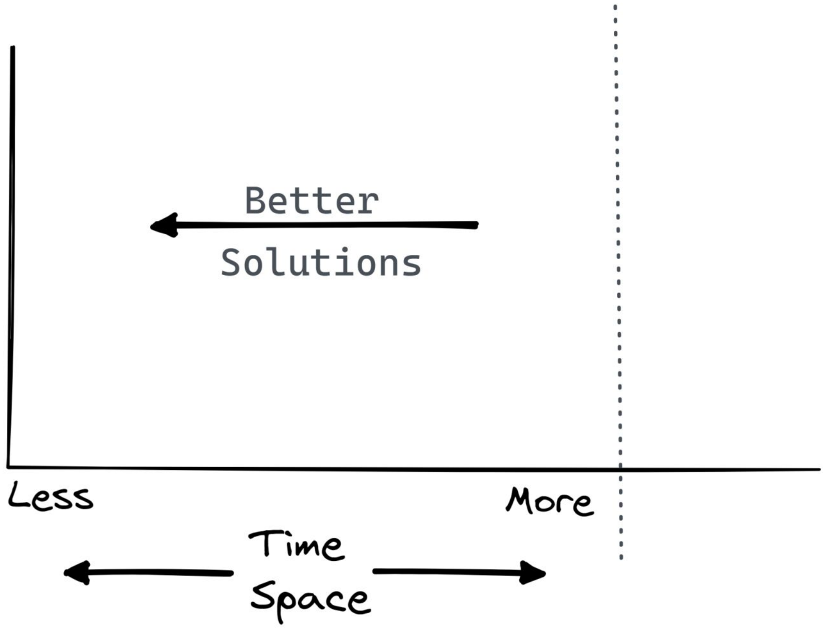
Time & Space Complexity

- Let's try to make Time & space complexity Easy to understand and more practical!.

- Time and Space Complexity problems are quite popular in coding interviews and can save a business a lot of money in things like bandwidth and storage.



Time Complexity

- Time complexity is the time taken by the algorithm to execute each set of instructions. It is always better to select the most efficient algorithm when a simple problem can solve with different methods.

Space complexity

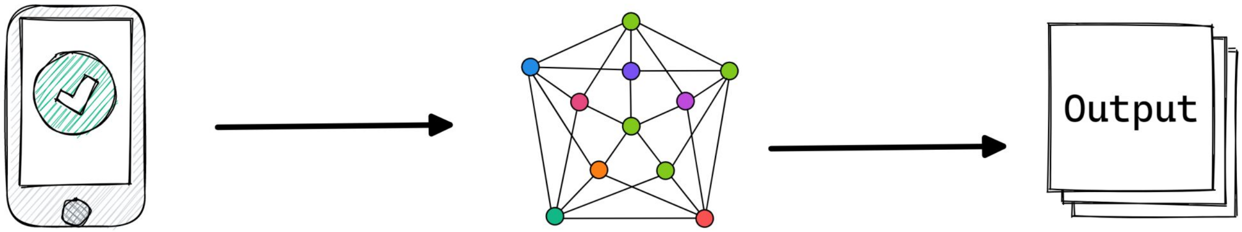
- Space complexity is usually referred to as the amount of memory consumed by the algorithm. It is composed of two different spaces, Auxiliary space and Input space.

Santosh Kumar Mishra

The factor of time is usually more important than that of space.

“47 percent of consumers expect a website to load in no more than two seconds.”

What it Takes to Create a Good Algorithm?



Ask for Input

Referencing to Algorithms

Provides you the output

→ A good algorithm is one that takes less time in execution and saves space during the process.

How Significant is Time Complexity?

→ Time complexity is profoundly related to the input size.
As the size of the input increases, the run time (which is – the time taken by the algorithm to run) also increases.

