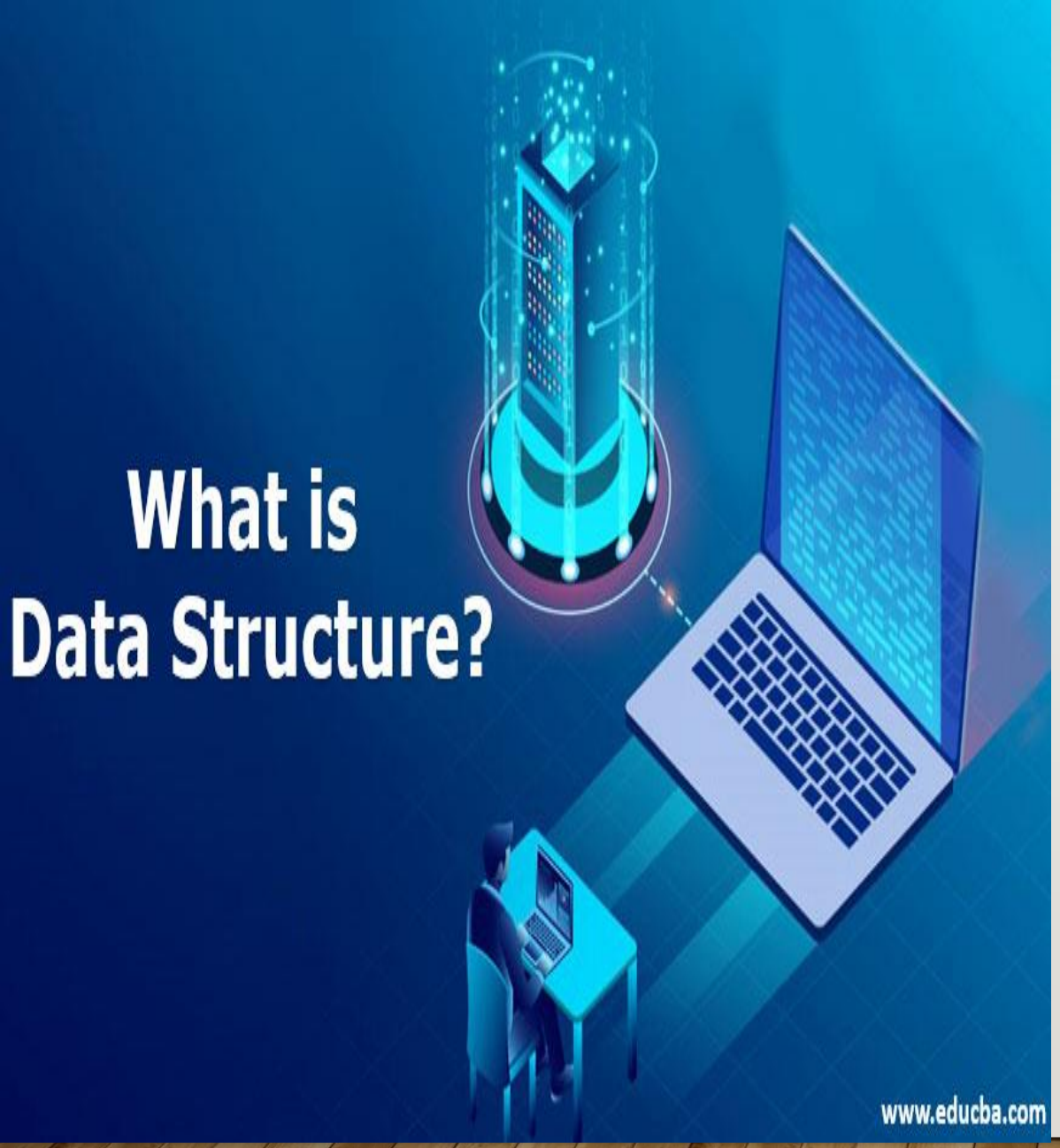
DATA STRUCTURES

LECTURE-1

INTRODUCTION

Dr. Sumitra Kisan



- A data structure is a specialized format for organizing, processing, retrieving and storing data.
- In computer programming, a data structure may be selected or designed to store data for the purpose of working on it with various algorithms.
- Data Structure can be defined as the group of data elements which provides an efficient way of storing and organizing data in the computer so that it can be used efficiently.

