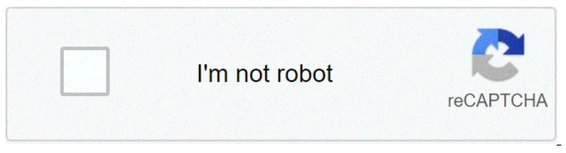

Continue

Introduction to algorithms and data structures

Data structures and algorithms are essential for any programmer. I strongly believe that a thorough knowledge and skill of these two topics are the key to becoming a better programmer. An engineer with a deep understanding of algorithms and data structures will be able to make informed design choices, and write programs that are more performant and easier to change. I came, I saw, I conquered As a beginner, you may feel discouraged and frustrated by these concepts. The terms data structures and algorithms both sound abstract and obscure. Don't worry, though. That's a normal reaction for a beginner. I had the same reaction when I started learning about the subject. The reason you feel that way is that you haven't found the right method to let you study the subject deliberately. I'm going to tell you about my roadmap and tips. Be patient, and always keep moving forward. Data structures and algorithms do involve some mathematical reasoning and proofs, particularly when analyzing the time- and space-complexity of an algorithm. Being able to perform a big-O complexity analysis is certainly important, but you don't need to worry about it too much to start with. You don't need a high IQ or abstract mathematical knowledge. As long as you understand high school mathematics, you have the tools needed to understand data structures and algorithms. That may seem unlikely to you right now. In this article, I'll argue otherwise: after a few months of directed study and practice, you'll be able to approach the subject with confidence. What is data structure and algorithm essentially? Generally speaking, a data structure is a way to organize data, while an algorithm is a method or pattern for solving problems. To illustrate this, let's say you want to find a specific book in a library. Take an example How do you find a specific book from a library? Method #1: You could check each book by shelves, one by one, first to last, until you find the book in question. Method #2: You could first locate the bookshelf according to the category of the book in question. You determine whether the subject is the humanities, science, computer science and so on, and then you search only that specific bookshelf. Each of these methods are algorithms. This is the definition of an algorithm: a method for solving problems which can be implemented via programming. In this analogy, the books, the shelves, and the way they are arranged are all data structures. In concrete terms, when we speak about data structures, we are talking queues, stacks, heaps; when we say algorithms, we speak of binary search, dynamic programming, and so on. Together, data structures and algorithms represent tried and tested patterns for abstraction and problem-solving. They were invented by pioneers in the field. They did the hard work that we can leverage in order to solve many common development problems. How data structures and algorithms relate Data structures and algorithms complement each other. The data structure exists for the algorithm, and an algorithm generally suits a specific data structure. For instance, arrays are contiguous. The binary search algorithm applies to direct access of contiguous memory, so an array is used to store the data for a binary search algorithm. If instead we want to use a different data structure such as a linked list, the binary search algorithm wouldn't work, as the linked list data structure doesn't support direct access. The StepsPick a few good books and resources There are several classic works on data structure and algorithms. They are useful resources, but can be difficult for a beginner. At the outset, you're better off working with resources that use a programming language you already know. C: Algorithms is your choice Java: Data Structures and Abstractions with Java Python : Problem Solving with Algorithms and Data Structures Using Python Visualizations will help you understand how data structure and algorithms works. You can find awesome animations of this kind from the site: visualgo.net. Learning basic data structures Some data structures and algorithms, such as bipartite graphs, maximum streams etc, are less intuitive than others. All general algorithms are useful tools, but you rarely need to use these in day-to-day development. If you stick to learning the more intuitive and general algorithms first, you'll eventually be able to master more difficult, niche techniques. I recommend these eleven basic data structures to start: array, linked list, stack, queue, hash table, map, heap, binary tree, trie tree, graph, skip list Learn how they work, how they are implemented, their common APIs, and how they perform in terms of big-O complexity. Additionally, GeeksforGeeks has a bunch of good practices for using data structures. Learning basic algorithm design patterns From my experience, when studying algorithms, trying to memorize the steps and implementation details often isn't the best strategy. [bctt tweet="If you don't understand how it works, chances are you won't remember the particulars anyway; when it comes to algorithms, general understanding is what counts."] Don't try to remember every last detail about the implementation you read about. Instead, try to reinvent, reason and reimplement the algorithm by yourself. Here's my preferred workflow for learning algorithms. Repeatedly following it will give substantial results. Repeat this procedure will make you learn more effectively. Below basic algorithms are the ones most commonly used. By following the principle of deliberate practice, you should focus on one specific category of algorithms at a time, and try to understand them in order. sorting, binary search, search, string matching, recursion,hash algorithm, greedy algorithm, divide