TREE

=> What is tree in datastructure ?

A tree is a hierarchical datastructure that is used to represent and organize data in a way that is easy to navigate and search. It is a collection of nodes that are connected by edges and has a hierarchial relationship between the nodes.

-> Root node & Child nodes

The topmost node of the tree is called Root and the nodes below it are called Child nodes.

=> Why Tree is considered as a non-linear data structure?

In a tree data structure it is not arranged in a sequential manner or in a linear manner. We can say that it follows a hierarchial manner. So it is called non-linear data structure.

=> Basic Terminologies in Tree

1) Parent node : The node which is a predecessor of another node is called parent node.

2) Child node : The node which is a immediate successor of a node is called Child node

3) Leaf node or External node : The nodes which do not have any child nodes are Leaf node.

4) Ancestor : Any predessor nodes on the path of the root to that node are called Ancestors of that node.

5) Descendant : An successor node on the path from the leaf node to that node.

6) Sibling: Children of the same Parent node are Called Siblings. are Called Siblings.

7) Level of a node: The Count of edges on the Path from the root node to that node. The root node has level 0.

8) Neighbour: The Parent or Child nodes of that node are Called neighbours.

9) Subtree: Any node of the tree along with its descendant.

=> Properties of a tree

1) Number of edges: An edge Can be defined as the Connection between two nodes. If a tree has N nodes it has N-1 edges.

2) Depth of a node: The depth of a node is defined as the length of Path from the root to that node.

3) Height of a node: The height of a node Can be defined as the length of the longest Path from the node to the leaf node of the tree.

4) Height of the tree: The height of a tree is the length of the longest Path from the root of the tree to a leaf node of the tree

5) Degree of a Node: The total Count of Subtree attached to that node is Called the degree of the node. The degree of the leaf node must be 0. The degree of a tree is the maximum degree of a node among all the nodes in the tree.

Q why to use tree data Structure ?

-> If we want to store information in the form of hierarchy we use trees

-> Trees Provide moderate access / search (quicker than linked list and slower than Arrays)

-> Trees Provide Moderate insertion / deletion (quicker than Arrays and slower than Unordered Linked lists)

-> Like Linked list and Unlike Arrays, Trees don't have an upper limit on the number of nodes as nodes are ~~like~~ linked using Pointers.

=> ~~Stem~~ Syntax

```
Struct Node
{
 int data;
 Struct Node *left_child;
 Struct node *right_child;
};
```

=> Basic Operation of Tree

-> Create
-> Insert
-> Search
-> Pre Order Traversal -> Travelling a tree in a Pre-order manner

-> In Order Traversal -> Travelling a tree in a In Order Manner

-> Post Order Traversal -> Travelling a tree in a Post order Manner.

=> Types of Tree datastructure.

1) General Tree

In a general tree data structure has no restriction on the number of nodes. It means that a Parent node can have any number of child nodes.

2) Binary Tree

A node of a binary tree can have of maximum two child nodes.

3) Balanced Tree

If the height of the left Subtree is equal to the right Subtree then it is called balanced ~~binary~~ tree.

4) Binary Search Tree

Binary Search tree is used for various Searching and Sorting algorithms The examples include AVL tree and red-black tree. The value of the left most node is less than Parent node and value of the right most node is greater than it's Parent.

=> Applications of Tree data Structure.

1) Spanning trees.
2) Binary Search tree
3) Storing hiearchial data
4) Syntax tree
5) Trie
6) Heap
7) Artificial intelligence
8) Database
9) Network.

=> Advantages of Tree data Structure

-> Efficient insertion, Deletion and Search operation
-> Trees are flexible
-> Used to represent hierarchial relationship
-> It has ability to represent recursive structure
-> Use less space than other data structure
-> Trees are dynamic in nature
-> Trees can automatically self-organize

=> Disadvantage of Tree datastructure

-> Trees use additional m/y for pointers
-> Trees are not best choice for data does not have hierarchial relationship
-> Limited scalability compared to other data structures like hashtables.
-> Not suitable for large datasets
-> Trees can become unbalanced, leading to poor performance and decreased efficiency

=> Trie Data Structure

A trie also known as a prefix tree or a digital tree, is a specialized tree based data structure used for efficient retrieval and storage of strings. The word "trie" comes from the word "retrieval"

In a trie, each node represents a character or a part of a string. The structure of a trie allows for fast searching, insertion, and deletion of strings based on their prefixes

=> Properties related to Trie

1) The root node of the Trie always represents the null node.
2) Each Child of nodes is sorted alphabetically
3) Each node can have a maximum of 26 Children
4) Each node can store one letter of the alphabet.

-> Basic Operations

1) Insertion
2) Searching a node
3) Deletion of a node

=> Applications of Trie

1) Spell Checker
2) Auto Complete
3) Browser history

-> Advantages of Trie

1) It can be insert faster and search the string than hash tables and binary search trees
2) It provides an alphabetical filter of entries by the key of the node.

-> Disadvantages of Trie

1) It requires more memory to store the strings
2) It is slower than the hash table

=> Binary Search Tree

A binary search tree follows some order to arrange the elements. In a Binary search tree, the value of left node must be smaller than the parent node and value of the right node mus be greater than the parent node. This rule is applied recursively to the left and right subtrees of the root.

=> Complexities of Binary Search tree

1) Time Complexity

-> Insertion

Best Case = O(log n)
Average Case = O(log n)
Worst Case = O(n)

-> Deletion

Best Case = O(log n)
Average Case = O(log n)
Worst Case = O(n)

-> Search

Best Case = O(log n)
Average Case = O(log n)
Worst Case = O(n)

2) Space Complexity

-> Insertion O(n)
-> Deletion O(n)
-> Search O(n)

=> Advantages and disadvantages of Binary Search tree

-> Advantages

1) Efficient Search
2) Ordered data
3) Quick insertion and Deletion
4) Dynamic Data Structure

-> Disadvantages

1) Unbalanced Trees
2) Memory overhead
3) Inefficient for Sorted data
4) Lack of Direct Access

=> Applications of Binary Search Tree

1) Dictionary
2) Database Systems
3) Range Queries
4) Auto-Complete and Predictive Text
5) Binary Search
6) Huffman Coding
7) File Systems.

=> Heap Data Structure

A heap datastructure that represents a nearly Complete binary tree. It is Commonly used to efficiently manage and Organize Priority queues. The heap Data Structure has two variations.

-> Min heap
-> Max heap

min heap

For any given node, the value of that node is less than or equal to the values of its child nodes. It means that the smallest element will be the root node.

Max heap

In max heap the value of the given node is greater than or equal to the values of the child nodes. It means that the largest value will be the root node.

-> Primary Operations in heap

1) Insertion
2) Extraction

=> Advantages & disadvantages of heap

-> Advantages

1) Efficient insertion and Extraction
2) Partially ordered structure
3) Space efficiency

-> Disadvantages

1) No efficient search operation
2) Limited Flexibility
3) Lack of structural ordering
4) Inefficient for maintaining sorted order

=> Applications of heap

1) Priority Queues
2) Graph Algorithms
3) Heapsort
4) Event Scheduling