2, May, 2022

TREES

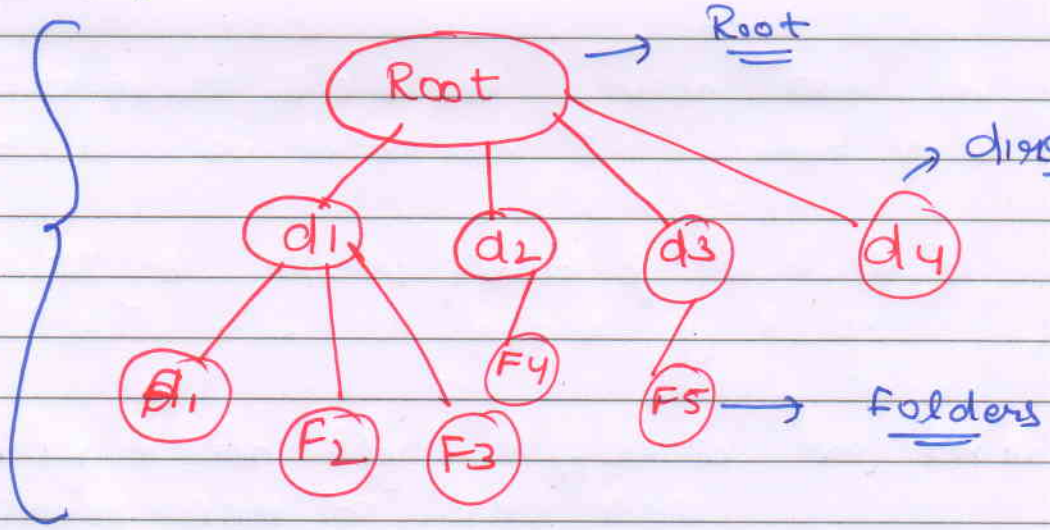
Generic TREE → [CN

INTRODUCTION

A tree is a data structure in which one node may be connected to any number of nodes

Example →

In Linux, we have root directory. Root directory contain some or more directories, and further they can contain more folders.

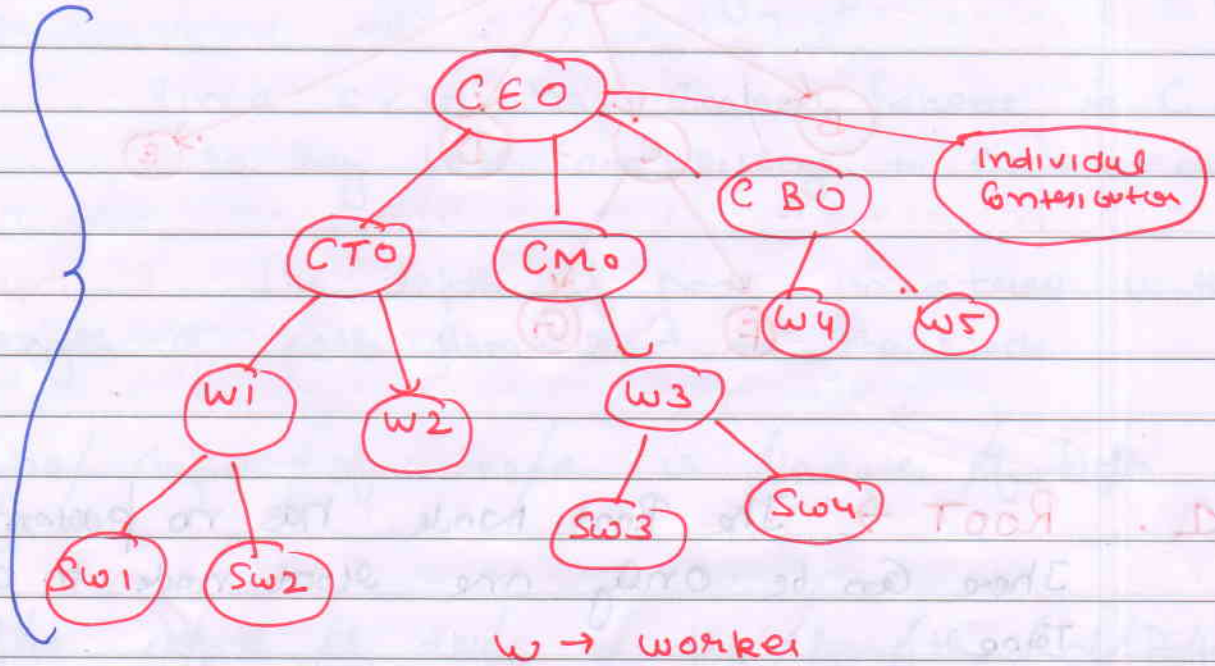


Another example →

Consider a company we have CEO, under CEO we have CTO, CBO, CMO, individual contributors under them we have workers.