➢It plays a vital role in enhancing the performance of a software or a program as the main function of the software is to store and retrieve the user's data as fast as possible

➢ Data Structures are widely used in almost every aspect of Computer Science i.e. Operating System, Compiler Design, Artificial intelligence, Graphics and many more

➢ Data Structure mainly specifies the following four things:

1)organization of data 2)accessing method 3)degree of associativity 4) processing alternative for information

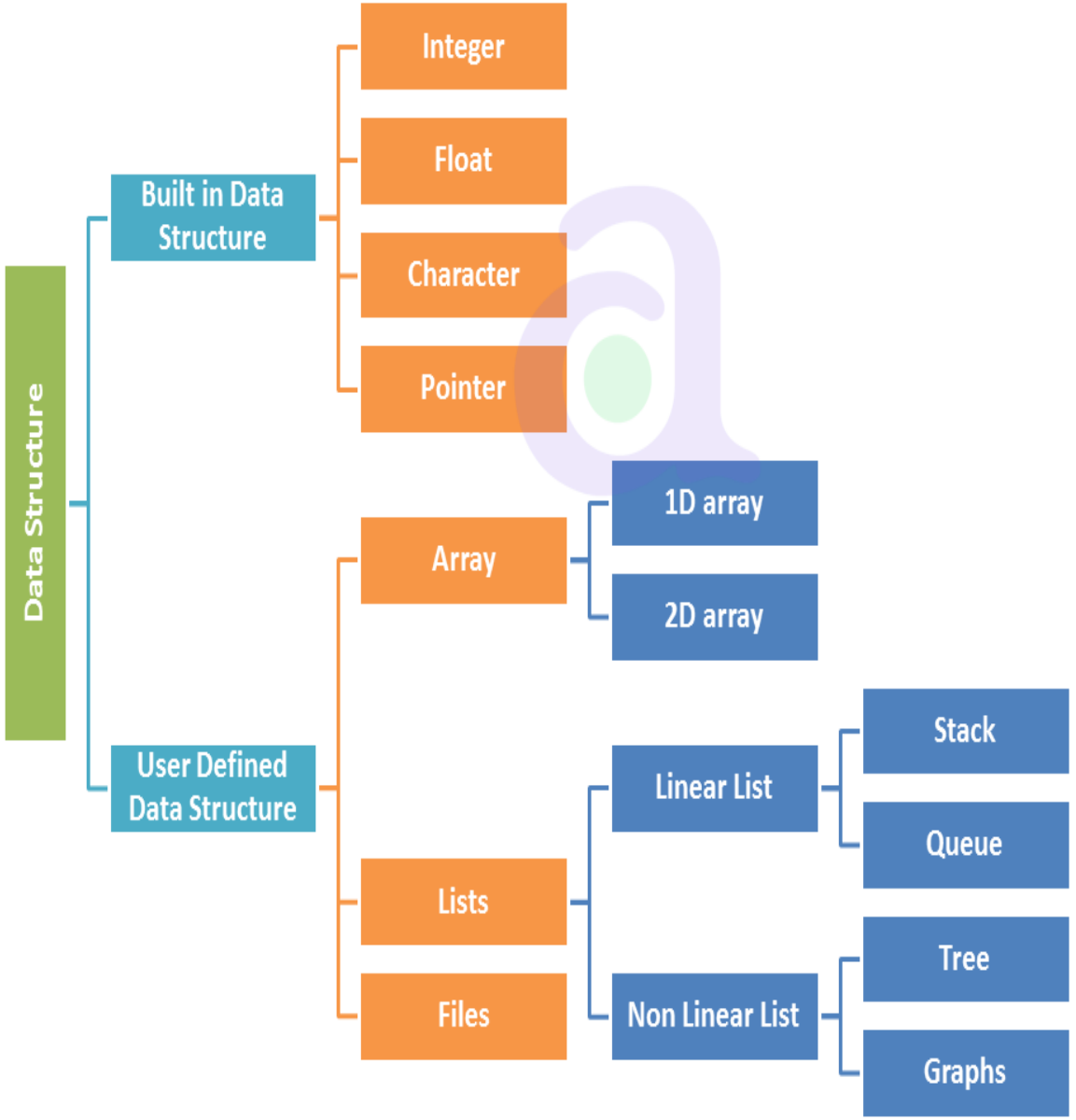
➢ **Data Structure study Covers the following points**

