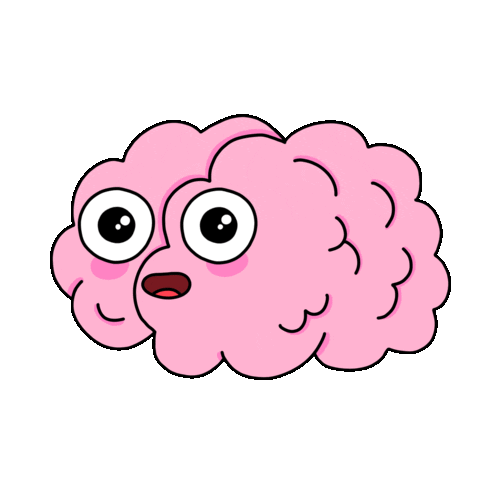
**Complete Roadmap to Learn Data Structure and Algorithms**

[#beginners](https://dev.to/t/beginners) [#algorithms](https://dev.to/t/algorithms) [#programming](https://dev.to/t/programming) [#productivity](https://dev.to/t/productivity)

The very 1st step you have to do is, **choose a programming language,** either **Java** or **C++** (or it's up to you which language you want to learn).

**🎯 Now, start learning the language 1st**

[](https://i.giphy.com/media/LpiVeIRgrqVsZJpM5H/giphy.gif)

Like…  
📌 Basic Syntax  
📌 Data Types  
📌 Operators, Variables, functions  
📌 Conditional Statement, loops  
📌 And the most important OOP(**Object-Oriented Programming**)

Learn about **Collection frameworks** if you chose java or STL(**Standard Template Library**) in C++.  
(Or you can skip this as of now).

Now, I can assume that you are a bit familiar with your preferred language.  
Like if I say every language has their own unique features/ operations which make them different from others right?

So, my point is that learn the language in such a manner that you don't suffer later.

Suppose, as an example, I am taking Java language here.  
So, in this you should know about java…

* Whether it is a OOP or procedural language?
* And if OOP, whether it is fully OOP or not?
* Why java does not support pointers?
* Why java is a platform independent language?
* Some features of it!

I will encourage you when you start leaning in any language to ask these type of question and try to explore more.

🎯 Now, you learned the language, got familiar with syntax and OPP concepts. So, It's time to deep dive into DSA(**Data Structure and Algorithm**).

You may or may not be familiar with this term *DSA* so, I am just briefing you here or in the alternative, you can search into google baba:

**Let me explain**

[](https://i.giphy.com/media/26gR0mfgaRrGlTsoE/giphy.gif)

**Data Structure and Algorithms play a vital role in solving problems (industrial problem, real scenario problem, etc…)**

Basically, DSA is a very crucial thing in a software field as even in real world scenario as well. In day to day life, you may or may not be notice these, or even we don't have any idea related to it…  
Like...

**Example 1:**

In your phone contacts, you must be search some contacts by their name, right?  
Here, have you thought that how fast your phone shows you the results?? *Probably in secs or even less, right!*

How it searches so fast even we have more than 200 or 500 contacts, but it gets back with results within secs  
How??

[](https://i.giphy.com/media/fUqfaPVjiAQcfticZH/giphy.gif)

You would be thinking about that, which algorithm it uses or which data structure it uses…??

So, before knowing the data structure, you should know about Time and Space complexity because of it you can analyze your algorithm that how fast and how efficient it is (**It's very important concept**)!

**What exactly is it, and why is it so important?**  
So, here Asymptotic Notation comes into picture...in this you will be learning about How can you make your algorithms more efficient or more optimized by reducing their time and space complexity.

Basically this concept is very, very important, or I can say it is a building block of your DSA.  
In this, you would learn about…

**Two type of Complexities:**

* Time
* Space

**Notations:**

* Big-O Notation (Ο) – Big O notation specifically describes worst case scenario.
* Omega Notation (Ω) – Omega(Ω) notation specifically describes best case scenario.
* Theta Notation (θ) – This notation represents the average complexity of an algorithm.

