**What is Data Structures?**

A data structure is a particular way of organizing data in a computer so that it can be used effectively. For example, we can store a list of items having the same data-type using the *array* data structure.

The main idea of using data structure is to reduce the space and time complexities of different tasks.

**Application of Data Structure:-**

* For representing a city region telephone network.
* To implement back functionality in the internet web browser.
* To store dynamically growing data which is accessed very frequently, based upon a key value.
* To implement the undo function in a text editor.
* To store information about the directories and files in a system.

**Various operations that can be performed on different Data Structures**

* Insertion
* Deletion
* Traversal
* Searching
* Sorting

**Categories of DS**

* Linear Data Structure
  + Array
  + Linked List
  + Stack
  + Queue
* Non-Linear Data Structure
  + Tree
  + Graph

**Array:-**

* Array is a data structure used to store homogeneous elements at contiguous memory locations.
* Element stored in it can be of any valid data type like char, int, float or double.
* Size of an array must be provided before storing data.
* **Example:** For example, let us say, we want to store marks of all students in a class, we can use an array to store them. This helps in reducing the use of number of variables as we don’t need to create a separate variable for marks of every subject. All marks can be accessed by simply traversing the array.

**Types of Array:-**

* One Dimensional Array (One Subscripts)
* Two Dimensional Array (Two Subscripts)

**Linked List:-**

* A Linked List is a data structure which stores data in a separate object. This object is usually called Node.
* A Node consists of two parts, one is data part and another one is reference part.
* Data part is used to store the required data and reference part is used to point to next Node.

**Types of Linked List:-**

* **Singly Linked List:-**
  + This is a type of linked list in which Node consists of two part, one is data part used to store data and another part is reference part which stores the address of next node. Last node’s address part contains value Null which represents the end of List.
  + Example:-
* **Doubly Linked List:-**
  + In this type of Linked List, There are two reference part and one data part, Data part is used to store data and one of the reference part contain the address of next node and another one consists the address of previous node.
  + Advantage of doubly Linked list over singly linked list is that from any given node we can traverse in both the directions.

**Stack:-**

Stack is data structure which process elements stored in it in LAST IN FIRST OUT manner.

Various operations that we can perform on a stack is

* Push (Used to push element into stack)
* Pop (Used to pop last entered element from stack)
* Top (Used to find top element in the stack)
* Empty (used to find whether stack is empty or not)

**Application of stack:-**

Stacks are used for maintaining function calls in recursion.

Checking for parenthesis whether it is balanced or not (in case of editor)

Expression evaluation

**Queue:-**

Queue is a linear data structure which process data stored in it in FIRST IN FIRST OUT manner.

Elements are entered from rear end of queue and removed from front end.

Various operations that can be performed on a queue is

* Enqueue (used to push element in the queue)
* Dequeue (used to pop first entered element from queue)

**Applications of Queue:-**

* Queues are used as waiting lists for a single shared resource such as printer, disk, CPU.
* Queue is used as buffers in most of the applications like MP3 media player, CD player etc. It is also used to maintain playlist in media players.
* Queue are used in handling interrupts in operating systems.

**Tree:-**

Trees are hierarchical data structures.

In which data is stored using Nodes and Each Node has its own parent Node (except Root Node) as well as child Nodes (except leaf Nodes).

If we talk about Binary tree, In that each node can have at most two children which is called is left child and right child. It is implement using Nodes which consists of one data part and two reference part. Data part is used to store data, one reference part is used to store address of left child, and another reference is used to store address of right child.

**Graph:-**

Graph is data structure which consists of two components:

* A finite set of vertices also called as nodes.
* A finite set of ordered pair of the form (u, v) which denotes an edge from node u to node v.

Adjacency Matrix and Adjacency list is used to represent graph.

Graph can be classified into two categories:-

* Undirected Graph: - All edges are bidirectional.
* Directed Graph: - All edges are unidirectional.

**Difference between Array and Linked List:-**

* In Array, Elements can be accessed directly as it supports Random Access while Linked List don’t support Random access time requires for accessing any element is O (n).
* In Array, Elements are stored in contiguous memory locations, it is not compulsory in Linked list. Elements can be stored anywhere in the memory.
* In Array, We have static memory allocation but in linked list we have dynamic memory allocation.
* In Array, Insertion and deletion operation will take O (n) time while in case of linked list if we have pointer node ahead of which insertion is to be done then Insertion operation can be done in constant time.

REMAINING

* BFS and DFS of (Graph) and Tree.
* Memory efficient Linked List
* Implement stack using queue……
* Implement queue using stack……
* LRU cache
* How to check if a given binary tree is BST or not.
* Program on 1. Inserting element in Linked List 2. Deleting element from Linked list 3. Middle of a given linked list 4. Nth Node from the end of LL.
* Program on 1. Inorder, preorder traversal, postorder Traversals, level order traversal, Height of Binary tree.
* Program on converting a Doubly linked list to Binary tree in-place and vice versa also
* Reverse a linked list.
* Detect a loop in a linked list.
* Which data structure is used for dictionary and spell checker?
* Trie Data structure
* What are the drawbacks of array implementation of queue?
* Circular queue.
* What is dequeue (double-ended queue).
* Maximum number of nodes in a binary tree of height k?
* AVL Tree and its properties
* BST Tree and its properties
* B-Tree and its properties
* Application of Graph Data Structure.
* In Which Scenario, Binary Search can be used?
* Difference between Null and void? (Null is a value, void is a data type)
* Heap data structure.
* How searching is more efficient in BST as compared to Binary Tree.
* Can we apply Binary Search algorithm to a sorted Linked List.
* When can you tell that a memory leak will occur?
* Sorting a stack using temporary stack
* Program to reverse a queue.
* Check if given graph is tree or not.
* Kth smallest element is an unsorted array.
* How to find shortest path between two vertices.
* Memory areas like heap and stack.