1) Amount of memory require to store
2) Amount of time require to process
3) Representation of data in memory
4) Operations performs on data

Types Of DS

The DS are divided into two types:
1. Primitive
2. Non primitive

Non primitive divided into two type
1. Linear DS
2. Non linear DS



Primitive Data Structure

- Primitive Data Structure are basic structure and directly operated upon by machine instructions.
- Primitive data structures have different representations on different computers.
- Integers, floats, character and pointers are example of primitive data structures.
- These data types are available in most programming languages as built in type.

Integer: It is a data type which allows all values without fraction part. We can used it for whole numbers.
Float: It is a data type which is use for storing fraction numbers.
Character: It is a data type which is used for character values.
Pointer: A variable that hold memory address of another variable are called pointer.

Non Primitive Data Type

- These are more sophisticated data structures.
- These are derived from primitive data structure.
- The non – primitive data structures emphasize structuring of a group of homogeneous or heterogeneous data items.

Example of non – primitive data types are Array, List, and File etc.

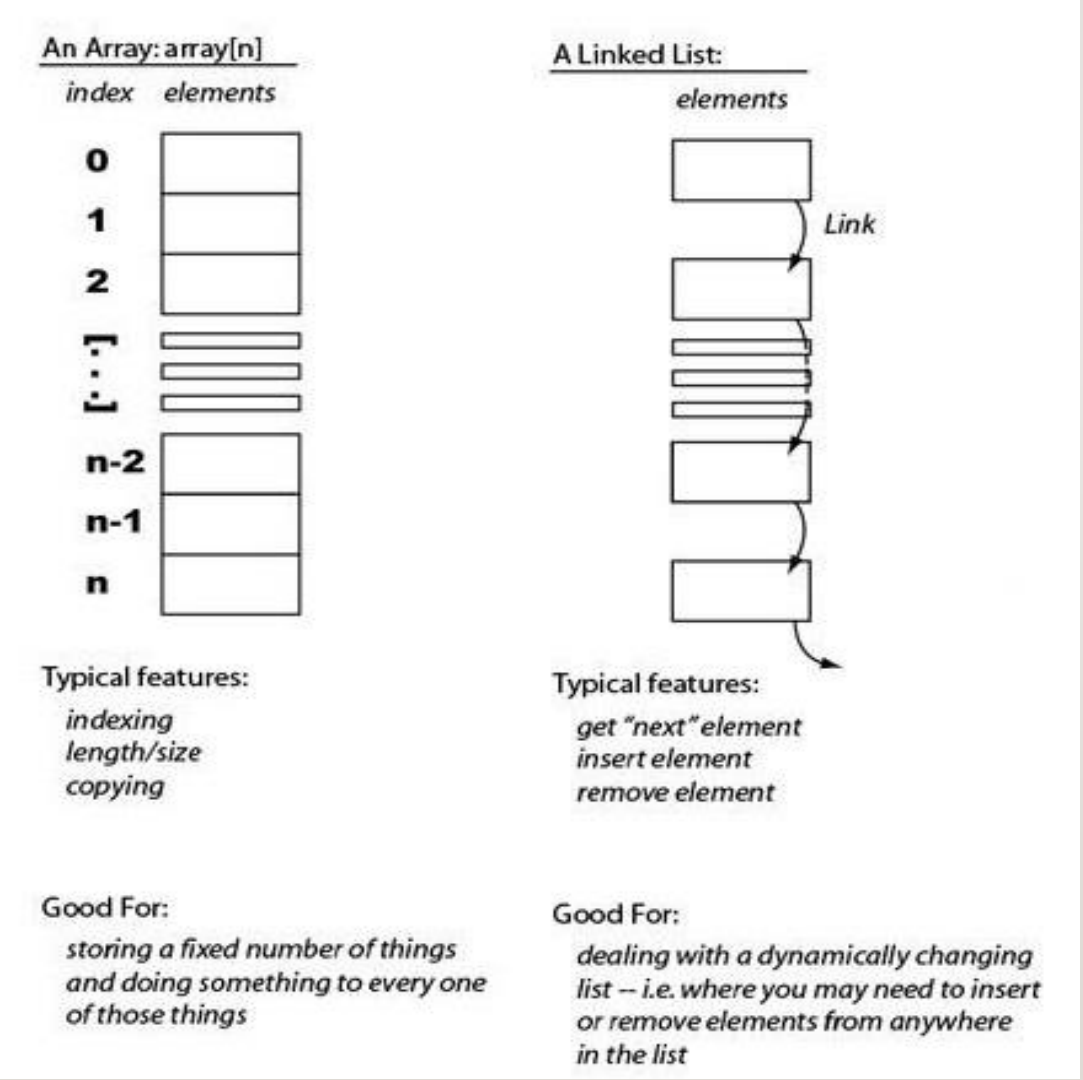
A non – primitive data type is further divided into Linear and non – Linear data structure.

Array: An array is a fixed size sequenced collection of elements of the same data type.

List: An ordered set containing variable number of elements is called as List.

File: A file is a collection of logically related information. It can be viewed as a large list of records consisting of various fields.

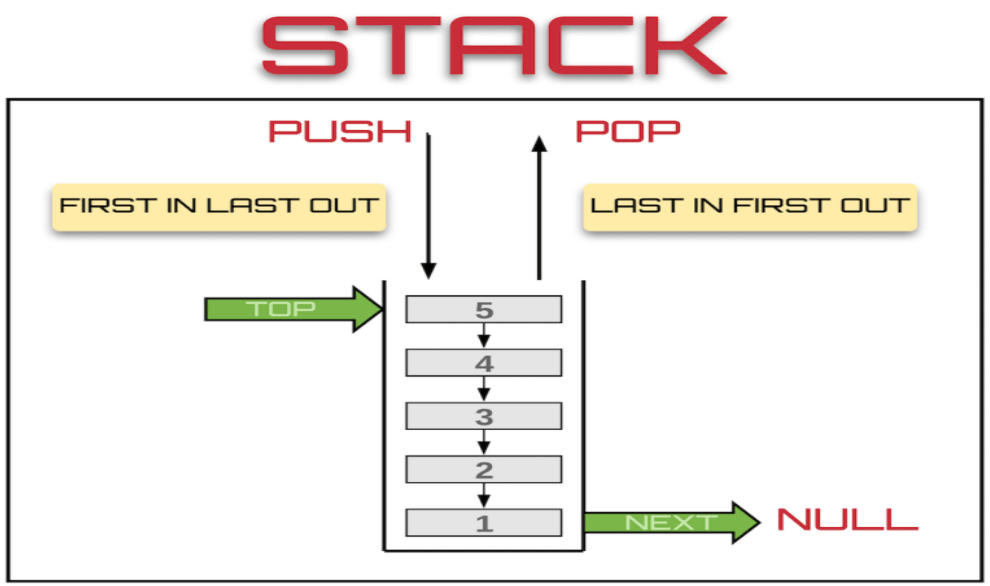
Linear Data Structures

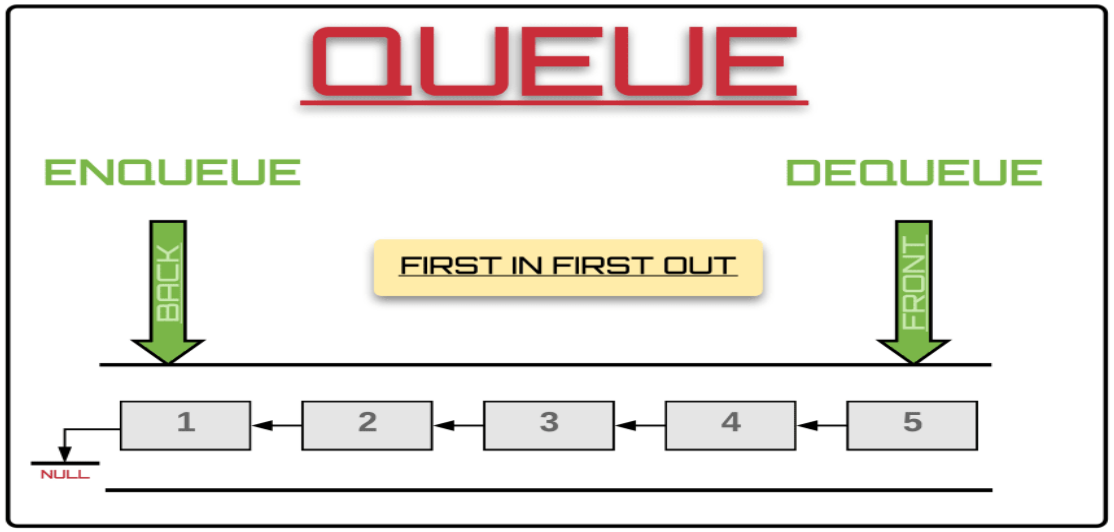


- A linear data structure simply means that it is a storage format of the data in the memory in which the data are arranged in contiguous blocks of memory.
- Example is the array of characters it represented by one character after another.
- In the linear data structure, member elements form a sequence in the storage.
- There are two ways to represent a linear data structure in memory.
  - ❖ **Static memory allocation**
  - ❖ **Dynamic memory allocation**

The possible operations on the linear data structure are:

1) Traversing 2) Insertion 3) Deletion 4) searching
5) sorting 6) merging





Example of Linear data structure are Stack and Queue

Stack

- Stack is a data structure in which insertion and deletion operations are performed at one end only.
- The insertion operation is referred to as ‘PUSH’ and deletion is referred as ‘POP’ operation
- Stack is also called as Last In First Out (LIFO) data structure.

Queue

- The data structure which permits the insertion at one and deletion at another end, known as Queue.
- End at which deletion is occurs is known as FRONT end and another end at which insertion occurs is known as REAR end.
- Queue is also called as First In First Out (FIFO)

