PRECAT: OM19

Data Structure

SunBeam Institute of Information & Technology, Hinjwadi, Pune & Karad.

SACHIN PAWAR

Data Structures: Introduction

 # PreCAT Scope : Data Structure

- In Section B, 7 Questions are reserved for this subject

and mostly all questions are pseudocode oriented (i.e. theory based/concepts).

- In this course the main focus is on **implementation of basic data structures** and **introduction to an advanced data structures** to build base required to learn and implement advanced data structures and algorithms in CDAC courses.

Data Structures: Introduction

+ **Introduction**
  - Data structure
  - Algorithm and analysis of an algorithm
+ **Array**
  - Concept & definition
  - **Searching Algorithms:**
    1. Linear Search (algorithm & implementation)
    2. Binary Search (algorithm & implementation)
  - **Sorting Algorithms:**
    1. Selection Sort (algorithm & implementation)
    2. Bubble Sort (algorithm & implementation)
    3. Insertion Sort (algorithm & implementation)
    4. Quick Sort (algorithm)
    5. Merge sort (algorithm)

Data Structures: Introduction

+ **Linked List**
  - Concept & definition
  - Types of Linked List
  - Operations on Linked List
  - Differnce between an array and linked list
+ **Stack**
  - Concept & definition
  - Implementation of stack data structure
  - Stack applications algorithms:
  1. Conversion of infix expression into its equivalent prefix
  2. Conversion of infix expression into its equivalent postfix
  3. Conversion of prefix expression into its equivalent postfix
  4. Postfix expression evalution

Data Structures: Introduction

+ Queue

- Concept & definition
- Types of queue
- Implementation of queue data structure
- Applications of queue

+ Introduction to an advanced data structure

- Tree
- Binary Heap
- Graph
- Hash Table

Data Structures: Introduction

Q. Why there is a need of data structure?

There is a need of data structure to achieve 3 things in programming:

1. Efficiency
2. Abstraction
3. Reusability

Q. What is Data Structure?

Data Structure is a way to store data elements into the memory (i.e. into the main memory) in an organized manner so that operations like addition, deletion, traversal, searching, sorting etc... can be performed on it efficiently.

Data Structures: Introduction

Two types of **Data Structures** are there:

1. Linear/Basic: data elements gets stored into the memory in a linear manner (e.g. sequentially) and hence can be accessed linearly/sequentially.

- Array
- Structure & Union
- Class
- Linked List
- Stack
- Queue

2. Non-linear/Advanced: data elements gets stored into the memory in a non-linear manner (e.g. hierarchical) and hence can be accessed non-linearly.

- Tree (Hierarchical)
- Graph
- Hash Table
- Binary Heap

Data Structures: Introduction

Array: It is a **basic/linear data structure** which is a collection/list of **logically related similar type of elements** in which data elements gets stored into the memory at **contiguos locations.**

Structure: It is a **basic/linear data structure** which is a collection/list of **logically related similar and disimmilar type of elements** gets stored into the memory **collectively (as a single entity/record).**

Sizeof of the structure = sum of size of all its members.

Union: Union is same like structure, except, memory allocation i.e. size of union is the size of max size member defined in it and that memory gets shared among all its members for effective memory utilization (can be used in a special case only).

Data Structures: Introduction

Q. What is a Program?

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

Q. What is an Algorithm?

- An algorithm is a finite set of instructions written in human understandable language (like english), if followed, acomplishesh a given task.

- An algorithm is a finite set of instructions written in human understandable language (like english) **with some programming constraints**, if followed, acomplishesh a given task, such an algorithm also called as **pseudocode.**

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

Data Structures: Introduction

Example: An algorithm to do sum of all array elements

```
Algorithm ArraySum(A, n)//whereas A is an array of size n
{
   sum=0;//initially sum is 0
   for( index = 1 ; index <= size ; index++ ) {
   sum += A[ index ];//add each array element into the sum
   }
   return sum;
}
```

- In this algorithm, **traversal/scanning** operation is applied on an array. Initially sum is 0, each array element gets added into to the sum by traversing array sequentially from the first element till last element and final result is returned as an output.

Data Structures: Introduction

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

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

1. Time Complexity of an algorithm is the amount of time i.e. computer time required for it to run to completion.

2. Space Complexity of an algorithm is the amount of space i.e. computer memory required for an algorithm to run to completion.

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 type of analysis, analysis can be done on the basis of **basic operation** in that algorithm.

e.g. in searching & sorting algorithms comparison is the basic operation and hence analysis gets done on the basis of no. of comparisons, in addition of matrices algorithms addition is the basic operation and hence on the basis of addition operation.

Data Structures: Introduction

"Best case time complexity": if an algo takes min amount of time to complete its execution then it is referred as best case time complexity.

"Worst case time complexity": if an algo takes max amount of time to complete its execution then it is referred as worst case time complexity.

"Average case time complexity": if an algo takes neither min nor max amount of time to complete its execution then it is referred as an average case time complexity.

"Asympotic Notations":

1. Big Omega (Ω): this notation is used to denote best case time complexity – also called as **asymptotic lower bound**

2. Big Oh (O): this notation is used to denote worst case time complexity - also called as **asymptotic upper bound**

3. Big Theta (θ): this notation is used to denote an average case time complexity - also called as **asymptotic tight bound**

Data Structures: Searching Algorithms

1. Linear Search/Sequential Search:

- 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.

```
Algorithm LinearSearch(A, size, key){
    for( int index = 1 ; index <= size ; index++ ){
    if( arr[ index ] ==  key )
    return true;
    }
    return false;
}
```

Data Structures: Searching Algorithms

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) => and hence time complexity = **Ω(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) => and hence time complexity = **O(n)**

Average Case: If key is found at any in between position it is considered as an average case and running time of an algorithm in this case is O(n/2) => and hence time complexity = **θ(n)**