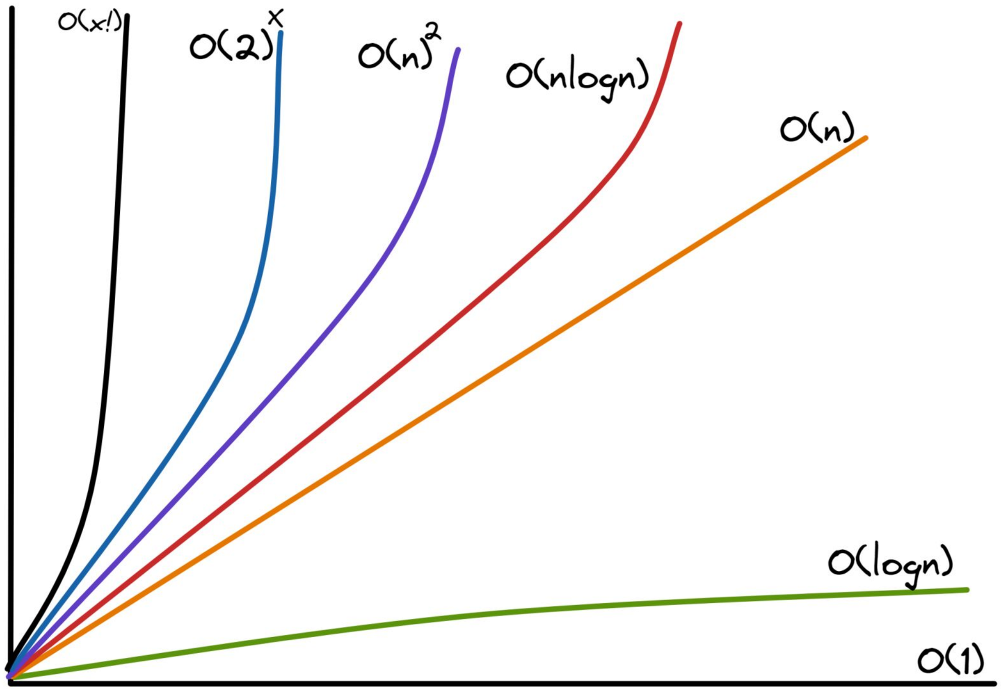
Introduction to Asymptotic Notations

→ Asymptotic Notations are the expressions that are used to represent the complexity of an algorithm.

Types of Data Structure Asymptotic Notation

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

→ "Big O notation is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity."



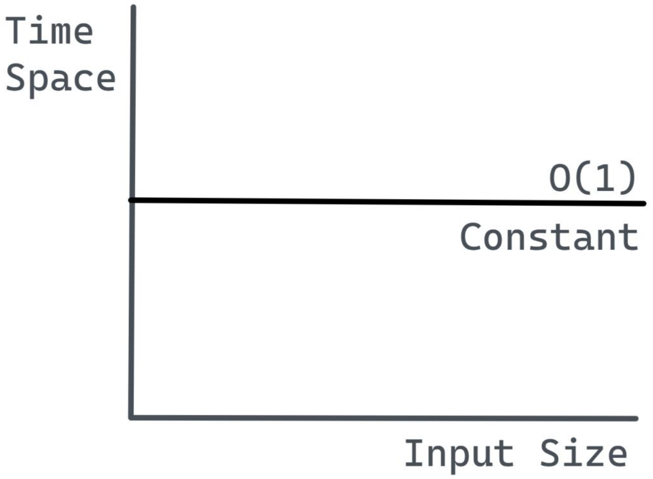
O(1) < O(logn) < O(n) < O(nlogn) < O(n^2) < O(n^3) < O(2^n) < O(n!)

Santosh Kumar Mishra

Common O Notations or Complexity Types:

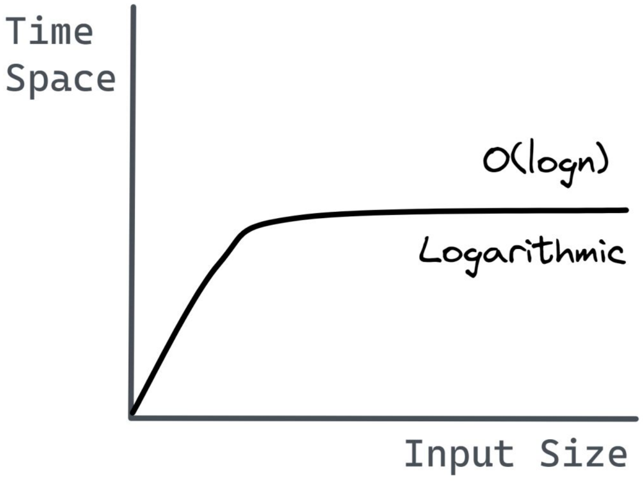
Constant Complexity O(1)

→ Big O notation O(1) represents the complexity of an algorithm that always execute in same time or space regardless of the input data.



Logarithmic Complexity O(logn)

→ Increases at first but then it stabilizes and changes less.



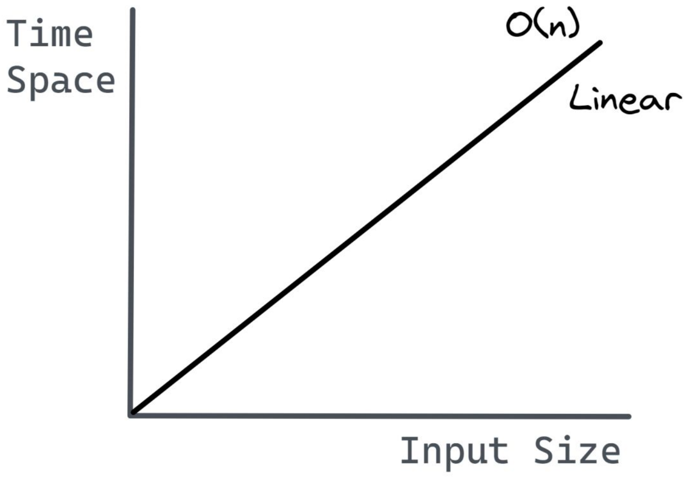
Exponential & Factorial Complexity

→ complexity/time rapidly increases until time and space requirements tend towards the infinite.

Santosh Kumar Mishra

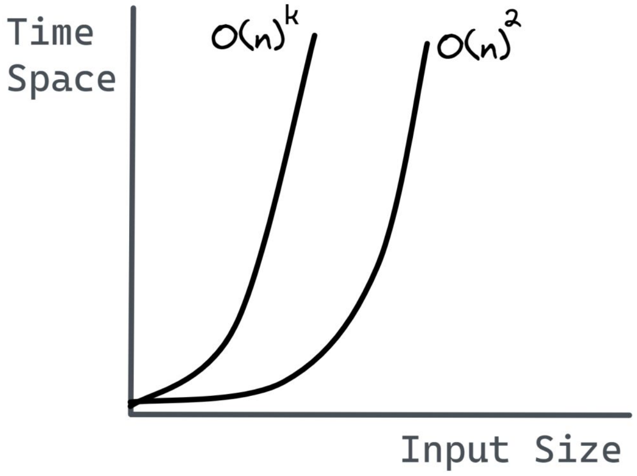
Linear Complexity O(n)

→ Big O notation O(N) represents the complexity of an algorithm, whose performance will grow linearly (in direct proportion) to the size of the input data.



Quadratic O(n^2) & Polynomial O(n^k)

→ In general as your inputs increase your, time becomes greater at a certain greater rate which is related to the exponent, ^2 for quadratics, ^K for Polynomials and everything in between:



Best case, Worst case, and Average case in Asymptotic Analysis:

Best Case:

→ It defines as the condition that allows an algorithm to complete the execution of statements in the minimum amount of time.

In this case, the execution time acts as a lower bound on the algorithm's time complexity.

Average Case:

→ In the average case, we get the sum of running times on every possible input combination and then take the average.

In this case, the execution time acts as both the lower bound and upper bound on the algorithm's time complexity.

Worst Case:

→ It defines as the condition that allows an algorithm to complete the execution of statements in the maximum amount of time.

In this case, the execution time acts as an upper bound on the algorithm's time complexity.

Get my Book "The Art of Data Structures and Algorithms" and Access all the most important DSA Problems and Solutions.

Download Interview Cafe Android App and Get this book

Get the Book from Amazon

Santosh Kumar Mishra

SDE @ Microsoft

Follow for more

iamsantoshmishra

iamsantoshmishra

Interview Cafe

Interview Cafe Notes