**Trees and   
 Binary Trees**

Presented By

Suma.J

Poornima.R

**TREES**

* **A tree is a Multilevel data structure that represent a hierarchical relationship between the Set of individual elementTrees are one of the important non- Linear data structure. is called nodes.**
* **Each tree structure starts with a node Which is called the root node of the Tree.**

**Tree Terminology**

* **Root: node without parent (A)**
* **Siblings: nodes share the same parent**
* **Internal node: node with at least one child (A, B, C, F)**
* **External node (leaf ): node without children (E, I, J, K, G, H, D)**
* **Ancestors of a node: parent, grandparent, grand-grandparent, etc.**
* **Descendant of a node: child, grandchild, grand-grandchild, etc.**
* **Depth of a node: number of ancestors**
* **Height of a tree: maximum depth of any node (3)**
* **Degree of a node: the number of its children**
* **Degree of a tree: the maximum number of its node.**
* **Subtree: tree consisting of a node and its descendants**

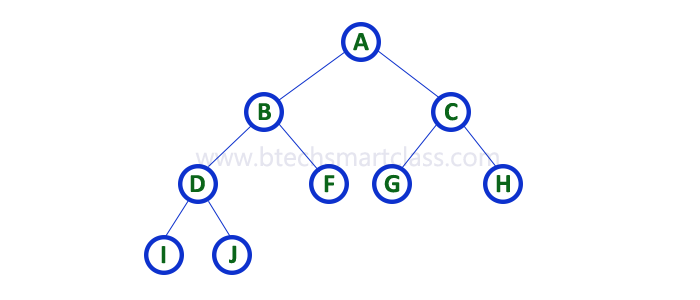
**Binary Tree**

* **A tree in which every node can have a maximum of two children is called as Binary Tree**

****

**Full Binary Tree**

* **A binary tree in which every node has either two or zero number of children is called Full Binary Tree**

****

**Complete Binary Tree**

* **A binary tree in which every internal node has exactly two children and all leaf nodes are at same level is called Complete Binary Tree. Complete binary tree is also called as Perfect Binary Tree**
* **Number of nodes = 2d+1 – 1**
* **Number of leaf nodes = 2d**
* **Where, d – Depth of the tree**

**Binary Tree Representation**

**1. Sequential representation using arrays**

**2. List representation using Linked list**

**List representation**

* **Advantages:** 
  + **Height of tree need not be known**
  + **No memory wastage**
  + **Insertion and deletion of a node is done without affecting other nodes**
* **Disadvantages:** 
  + **Direct access to node is difficult**
  + **Additional memory required for storing address of left and right node**

**ALGORITHM:**

**Algorithm minimum element using binary tree**

**//Input :A[0 to n-1] elements in ascending order**

**//output:minimum element in a binary tree**

**If root=null**

**Return-1**

**While current ->left !=Null**

**Current= current ->left**

**Return current->data**

**Algorithm maximum element using binary tree**

**//Input :A[0 to n-1] elements in ascending order**

**//output:maximum element in a binary tree**

**If root=null**

**Return-1**

**current=root**

**While current->right !=Null**

**current= current ->right**

**Return current->data**

**Find Minimum and Maximum Element in an Binary Tree**

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

// A tree node

struct Node

{

int data;

struct Node\* left, \*right;

};

// A utility function to create a new node

struct Node\* newNode(int data)

{

struct Node\* node = (struct Node\*)

malloc(sizeof(struct Node));

node->data = data;

node->left = node->right = NULL;

return(node);

}

// Returns maximum value in a given Binary Tree

int findMax(struct Node\* root)

{

// Base case

if (root == NULL)

return INT\_MIN;

// Return maximum of 3 values:

// 1) Root's data 2) Max in Left Subtree

// 3) Max in right subtree

int res = root->data;

int lres = findMax(root->left);

int rres = findMax(root->right);

if (lres > res)

res = lres;

if (rres > res)

res = rres;

return res;

}

// Returns minimum value in a given Binary Tree

int findMin(struct Node\* root)

{

// Base case

if (root == NULL)

return INT\_MAX;

// Return minimum of 3 values:

// 1) Root's data 2) Max in Left Subtree

// 3) Max in right subtree

int res = root->data;

int lres = findMin(root->left);

int rres = findMin(root->right);

if (lres < res)

res = lres;

if (rres < res)

res = rres;

return res;

}

// Driver program

int main(void)

{

struct Node\*NewRoot=NULL;

struct Node \*root = newNode(2);

root->left = newNode(7);

root->right = newNode(5);

root->left->right = newNode(6);

root->left->right->left=newNode(1);

root->left->right->right=newNode(11);

root->right->right=newNode(9);

root->right->right->left=newNode(4);

printf("Maximum element is %d \n", findMax(root));

printf("Minimum element is %d \n", findMin(root));

return 0;

}

**Output**:

