**What is a Data Structure ?**

A Data Structure is a way of organizing data so that it can be used effectively/efficiently.

**Why DS is important?**

They are essential ingredients in creating fast and powerful algorithms.

They help to manage and organize data.

They made code cleaner and easier to understand.

***Abstract Data Type***

An ADT (abstract data type) is an abstraction of a data structure which provides only the interface to which a DS must adhere to.

The interface does not give any specific details about how something should be implemented or in what programming language.

Example : the mode of type of transportation from point A to point B, there are so many ways that’s like there are abstract data types, the one we use is known as data structure.

For ex:-

|  |  |
| --- | --- |
| Abstraction (ADT) | Implementation(DS) |
| List | Dynamic Array,  Linked list |
| Queue | Linked List based Queue, Array based Queue,  Stack based queue |
| Map | Tree Map Hash Map / Hash Table |
| Vehicle | Bicycle, Smart car |

***Computational Complexity Analysis***

**How much time does this algorithm need to finish?**

**How much space does this algorithm need for its computation?**

# **Big – O Notation**

**Big-O Notation gives an upper bound of the complexity in the *worst* case helping to quantify performance as the input size becomes *arbitrarily large***.

n – The size of the input complexities ordered in from smallest to largest

Constant Time : O(1)

Logarithmic Time: O(log n)

Linear Time : O(n)

Linearithmic Time: O(n log(n))

Quadric Time : O(n^2)

Cubic Time: O(n^3)

Factorial Time : O(n!)

# Big – O properties

O(n + c) = O(n)

O( cn ) = O(n) , c>0

But this is theoretical, if c is 200M it will surely impact the n. 😊

Ex -> Let f be a function that describes the running time of a particular algorithm for an input of size n:

f(n) = 7 log(n)^3 + 15 (n)^2 + 2(n)^3 +3132

O(f(n)) = O(n^3)

**The following run in constant time : - O(1)**

a = b + 1;

a = v\*t + 2

i:=0 // this loop as well run in constant time   
while i<11 Do

i = i+1

**The following run in linear time : - O(n)**

|  |  |
| --- | --- |
| i:=0  while i<n Do  i = i+1 | i:=0  while i<n Do  i = i+3 |
| f(n) = n -> O(f(n)) = O(n) | f(n) = n/3 -> O(f(n)) = O(n) |

**The following run in Quadratic time : - O(n^2)**

n\*n = O(n^2)

1st loop for quadratic time

for( i:= 0 ; i<n; i=i+1)

for( j:= 0 ; j<n; j=j+1)

f(n) = n\*n = n^2, O(f(n)) = O(n^2)

2nd loop for quadratic time

for( i:= 0 ; i<n; i=i+1)

for( j:= i ; j<n; j=j+1)

f(n) = n\*n = n^2, O(f(n)) = O(n^2)

so why 2nd loop is also quadratic time algorithm

For a moment just focus on the second loop. Since i goes from [0,n) the amount of looping done is directly determined by what i is. Remark that if i=0 we do n work, if