Data Structure

Trainer : Nisha Dingare

Email : nisha.dingare@sunbeaminfo.com

Introduction :

- Data structure is a form of Organising, Processing and Storing the data in the memory.
- There is a need of Data structures in programming to achieve efficiency in operations like addition, deletion, traversal, searching, sorting, etc.

- Linear/Basic Data Structures :
  1. Array
  2. Linked List
  3. Stack
  4. Queue
  5. Hash Tables

- Non-linear/Advanced Data Structures :
  1. Tree
  2. Graph

What is an Algorithm? :

- **What is a Program?**
  - A program is a set of instructions written in any programming language (like C, C++, Java, Python, Assembly etc...) given to the machine to do specific task.

- **What is an Algorithm?**
  - An algorithm is a finite set of instructions written in human understandable language (like english), if followed, accomplishes a given task. It is also called as pseudocode.

An algorithm is a template whereas a program is an implementation of an algorithm.

Analysis of an algorithm :

- Analysis of an algorithm is a work of determining how much time (computer time) and space (computer memory) it needs to run to completion.

- There are two measures of an analysis of an algorithms:

1. **Time Complexity:** It is the amount of time (computer time) required for an algorithm to run to completion.
2. **Space Complexity:** It is the amount of space (computer memory) required for an algorithm to run to completion.

Asymptotic Analysis :

- **Asymptotic analysis**: it is a mathematical way to calculate time complexity and space complexity of an algorithm without implementing it in any programming language. –
- In this kind of analysis, focus is on basic operation in that algorithm :
  - e.g. searching => basic operation is comparison, and hence analysis can be done depending on no. of comparisons taking place in different cases.
  - sorting => basic operation is comparison, and hence analysis can be done depending on no. of comparisons takes places in different cases.

- there are 3 asymptotic notations:
- 1**. Big Omega (Ω)** - this notation is used to represent best case time complexity. - big omega is referred as asymptotic lower bound.
- 2. **Big Oh (O)** - this notation is used to represent worst case time complexity - big oh is referred as asymptotic upper bound.
- 3. **Big Theta (θ)** - this notation is used to represent an average case time complexity. - big theta is referred as asymptotic tight bound

Searching Algorithms : Linear Search

- **Linear Search** : This is an algorithm to find the given number from the list of elements of an array, in a linear order, starting from the 0^{th} index.
- In this algorithm, key element gets compared sequentially with each array element by traversing it from first element till either match is found or maximum till the last element.

- **Time complexity** :
  - **Best Case**: If key is found at very first position in only 1 no. of comparison then it is considered as a best case and running time of an algorithm in this case is $O(1) = \Omega(1)$.
  - **Worst Case**: If either key is found at last position or key does not exists, maximum n no. of comparisons takes place, it is considered as a worst case and running time of an algorithm in this case is $O(n) = O(n)$.
  - **Average Case**: If key is found at any position in between, it is considered as an average case and running time of an algorithm in this case is $O(n/2) = \theta(n)$.

Searching Algorithms : Binary Search

- This algorithm follows divide-and-conquer approach.
- To apply binary search on an array prerequisite is that array elements must be in a sorted manner.
- In this algorithm, in first iteration, by means of calculating mid position big size array gets divided into two subarray's, **left subarray** and **right subarray**. Then key element gets compared with element which is at mid position. if key is found at mid position in very first iteration in only 1 no. of comparison, then it is considered as a best case and in this case an algortihm takes O(1) time.
- If key is not found in the first iteration then either it will get searched into left subarray or into the right subarray by applying same logic repeatedly till either key is not found or till max the size of any subarray is valid i.e. size of subarray is greater or equal to 1.
- **Time Complexity :**
  - **Best Case**: if the key is found in very first iteration at mid position in only 1 no. of comparison it is considered as a best case and running time of an algorithm in this case is $O(1) = \Omega(1)$.
  - **Worst Case**: if either key is not found or key is found at leaf position it is considered as a worst case and running time of an algorithm in this case is $O(\log n) = O(\log n)$.
  - **Average Case**: if key is not found in the first iteration and it is found at non-leaf position it is considered as an average case and running time of an algorithm in this case is $O(\log n) = \theta(\log n)$.

Thank You!