Non-Linear Data Structure

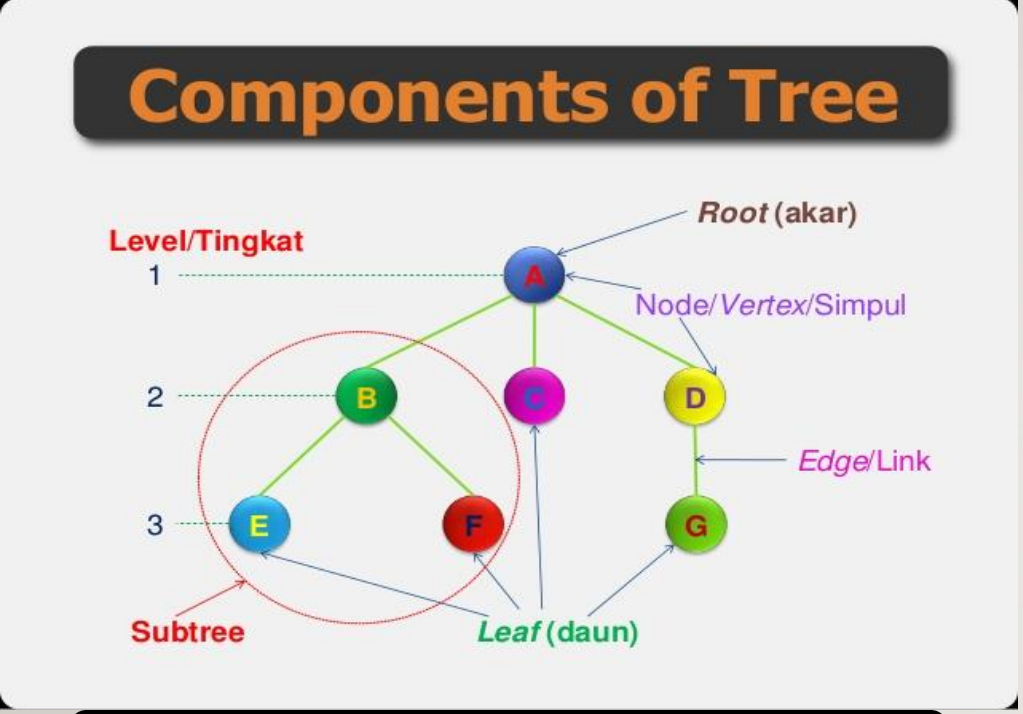
- Non linear DS are those data structure in which data items are not arranged in a sequence.
- Example on Non Linear DS are Tree and Graph.

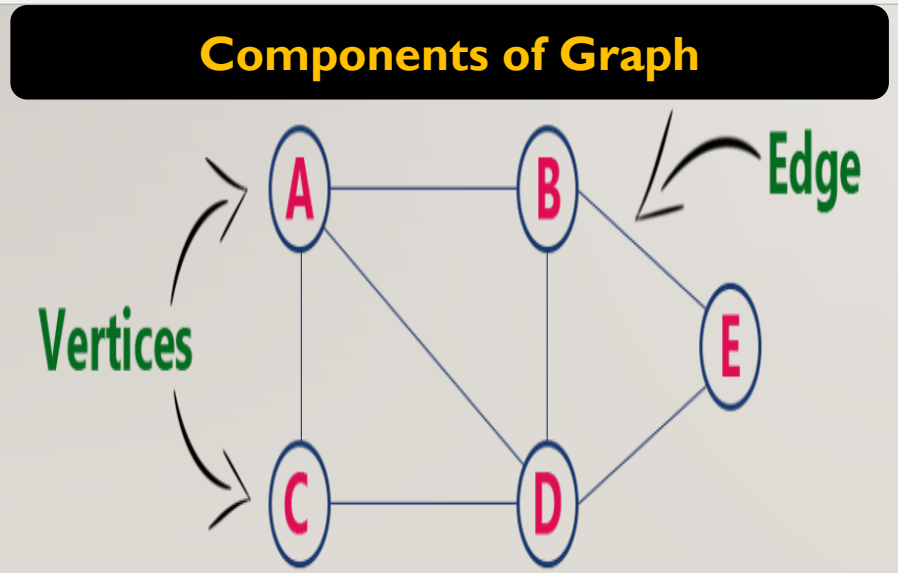
TREE

- A Tree can be define as finite data items (nodes) in which data items are arranged in branches and sub branches
- Tree represent the hierarchical relationship between various elements
- Tree consist of nodes connected by edge.

Graph

- Graph is collection of nodes (information) and connecting edges (Logical relation) between nodes.
- A tree can be viewed as restricted graph
- Graph have many types: 1) Simple graph 2) Mixed graph 3) Multi graph 4) Directed graph 5) Un-directed graph





DIFFERENCE BETWEEN LINEAR AND NON LINEAR DATA STRUCTURE

Linear Data Structure

- Every item is related to its previous and next item.
- Data is arranged in linear sequence.
- Data items can be traversed in a single run
- E.g. Array, Stacks, Linked list, Queue
- Implementation is easy.

