Trees
Session 1

Binary Tree Data Structure

A tree whose elements have at most 2 children is called a binary tree.

Binary Tree Representation :

A tree is represented by a pointer to the topmost node in a tree. If the tree is empty, then the value of root is NULL.

Tree node contains the following parts.

1. Data
2. Pointer to left child
3. Pointer to right child

We can represent a tree node using structures.

```
struct node
{
        int data;
        struct node* left;
        struct node* right;
};
```

Let us create a simple tree

```
#include <bits/stdc++.h>
using namespace std;

struct Node {
    int data;
    struct Node* left;
    struct Node* right;

    Node(int x)
    {
        data = x;
        left = NULL;
        right = NULL;
    }
};

int main()
{

    struct Node* root = new Node(1);

    root->left = new Node(2);
    root->right = new Node(3);

    root->left->left = new Node(4);

    return 0;
}

```

Important Properties:

1) The maximum number of nodes at level 'i' of a binary tree is 2^i.

2) The maximum number of nodes in a binary tree of height 'h' is $2^h - 1$.

3) The minimum number of nodes in a binary tree of height 'h' is $h + 1$.

Maximum number of nodes in a binary tree of height h is 1 + 2 + 4 + ... This is a simple geometric series with h terms and the sum of this series is $2^h - 1$.

4) In a Binary Tree with N nodes, minimum possible height or the **minimum number of levels is $Log2(N+1)$.**

5)In a Binary Tree with N nodes, maximum possible height or the **maximum number of levels is $N-1$.**

6) A Binary Tree with L leaves has at least $|Log2L| + 1$ levels

A Binary tree has the maximum number of leaves (and a minimum number of levels) when all levels are fully filled. Let all leaves be at level i, then below is true for the number of leaves L.

$L \leq 2^{i-1}$
$i = |Log2L| + 1$

where i is the minimum number of levels.

Types

Full Binary Tree A Binary Tree is a full binary tree if every node has 0 or 2 children. We can also say a full binary tree is a binary tree in which all nodes except leaf nodes have two children.

Complete Binary Tree : A Binary Tree is a Complete Binary Tree if all the levels are completely filled except possibly the last level and the last level has all keys as left as possible. Sometimes this is known as almost complete binary tree,

Perfect Binary Tree A Binary tree is a Perfect Binary Tree in which all the internal nodes have two children and all leaf nodes are at the same level.

In a Perfect Binary Tree, the number of leaf nodes is the number of internal nodes plus 1

$L = I + 1$

Where

L = Number of leaf nodes,

I = Number of internal nodes.

A degenerate (or pathological) tree A Tree where every internal node has one child. Such trees are performance-wise the same as linked list.

```
 10
   /
  20
  \
   30
   \
    40
```

Binary Tree (Array implementation)

We are going to talk about the sequential representation of the trees. To represent a tree using an array, the numbering of nodes can start either from 0 to (n-1) or 1 to n.

```
#include<bits/stdc++.h>
using namespace std;
char tree[10];
int root(char key)
{
        if(tree[0] != '\0')
                cout << "Tree already had root";
        else
                tree[0] = key;
        return 0;
}
int set_left(char key, int parent)
{
        if(tree[parent] == '\0')
                cout << "\nCan't set child at"
                        << (parent * 2) + 1
                        << " , no parent found";
        else00
                tree[(parent * 2) + 1] = key;
        return 0;
}
int set_right(char key, int parent)
{
        if(tree[parent] == '\0')
```

```
                cout << "\nCan't set child at"
                        << (parent * 2) + 2
                        << " , no parent found";
        else
                tree[(parent * 2) + 2] = key;
        return 0;
}
int print_tree()
{
        cout << "\n";
        for(int i = 0; i < 10; i++)
        {
                if(tree[i] != '\0')
                        cout << tree[i];
                else
                        cout << "-";
        }
        return 0;
}
int main()
{
        root('A');
        //insert_left('B',0);
        set_right('C', 0);
        set_left('D', 1);
        set_right('E', 1);
        set_right('F', 2);
        print_tree();
        return 0;
}
```

Traversals:

What are BFS and DFS for Binary Tree?

A Tree is typically traversed in two ways:

- Breadth First Traversal (Level Order Traversal)
- Depth First Traversals
    - Inorder Traversal (Left-Root-Right)
    - Preorder Traversal (Root-Left-Right)
    - Postorder Traversal (Left-Right-Root)

Extra Space required for Level Order Traversal is O(w) where w is the maximum width of Binary Tree. In level order traversal, queue one by one stores nodes of different levels.

Extra Space required for Depth First Traversals is O(h) where h is the maximum height of the Binary Tree. In Depth First Traversals, stack (or function call stack) stores all ancestors of a node.

Applications :

1. Store hierarchical data, like folder structure, organization structure, XML/HTML data.
2. Binary Search Tree is a tree that allows fast search, insert, delete on a sorted data. It also allows finding closest item
3. Heap is a tree data structure which is implemented using arrays and used to implement priority queues.
4. B-Tree and B+ Tree : They are used to implement indexing in databases.
5. Syntax Tree: Used in Compilers.
6. K-D Tree: A space partitioning tree used to organize points in K dimensional space.
7. Trie : Used to implement dictionaries with prefix lookup.
8. Suffix Tree : For quick pattern searching in a fixed text.
9. Spanning Trees and shortest path trees are used in routers and bridges respectively in computer networks
10. As a workflow for compositing digital images for visual effects.

1. height-of-binary-tree
2. count-leaves-in-binary-tree
3. size-of-binary-tree
4. sum-of-binary-tree
5. count-non-leaf-nodes-in-tree