**Then learn about**

* O(1)
* O(log N)
* O(N log N)
* O(N)
* O(N^2)
* O(2^N)
* O(N!)

[You can follow this book for learning these concept](https://cin.ufpe.br/~fbma/Crack/Cracking%20the%20Coding%20Interview%20189%20Programming%20Questions%20and%20Solutions.pdf)

**Example : 2**

In your phone suppose you opened multiple websites suppose 1, 2, 3, 4 then you tapped in back what you see means in which order the websites appear…  
In this way, right!  
4 3 2 1 (LIFO) it is the concept of Stack(Which you will be studying in DS)

**Example : 3**

In day to day life most of you use Google Map right!  
Have you ever thought that how easily google find your specified location and get back to you with some results within secs?  
(**Basically the Graph concept involves here**)

Also involves lots of algorithms and logics behind on it.

Now you would have some idea that how important the DSA is!!

So, without any worry, let's get started. Learn the basic Data Structure.  
Like 1st know  
Types of DS  
It is of two Linear and Non-linear

**Linear DS**

* Array
* String
* Linked list
* Stack
* Queue
* Hash Table etc…

**Non-linear**

* Tree
* Graph
* Heap etc…

[You can learn these from here](https://practice.geeksforgeeks.org/batch/DSA-Workshop)  
Pick those DS one by one and start to learn those.  
Now, you can explore or learn Collection frameworks or STL.

🎯 In mean time pick any one of these [HackerRank](https://www.hackerrank.com/dashboard), [HackerEarth](https://www.hackerearth.com/), [Leetcode](https://leetcode.com/), [Codechef](https://www.codechef.com/) platform for solving the problems and test your understanding on those topics. I will suggest you that just complete one topic in DSA and try to solve some questions(**solve topic wise**). In this way, you will get more productive results to grasp the concept!

**Now learn some basics Algorithms…**

**Sorting Algorithms**

* Bubble Sort
* Selection Sort
* Insertion Sort
* Merge Sort
* Quick Sort
* Count Sort etc…

[👩‍🏫 Learn it here](https://www.hackerearth.com/practice/algorithms/sorting/bubble-sort/tutorial/)

**Searching Algorithms**

* Linear Search O(N)
* Binary Search O(log N)

[👩‍🏫 Learn it from here](https://www.hackerearth.com/practice/algorithms/sorting/bubble-sort/tutorial/)

**Recursion :**

(Grasp the concept very clearly)  
It is a technique in which the function call itself again and again until the base case occurs.

[👩‍🏫 Learn it from here](https://leetcode.com/explore/learn/card/recursion-i/)

**Backtracking :**

It is itself a recursion, but here it has some additional conditions which makes it more efficient.

Backtracking is an algorithmic-technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time (by time, here, is referred to the time elapsed till reaching any level of the search tree).

**Some standard problems**

* Sudoku Solver Problem
* Rat In Maze
* Combination sum, Combination Sum II, Combination Sum III
* N Queen Problem

[🐱‍🏍 You can visit here](https://github.com/SUCHITRAGIRI/DailyCodeWithJava/tree/main/Backtracking)  
[👩‍🏫 Check it here](https://leetcode.com/explore/learn/card/recursion-ii/)

**Greedy Approach :**

Greedy is an algorithmic paradigm that builds up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. So the problems where choosing locally optimal also leads to global solution are best fit for Greedy.  
For example, consider the Fractional Knapsack Problem.  
[👩‍🏫 Check it here](https://www.hackerearth.com/practice/algorithms/greedy/basics-of-greedy-algorithms/tutorial/)

**Dynamic Programming:**

DP = Recursion + some memory

It has two approach:

* Memoization (Top down 👇)
* Tabulation (Bottom Up ☝)

[You can follow these resources to learn the concept](https://www.hackerearth.com/practice/algorithms/dynamic-programming/introduction-to-dynamic-programming-1/tutorial/)

**Bit Manipulation / Bit Masking :**

* AND (&)
* OR (|)
* XOR (^)
* NOT (~)
* Left Shift (<<)
* Right Shift(>>)