Non – Linear Data Structure

- Every item is attached with many other items.
- Data is not arranged in sequence.
- Data cannot be traversed in a single run.
- E.g. Tree, Graph
- Implementation is difficult.

Operation on Data Structures

Design of efficient data structure must take operations to be performed on the DS into account. The most commonly used operations on DS are broadly categorized into following types

1. **Create:** This operation results in reserving memory for program elements. This can be done by declaration statement Creation of DS may take place either during compile-time or run-time.
2. **Destroy:** This operation destroy memory space allocated for specified data structure .
3. **Selection:** This operation deals with accessing a particular data within a data structure.
4. **Updating:** It updates or modifies the data in the data structure.
5. **Searching:** It finds the presence of desired data item in the list of data items, it may also find locations of all elements that satisfy certain conditions.
6. **Sorting:** This is a process of arranging all data items in a DS in particular order, for example either ascending order or in descending order.
7. **Splitting:** It is a process of partitioning single list to multiple list.
8. **Merging:** It is a process of combining data items of two different sorted list into single sorted list.
9. **Traversing:** It is a process of visiting each and every node of a list in systematic manner.

Algorithm

Definition: - An algorithm is a Step By Step process to solve a problem, where each step indicates an intermediate task. Algorithm contains finite number of steps that leads to the solution of the problem.

Properties /Characteristics of an Algorithm:-

Algorithm has the following basic properties

- **Input-Output**:- Algorithm takes ‘0’ or more input and produces the required output. This is the basic characteristic of an algorithm.
- **Finiteness**:- An algorithm must terminate in countable number of steps.
- **Definiteness**:- Each step of an algorithm must be stated clearly and unambiguously.
- **Effectiveness**:- Each and every step in an algorithm must be simple and basic so that any person can carry out these steps easily and effectively using a pen and paper.
- **Generality**:- Algorithm is generalized one. It works on all set of inputs and provides the required output. In other words it is not restricted to a single input value.

Categories of Algorithm

Based on the different types of steps in an Algorithm, it can be divided into three categories:-

- Sequence
- Selection
- Iteration

Sequence: The steps described in an algorithm are performed successively one by one without skipping any step. The sequence of steps defined in an algorithm should be simple and easy to understand. Each instruction of such an algorithm is executed, because no selection procedure or conditional branching exists in a sequence algorithm.

Example: // adding two numbers

Step 1: start
Step 2: read a,b
Step 3: Sum=a+b
Step 4: write Sum
Step 5: stop

Selection:

The sequence type of algorithms are not sufficient to solve the problems, which involves decision and conditions. In order to solve the problem which involve decision making or option selection, we go for Selection type of algorithm. The general format of Selection type of statement is as shown below:

if(condition)
Statement-1;
else
Statement-2;

The above syntax specifies that if the condition is true, statement-1 will be executed otherwise statement-2 will be executed. In case the operation is unsuccessful, then sequence of algorithm should be changed/ corrected in such a way that the system will re-execute until the operation is successful.

Example1:

```
// Person eligibility for vote
Step 1 : start
Step 2 : read age
Step 3 :  if age > = 18 then step_4 else step_5
Step 4 : write "person is eligible for vote"
Step 5 : write " person is not eligible for vote"
Step 6 : stop
```

Iteration:

Iteration type algorithms are used in solving the problems which involves repetition of statement. In this type of algorithms, a particular number of statements are repeated ‘n’ no. of times.

Example1:

Step 1 : start
Step 2 : read n
Step 3 : repeat step 4 until n>0
Step 4 : (a) r=n mod 10
(b) s=s+r
(c) n=n/10
Step 5 : write s
Step 6 : stop

1. Write an algorithm for roots of a Quadratic Equation?
2. Write an algorithm to find the largest among three different numbers entered by user
3. Write an algorithm to find the factorial of a number entered by user.
4. Write an algorithm to find the Simple Interest for given Time and Rate of Interest .