and conquer algorithm, backtracking algorithm,dynamic programming There are harder problems that need tweaking or require you to combine several algorithms for a viable solution. You should try them after mastering the basics. Learn by doing Several websites can help with deliberate practice of data structures and algorithms. I recommend these two, which both have excellent online judging systems: Leetcode: which details almost all common interview questions asked by the big companies (such as Facebook, Google, Amazon); a very helpful resource for interviewing. HackerRank: which has very clean categories for data structures and algorithms, and offers lessons in mathematics, database, and security. It also lets you enter programming contests for fun. Another good way to practice is to create trivial projects that use one specific data structure or algorithm. For example, this project uses the union-set data structure to create a maze, and also tries to implement pathfinding algorithms. Finishing small projects of this sort will build your confidence, and also teach you its use for creating real applications. Go deeperMore books Introduction to Algorithms, 3rd Edition is a great choice for in-depth study. The section on algorithmic reasoning is awesome, and it also offers more details on complexity analysis and mathematical tools. The other book I would recommend is The Art of Computer Programming. Few programmers have read it from cover to cover, but it remains perhaps the ultimate and authoritative in-depth reference on the subject. It covers almost all facets of programming, and its section on data structures and algorithms is second to none. Learn from real projects Knowledge and skill only matter when applied to tangible products. For instance, you may be curious about how Google's search suggestions work in terms of data structures or algorithms. Well-crafted open-source projects are incredibly valuable learning resources. Redis demonstrates the use of many commonly used data structures, and has many performance-related optimizations. The Linux Kernel relies heavily on data structures like linked list, red-black trees, hashes, etc., and fine-tunes its implementations in various ways. You are not expected to master these things instantly. But after years of practice, you will discover that data structures have become your abstraction tools of choice, and that fitting data structures to custom algorithms has become a comfortable means of solving problems. Be curious and keep practice! Computer Science Department at New York University Warren Weaver Hall, Room 305 251 Mercer Street, New York, NY 10012 Contact Us Lecture notes files. LEC # TOPICS SUPPORTING FILES Introduction and document distance L1 Introduction and document distance (PDF) Document distance (docdist{1,2,3,4}.py) L2 More document distance, mergesort (PDF) Document distance (docdist{5,6}.py) Binary search trees L3 Airplane scheduling, binary search trees (PDF - 1.4 MB) Binary search trees (including code) L4 Balanced binary search trees (PDF - 1.2 MB) See binary search trees for AVL code Hashing L5 Hashing I: chaining, hash functions (PDF) Document distance (docdist-dict.py) L6 Hashing II: table doubling, Karp-Rabin (PDF) L7 Hashing III: open addressing (PDF - 1.1 MB) Sorting L8 Sorting I: heaps (PDF - 1.0 MB) L9 Sorting II: heaps (PDF) L10 Sorting III: lower bounds, linear-time sorting (PDF) L11 Sorting IV: stable sorting, radix sort (PDF - 1.0 MB) Searching L12 Searching I: graph search, representations, and applications (PDF - 1.6 MB) Simple Python code for graphs (PY) L13 Searching II: breadth-first search and depth-first search (PDF - 1.3 MB) L14 Searching III: topological sort and NP-completeness (PDF) Shortest paths L15 Shortest paths I: intro (PDF - 1.0 MB) L16 Shortest paths II: Bellman-Ford (PDF - 1.1 MB) L17 Shortest paths III: Dijkstra (PDF) L18 Shortest paths IV: Dijkstra speedups (PDF - 1.2 MB) Dynamic programming L19 Dynamic programming I: memoization, Fibonacci, Crazy Eights, guessing (PDF) L20 Dynamic programming II: longest common subsequence, parent pointers (PDF) L21 Dynamic programming III: text justification, parenthesization, knapsack, pseudopolynomial time, Tetris training (PDF) L22 Dynamic programming IV: piano fingering, structural DP (trees), vertex cover, dominating set, and beyond (PDF) Numerics L23 Numerics I (PDF) Demos: square root of 2, chord length Source code (ZIP) (This zip file includes: 14 .js files, 2 .html files, 1 .css file, and 1 .project file.) L24 Numerics II (PDF) Beyond 6.006 L25 Beyond 6.006: follow-on classes, geometric folding algorithms (PDF) If you are interested in folding algorithms, you can look at the previous offering of 6.885 and the associated textbook. How can one become good at Data structures and Algorithms easily?Designed by FreepikLet us first clarify the question. There is not any easy way to become good at anything but there is an efficient way to do everything. Let us try to understand the difference between easy and efficient here with the help of a programming question! Consider the problem of "Searching an element in a sorted array". So, person A here solved the problem in an easy way yet Person B solved the problem in an efficient way. Now, the efficient way of learning Data Structures and Algorithms depends on a several factors: Your prior knowledge of programming languages and basic DS and Algos.The purpose for which you want to learn it.The resources available to you.A perfect guide!Let us now take a deeper look at each of the points highlighted above: Prior knowledge of DS and Algo: If you are already well versed with the basic data structures like Arrays, Linked Lists etc. and some of the basic algorithms like Sorting, Searching etc. then you will comparatively take much less time than a complete newbie as you already know the basics. For