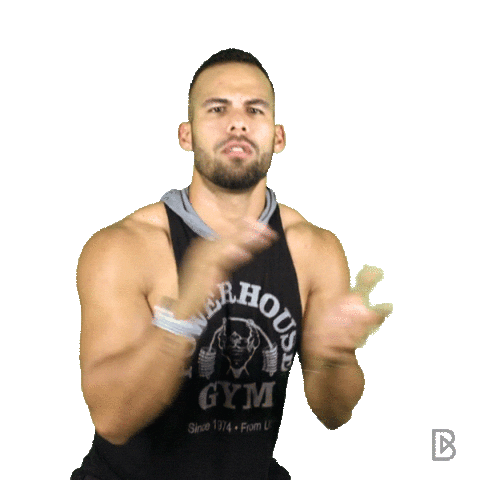
[📌 Learn from here!](https://leetcode.com/discuss/study-guide/1151183/bits-can-hack-the-world-beginners-guide-bit-hacks-with-proper-approach)

[👇 Here as well](https://codeforces.com/blog/entry/18169)

[📌 Another one 😀](https://practice.geeksforgeeks.org/tracks/dsa-workshop-1-bit-magic/?batchId=308)

**Some Advance Topics :**

* Segment Tree
* Tries
* Suffix Tree,
* Suffix Array,
* Advance Graph Theory

[](https://i.giphy.com/media/fwzgYx5zbOcaqGyYPA/giphy.gif)

Now, it's time to solve problems as much as you can:)  
(**Practice Makes Man perfect**)

Be consistent practicing, initially you will be facing difficulty to solve and even understand the logic by seeing the solution as well. These things are common, it happens to everyone. Just do and **don't give up**.  
(You will be master if you practice 😊)

Participate in contest if you want to test yourself or also interested in CP(**Competitive Programming**), give it try.

Otherwise, you are good enough for interview if you practice by yourself without giving contest as well!

**Some Tips :**

* While solving problems if you don't come up with solution with spending at most 1 hour, I will suggest you leave that question as for that time no need to spend more time on that!
* Try that question another day or some days later. If still stuck, then you can refer to some editorials or hints.
* One thing is very important is that suppose you successfully solved some questions 😀. Congratulations, but still I will suggest going to discussion and see other's solution as well. It will be more insightful, and you can get to know different approaches.
* And suppose you are in that situation that you can't solve the problem or even seeing the solution, you are unable to think that logic which are used in the solution. So, don't be panic and don't depress. Just dry run the solution, it really helps(**Personally it helps me a lot 😊**).
* If still you are thinking that the concept is not so much clear then try to write on paper what you understand the concept via dry running! (Also, you can refer to video lectures).

**Some Free Resources**

* [Algorithm Part I](https://www.coursera.org/learn/algorithms-part1)
* [Algorithm Part II](https://www.coursera.org/learn/algorithms-part2)
* [Durga Software Youtube Channel](https://www.youtube.com/c/DurgaSoftwareSolutions/playlists)
* [Pepcoding Youtube Channel](https://www.youtube.com/c/Pepcoding/playlists)
* [Aditya Verma Youtube Channel](https://www.youtube.com/c/AdityaVermaTheProgrammingLord/playlists)
* [Techdose Youtube Channel](https://www.youtube.com/c/TECHDOSE4u/playlists)
* [Tech You Forword](https://www.youtube.com/c/takeUforward/playlists)
* [Rachit Jain](https://www.youtube.com/c/RachitJain/playlists)
* [Geeks for Geeks](https://www.youtube.com/c/GeeksforGeeksVideos/playlists)
* [Code for Cause](https://www.youtube.com/c/CodeforCause/playlists)
* [Code with Harry](https://www.youtube.com/c/CodeWithHarry/playlists)
* [Apni kaksha](https://www.youtube.com/watch?v=lxja8wBwN0k&list=PLKKfKV1b9e8ps6dD3QA5KFfHdiWj9cB1s)
* [Simplilearn](https://www.youtube.com/c/SimplilearnOfficial/playlists)
* [Neso Academy](https://www.youtube.com/c/nesoacademy/playlists)