**Data Structures**

It is the way to organize Data

After organizing data it become easy to process it.

**Type of Data Structures**

**Linear** – Array, LinkedList, Stack, Queue

**Non-Linear** – Tree, Graph

**Algorithms**

An Algorithm is a set of instruction to perform a task or to solve a given problem.

**Analysis of Algorithm**

To find best algorithm to solve an Problem

**Time Complexity** –

* Its amount time taken by Algorithm to Run.
* The process of input by an Algorithm which helps to determine the time Complexity.

**Space Complexity** –

* It’s amount of memory or space taken by an algorithm to run.
* The memory required to process the input by an algorithm which helps to determine the space Complexity.

**Asymptotic Analysis of an Algorithm**

* Asymptotic analysis helps in evaluating performance of an algorithm in term of input size and its increase.
* Using asymptotic analysis we don’t measure actual running time of algorithm.
* It help in determine how time and space taken by algorithm increases with input size.

**Asymptotic Notation**

This is the mathematical tools use to describe the running time of an algorithm in term of input size.

* Best Case
* Average Case
* Worst Case

**Type of Asymptotic Notation**

Omega

Big O

Theta

Omega

* It is the formal way to express the lower bound of an algorithm’s running time.
* Lower bound means for any given input this notation determines best amount of time an algorithm can take to complete.

Big O

* It is the formal way to express the Upper bound of an algorithm’s running time.
* Upper bound means for any given input this notation determines longest amount of time an algorithm can take to complete.

Theta

* It is the formal way to express for both the upper and lower bound of an algorithm’s running time.
* By both Upper and Lower bound means for any given input this notation determines Average amount of time an algorithm can take to complete.

**Rules for Big O**

* It’s a single Processor
* It perform sequential Execution of statement
* Assignment operation takes 1 unit of time
* Arithmetic operation takes 1 unit of time
* Logical operation takes 1 unit of time
* Return statement takes 1 unit of time
* Other small single operation takes 1 unit of time
* Drop lower order terms
* Drop Constant Multipliers

Calculating Time complexity of Constant Algorithm

public int sum(int x , int y){

int result = x + y;

return result;

}

|  |  |  |
| --- | --- | --- |
| Line no. | Operation | Unit time |
| 2 | 1+1+1+1 | 4 |
| 3 | 1+1 | 2 |

T = 4 + 2 = 6

O(1)

Calculating Time complexity of Linear Algorithm

public int findSume(int n){

int sum = 0;

for(int i = 0; i <= n; i++){

sum = sum + i;

}

return sum;

}

|  |  |  |
| --- | --- | --- |
| Line no. | Operation | Unit Time |
| 2 | 1 | 1 |
| 3 | 1 + 3n + 3 + 3n | 6n + 4 |
| 4 | n(1+1+1+1) | 4n |
| 6 | 1+1 | 2 |

T = 1 + 6n +4 + 4n + 2

T = 10n + 7

O(n)

Calculating Time complexity of Polynomial Algorithm

public void print(int n){

int sum = 0;

for(int i = 1; i <= n; i++){

for(int j = 1; j <= n; j++){

sop

}

sop()

}

sop()

}

|  |  |  |
| --- | --- | --- |
| Line no. | Operation | Unit Time |
| 2 | 1 + 3n + 3 + 3n | 6n + 4 |
| 3 | n(1 + 3n + 3 + 3n) | n(6n + 4) |
| 4 | n^2(1 + 1 + 1) | 3n^2 |
| 6 | n(1) | n |
| 8 | 1 | 1 |

T = 6n + 4 + 6n^2 + 4n + 3n^2 + n + 1

T = 9n^2 + 11n + 5

O(n^2)

**Singly Linked List**

It is an data structure used to storing collection of nodes

* It contain sequence of nodes.
* Node has data and reference to next node in List.
* First node is the Head Node.
* Last node has Data and point to Null.

**Doubly Linked List**

It is two way Linked List

* Given a Node, we can navigate list in both forward and backward direction, which is not possible in singly Linked List.
* A node in singly Linked list can only be detected if we have a pointer to its previous node. But in Doubly Linked List we can detect the node even if we don’t have pointer to its previous node.
* List node in Doubly Linked List :- Previous | Data | Next

**Circular Singly Linked List**

In this the Last node to pointing to first node.

Instead of head we keep track of last node in circular singly linked list.

**Stack**

* It is Linear data structure used for storing Data.
* It is an ordered list in which insertion and deletion are done at one end called as Top.
* It is LIFO(Last In First Out).

**Tree**

It is non-linear data structure use for storing data.

It is made up of Nodes and Edges without having any cycle.

Each node in a tree can point to n number of nodes in tree.

It is way to representing hierarchical stricter with a parent node called as Root and many levels of additional nodes.

**Binary Tree**

A tree having Zero, One or Two Children is Binary Tree.

Left | Data | Right

Left | Data | Right

Left | Data | Right

Left | Data | Right

Left | Data | Right

Left | Data | Right

Left | Data | Right

PreOrder 🡪 Node – Left – Right

InOrder 🡪 Left – Node – Right

PostOrder 🡪 Left – Right – Node