank](https://www.hackerrank.com/domains/algorithms?filters%5Bsubdomains%5D%5B%5D=implementation&badge_type=problem-solving) you should feel comfortable to ace implementation questions. Again for this type, only practice can guarantee success. And that's it.

Programming Paradigm

Now, this is a huge deal for all of us, right?. So many topics to cover. But if you look closely the basic to ace this paradigm based question is **Recursion**. So before even try to understand DP, D&C etc. **Understand Recursion** and then come back. Recursion is the open secret tool to ace this type.

Best materials for recursion

[Recursion Playlist by mycodeschool](https://www.youtube.com/playlist?list=PL2_aWCzGMAwLz3g66WrxFGSXvSsvyfzCO) and **video no. 18 to no. 29** from [Abdul Bari's Algorithm Playlist](https://www.youtube.com/playlist?list=PLDN4rrl48XKpZkf03iYFl-O29szjTrs_O)***to understand Masters Theorem for the proof of recursion.***  
And another super important thing on recursion is to understand types of recursion like tail, head, nested, tree(the one you need everywhere) etc. You can study from [sparknotes tutorial on recursion types](https://www.sparknotes.com/cs/recursion/whatisrecursion/section2/" \t "_blank) or may follow a textbook. I will strongly recommend to study and master **Chapter 4 | Divide and Conquer | Page No.65 from Introduction to Algorithm by CLRS**. You have to spend sufficient time to understand recursion through studying and practicing, as recursion is will be the base of everything in this type.  
  
**Some Notes on recursion:** Almost everyone knows what recursion is, right? But that is not enough. You have to create some sort of mental model how recursion actually saves states by pushing function code to stack and reaches at the last/smallest problem and then solves it and then backtrack from there by poping function code from stack to top and etc. Hope those materials above will clarify everything.

Now about studying each topic of Programming Paradigm:

Divide and Conquer

Implement merge sort, segment tree, binary search etc. And study **Video no. 18, 33 to 38** from [Abdul Bari Algorithm's Playlist](https://www.youtube.com/playlist?list=PLDN4rrl48XKpZkf03iYFl-O29szjTrs_O)

Greedy

This is tough one. Proofing greedy algorithm is quite difficult. Studying known problems like knapsack, job schedule, optimal merge pattern, Huffman coding etc are enough to ace greedy questions. Study **Video no.39-no.45** from [Abdul Bari Algorithm's Playlist](https://www.youtube.com/playlist?list=PLDN4rrl48XKpZkf03iYFl-O29szjTrs_O)

Backtracking & Branch and Bound

Study **Video no.63 to no.71** from [Abdul Bari Algorithm's Playlist](https://www.youtube.com/playlist?list=PLDN4rrl48XKpZkf03iYFl-O29szjTrs_O). This topic is the **key ingredient** to solve Dynamic Programming questions.

Dynamic Programming😭

The big guy enters! After watching some top problem solver's code i get too much frustrated. They came up with all this sub problem based formula and solves it like a beginner question(Just kidding! they don't. It's because of their years of practice to recognize the pattern/formula for a dp problem). Say for example: A String based DP problem involves a 2D matrix where [i][j] generally refers to the solution for index i to j of the String and etc. Here is what you should do, try to understand backtracking very well as that will be the key in solving DP. After getting a backtracking solution you can memoize the previous solutions and reduce solutions to 2/3 Degree Polynomial Time. By the way, there is a **good news for Pythonistas**. After you just come up with a **2N** backtracking solution just use **functools.lru\_cache(maxsize=None) decorator and you will have a dp solution(almost 90% time)**. *More info here at:-*  
  
[What is memoization and how can I use it in Python?](https://stackoverflow.com/questions/1988804/what-is-memoization-and-how-can-i-use-it-in-python)  
  
Anyway, ***you have to study known DP problems as much as you possibly can and try to recognize the patterns and types***. *Here is the post that will do that for you:-*  
  
[Leetcode Coin(giveaway) winning post on Dynamic Programming Patterns by aatalyk.](https://leetcode.com/discuss/general-discussion/458695/dynamic-programming-patterns)  
  
*Study and solve all questions from here. Just stick with it till the last question of this article. And when studying the article try to follow:-*  
  
[Tushar Roy's Dynamic Programing Playlist](https://www.youtube.com/playlist?list=PLrmLmBdmIlpsHaNTPP_jHHDx_os9ItYXr) and **Video no.46 to no.60** from [Abdul Bari Algorithm's Playlist](https://www.youtube.com/playlist?list=PLDN4rrl48XKpZkf03iYFl-O29szjTrs_O).  
  
**I find Abdul bari's tutorial more effective and easy to follow. His style to teach students is quite exceptional.** Suppose you are studying Longest Common Subsequence, first understand the question really good -> try to solve a small problem of the main problem -> try to solve a bit big problem with the help of solution and see if you can find any formula/pattern -> if you can't find any then read discussion/solution(only algorithm not code) and try to code it up after understanding -> If still doesn't work for you then watch the video of that topic from the playlist i have mentioned and try hard this time to understand and visualize the algorithm. -> You solved a DP Question! Yahoo!.  
  
Now ***one of the most important study material*** for DP. How many of us know that a dynamic programming is nothing but a **topological sort of problem dependency directed acyclic graph** which means if you can generate a test case for a DP problem that has a cycle then that DP solution will fail for that cyclic graph. *To know all of this cool things and understand DP really good then study:-*  
  
**video no.19(MUST MUST!),20-22,26-27,39-45** from [MIT OCW Introduction to Algorithm](https://www.youtube.com/playlist?list=PLUl4u3cNGP61Oq3tWYp6V_F-5jb5L2iHb)  
  
and this will be enough!