example if you even don't understand the programming example mentioned at the start of the article then you have a long way to go.Purpose of learning DS and Algo: It also depends on the purpose for which you want to improve your knowledge of Data Structures. Some people learn them for job interviews, some for competitive programming and some for gaining knowledge. If you are preparing for Job Interviews then you have a limited set of Data Structures to learn which are most commonly asked in the interviews, if you want to become a good competitive programmer then you will have to focus on complex data structures like Segment Trees, Fenwik Tree, Binary Indexed Trees etc.Resources Available: Resources play a most important role in learning anything. You need a set of good tutorials which are descriptive enough to clear all of the concepts from basics to advanced. You must also have a popular set of questions to practice the knowledge you have gained.A Guide: Let's just say you have figured out all of the above three points. You know the things you want to learn, you know the purpose for which you want to learn and you also have all of the resources and tutorials to do so. But you are still confused on a lot of things like "Where to Start?", "How to Start?" etc. So, you need someone to guide you through the process. That is there must be someone to help you use the resources available in an efficient way.Solution to the above problems: At GeeksforGeeks, we already have a vast collection of tutorials and problems based on Data Structures and Algorithms written by some good competitive programmers around the globe. We also have many past, running and upcoming courses on Data Structures and Algorithms which can be found on GeeksforGeeks-Courses. If you are preparing for Interviews, we recommend you to go through the following links: If you are an aspiring Competitive Programmer, visiting below links is a must: Attention reader! Don't stop learning now. Get hold of all the important DSA concepts with the DSA Self Paced Course at a student-friendly price and become industry ready. To complete your preparation from learning a language to DS Algo and many more, please refer Complete Interview Preparation Course.In case you wish to attend live classes with experts, please refer DSA Live Classes for Working Professionals and Competitive Programming Live for Students. Why Data Structures and Algorithms Are Important to Learn?Array, Linked List, Stack, Queues, Searching, Sorting, Tree, Graph… Do you have questions that why should I study all the above-complicated stuff if it has absolutely no use in real life?? Why companies ask questions related to data structures and algorithms if it's not useful in a daily job?? A lot of beginners and experienced programmers avoid learning Data Structures and Algorithms because it's complicated and they think that there is no use of all the above stuff in real life. So before we discuss the topic we are going to throw a simple problem at you and you need to find the solution for that. If you need to search your roll number in 20000 pages of PDF document (roll numbers are arranged in increasing order) how would you do that? If you will try to search it randomly or in a sequential manner it will take too much time. You might get frustrated after some time.You can try another solution which is given below… Go to page no. 10000If your roll no. is not there, but all other roll no. in that page are lesser than your thanGo to page no. 15000Still if your roll no. is not there. but this time all other roll no. is greater than your.Go to the page no. 12500Continue the same process and within 30-40 seconds you will find your roll number. Congratulations… you just have used Binary Search algorithm unintentionally.. This was just a simple example and you might have understood a little bit that why learning data structure and algorithm is important in real life. There are plenty of examples you can find in your daily life. So if you think that this skill is only important to crack the interviews of product-based companies then you are totally wrong. From the above example, we can straight forward give two reasons to Learn Data Structure and Algorithms… If you want to crack the interviews and get into the product based companiesIf you love to solve the real-world complex problems. To Crack the Interviews of the Top Product Based CompaniesDo you know that under the hood all your SQL and Linux commands are algorithms and data structures? You might not realize this, but that's how the software works. Data structures and algorithms play a major role in implementing software and in the hiring process as well. A lot of students and professionals have this question that why these companies' interviews are focused on DSA instead of language/frameworks/tools specific questions? Let us explain why it happens… When you ask someone to make a decision for something the good one will be able to tell you "I chose to do X because it's better than A, B in these ways. I could have gone with C, but I felt this was a better choice because of this". In our daily life, we always go with that person who can complete the task in a short amount of time with efficiency and using fewer resources. The same things happen with these companies. The problem faced by these companies is much harder and at a much larger scale. Software developers also have to make the right decisions when it comes to solving the problems of these companies. Knowledge of data structures like Hash Tables, Trees, Tries, Graphs, and various algorithms goes a long way in solving these problems efficiently and the interviewers are more interested in seeing how candidates use these tools to solve a problem. Just like a car mechanic needs the right tool to fix a car and make it run properly, a programmer needs the right tool (algorithm and data structure) to make the software run properly. So the interviewer wants to find a candidate who can apply the right set of tools to solve the given problem. . If you know the characteristics of one data structure in contrast to another you will be able to make the right decision in choosing the right data structure to solve a problem. Engineers working in Google, Microsoft, Facebook, Amazon-like such companies are different than others and paid higher as compared to other companies…but why? In these companies coding is just the implementation and roughly takes 20-30% of the time allotted to a project. Most of the time goes into designing things with the best and optimum algorithms to save on the company's resources (servers, computation power, etc). This is the main reason why interviews in these companies are focused on algorithms as they want people who can think out of the box to design algorithms that can save the company thousands of dollars. Youtube, Facebook, Twitter, Instagram, GoogleMaps all these sites have the highest number of users in the world. To handle more users on these sites it requires more optimation to be done and that's the reason product-based companies only hire candidates who can optimize their software as per user demand. Example: Suppose you are working in Facebook company. You come up with an optimal solution of a problem (like sorting a list of users from India) with a time complexity of O(nLogn) instead of O(n^2) and assume that n for the problem here for the company in real life scenario is 100 million (very fair assumption considering the number of users registered on facebook exceeds 1 billion). nLogn would be 800 million, while n^2 would be 10^7 billion. In cost terms, you can see that the efficiency has been improved more than 10^7 times, which could be a huge saving in terms of server cost and time. Now you might have got that companies want to hire a smart developer who can make the right decision and save company resources, time and money. So before you give the solution to use a Hash table instead of List to solve a specific problem think about the big scale and all the case scenarios carefully. It can generate revenue for the company or the company can lose a huge amount of money. Have you ever scolded by your parents when you were unable to find your book or clothes in your messed up room? Definitely yes…your parents are right when they give the advice to keep everything in the right place so the next time you can get your stuff easily. Here you need to arrange and keep everything (data) in such a structure that whenever you need to search something you get that easily and as soon as possible. This example gives a clear idea that how important it is to arrange or structure the data in real life. Now take the example of a library. If you need to find a book on Set Theory from a library, you will go to the maths section first, then the Set Theory section. If these books are not organized in this manner and just distributed randomly then it will be frustrating to find a specific book. So data structures refer to the way we organize information on our computer. Computer scientists process and look for the best way we can organize the data we have, so it can be better processed based on input provided. A lot of newbie programmers have this question that where we use all the stuff of data structure and algorithm in our daily life and how it's useful in solving the real-world complex problem. We need to mention that whether you are interested in getting into the top tech giant companies or not DSA still helps a lot in your day to day life. Don't you believe us…Let's consider some examples… Facebook (Yes… we are talking about your favorite application). Can you just imagine that your friends on facebook, friends of friends, mutual friends they all can be represented easily by Graph? Relax….sit for a couple of moments and think again…you can apply graph to represent friends connection on facebook.If you need to keep a deck of cards and arrange it properly how would you do that? You will throw it randomly or you will arrange the cards one over another and from a proper deck. You can use Stack here to make a proper arrangement of cards one over another.If you need to search a word in the dictionary, what would be your approach? Do you go page by page or you open some page and if the word is not found you open a page prior/later to one opened depending upon the order of word to the current page (Binary Search).The first two were a good example of choosing the right data structure for a real-world problem and the third one is a good example of choosing the right algorithm to solve a specific problem in less amount of time. All the above examples give you a clear understanding that how the organization of data is really important in our day to day life. Arranging data in a specific structure is really helpful in saving a lot of time and it becomes easier to manipulate or use them. The same goes for the algorithm…we all want to save our time, energy and resources. We all want to choose the best approach to solve the problems in our daily life. A lot of problems exist in the world that can take hours or days to be solved with the native solution, it also may take years ! can you imagine! watch this : Importance of Data Structure and Algorithms We are surrounded by a lot of real-world complex problems for which no one has the solution. Observe the problems in-depth and you can help this world giving the solution which no one has given before. Data structure and algorithms help in understanding the nature of the problem at a deeper level and thereby a better understanding of the world. If you want to know more about Why Data Structures and Algorithms then you must watch this video of Mr. Sandeep Jain (CEO & Founder, GeeksforGeeks).