Graph Theory

[Graph Theory Easy to Advanced Course - Full Tutorial from a Google Engineer and ACM ICPC World Finalist](https://www.youtube.com/playlist?list=PLDV1Zeh2NRsDGO4--qE8yH72HFL1Km93P)  
  
**Comment:** There are so much overlaps in between greedy, dp with graph theory. Say for example Dijkstra, Prim's and Kruskal's Minimum Spanning tree are just Greedy Algorithms or backtracking is just DFS with branch pruning with condition. So you will find it a lot easier after studying programming paradigm section. In fact graph problems are either so easy to recognize that everything is given so explicitly that any one can recognize it as a typical graph question or may be it's too hidden to even think it as a graph question. So my suggestion is to think out of the box for a problem, think if a problem can be solved by using graphs. Never forget that, Interviewers are just obsessed with binary tree, so try to solve as many questions as you can related to tree, specifically binary tree(and also n-ary tree). And also solve at least 20 questions with tag BFS and DFS in Leetcode which will definitely boost your tree and graph problem solving skill as graph traversal is the main toolkit to solve tree/graph problems in interview. That's all about graph.

Math

Math is fun, Math is everywhere and Math can win you any war from coding interview to WW2(Remember Alan Turing and enigma!). Math problems are more common to competitive programming environment. Sometimes interviewers will ask you for proof by induction or contradiction for a problem. And even if they don't and you can show the proof it will be a huge boost to your success. *The best material to study:-*  
  
[MIT 6.042J Mathematics for Computer Science, Spring 2015](https://www.youtube.com/playlist?list=PLUl4u3cNGP60UlabZBeeqOuoLuj_KNphQ) and also try to examine your understand through their quiz and exams from [Mathematics for Computer Science MIT OCW main site](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-spring-2015/).  
  
*And CP(or may be CI) guys must read the****Algebra Section****from here:-*  
  
[English translation of e-maxx awesome algorithm text tutorial.](http://cp-algorithms.com/)  
  
This will be enough for the math topic at least for Coding Interviews. But may be not enough for CP guys.

Others

1. **String Processing:** Follow the string section from [CP Algorithms Site.](http://cp-algorithms.com/)  
2. **Bit Manipulation:** Follow [HackerEarth Bit Manipulation Tutorial](https://www.hackerearth.com/practice/basic-programming/bit-manipulation/basics-of-bit-manipulation/tutorial/" \t "_blank) and also [TopCoder Fun with Bits Tutorial.](https://www.topcoder.com/community/competitive-programming/tutorials/a-bit-of-fun-fun-with-bits/" \t "_blank)  
***[Comment if you have any other topic in mind]***

***Nota Bene(N.B):*** I didn't cover the study materials for advanced DS like SegTree, Fenwick tree/ BIT, Sparse Matrix etc. And also some non trivial algorithms. Don't worry about these, you won't face them in an interview in 95% case at least for entry levels or a bit higher from entry. But if you are studying for CP, good luck, just study everything you see on the fly.😛 And you might have seen i tried to categorize CI and CP guys as to differentiate the volume of study materials for each of them.

**Tips:**

While 1:

print("Deliberate Strategic Practice")

print("Study and implement New DSA")

print("Simulate real environement and test yourself") # do contest, contest and contest

and its cliché i know! But it's the only way. There is no secret sauce really. Suppose you just read backtracking from an article or may be a video, then why not solving 10 DFS and Backtracking tagged question in LC at a stretch ?. In university, i saw my classmates and seniors with no cs knowledge became red coder in CP platforms within ~1-2 years. Their secret is nothing but **Regular Deliberate Strategic Practice**. They always try to solve questions beyond their comfort zone and do it regularly. And that is it really. Always solve questions that you are not comfortable with, push yourself. Try to solve topic wise after you get a good general grasp of DSA. Find your weakness and work on it. Hard work will pay you off one day.

Other comprehensive CI (and CP) prep guides: (*Comment if you know any good material*)

* [CP and CI Study materials by Jasmine Chen(Google SWE). This whole write up would be incomplete without this link.](https://github.com/lnishan/awesome-competitive-programming)
* [Tech Interview Guide (This one is just gold!)](https://yangshun.github.io/tech-interview-handbook/introduction/)
* [Great guideline on acing interviews and what to study?](https://medium.com/@nick.ciubotariu/ace-the-coding-interview-every-time-d169ce1fd3fc)
* [DSA for CP guys(CI guys should also try if you have time)](http://cp-algorithms.com/)

Just want to finish with one of my favorite quote by Mark Manson:  
  
***""I wanted the reward and not the struggle. I wanted the result and not the process. I was in love not with the fight but only the victory. And life doesn't work that way.""***

Hope this will help you a bit to ace your next coding interviews or competitive programming contest. Any suggestions on improving materials or adding materials will be highly appreciated.

Thank you guys. I am from non CS background(studying Materials Enginnering as Major at Uni) and don't feel bad if you are also from non CS background as end of the day your curiosity and persistance is all it matters. I am quite new to this platform(CI platform to be specific) and more used to CP platforms. You guys are the best community i have ever seen. Cheers! Happy Acing Interviews!