DATE

CTO → Company technical officer
CMO → Marketing
CBO → Business



W → Worker
SW → Sub workers

Like, we have head in linked list
In Tree, we have Root node

Note ⇒ Tree is Non linear data structure, A tree structure is a way of representing hierarchical nature of a structure in graphical form

* This was all about the Introduction

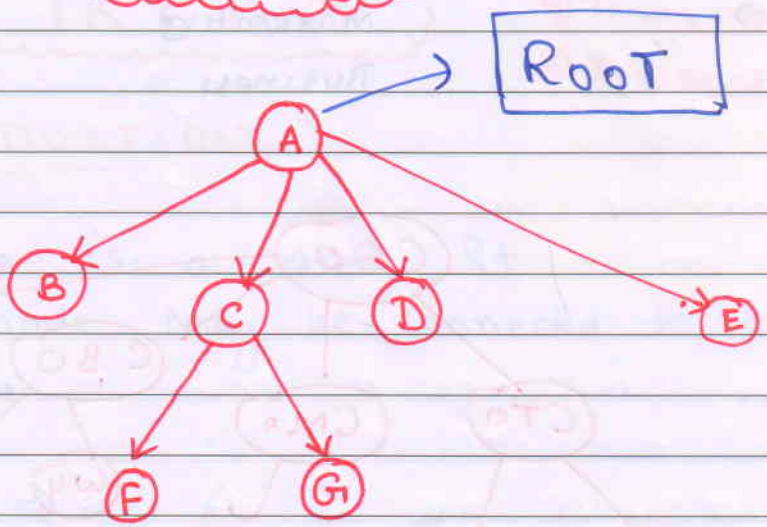
Now, we will see some important terminologies"

Source → Guided path [C N]

4, May, 2022

DATE

Terminologies



1. **ROOT** → The Root Node has no parent Node There Can be only one root node in a Tree

2. **Parent** → For a given node, its immidate predecessor is known as parent node

   eg → A is parent of B, C, D, E
   C is parent of F, G

3. **Edge** → An edge refers to the link from parent node to child node.

4. **Leaf** → A node with no Children

   eg → B, D, E
   F, G

5. **Siblings** → The nodes with same parents.

DATE

eg →

Since B, C, D, E are Child of A
So they are Siblings with each other

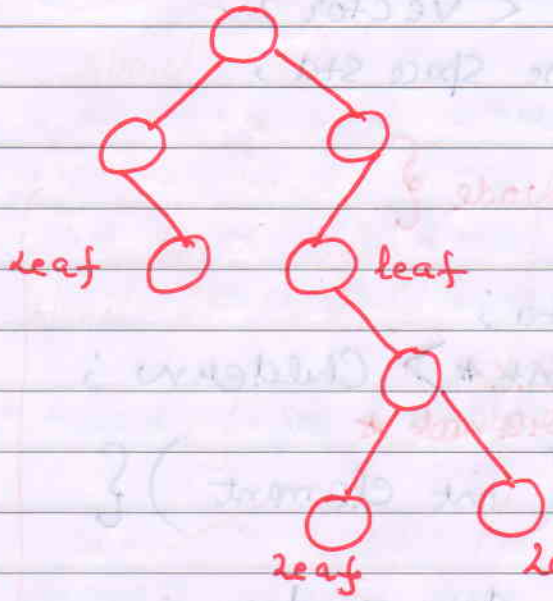
Since F, G has Common parent ie C
So they both are Sibling to each other

6. Depth → The depth of Node in a tree is the length of path from root to the Node

The depth of a Node is length of Path from

The depth of tree is the length of Path

The depth of the tree is the Maximum depth among all the Nodes in Tree.



DEPTH : 0

DEPTH : 1

DEPTH : 2

DEPTH : 3

DEPTH : 4

Hei
=

DATE

7 Decendent → Node k niche wali Sari Nodes

eg →

B, C, D, E, F, G are decendent of A

8 Ancestor → Node k upar wali ~~Sari~~ nodes

eg → A, C are ancestor of F

5, MAY, 2022

TREE Node Class

{A tree Node has a data and address of Multiple Nodes.}

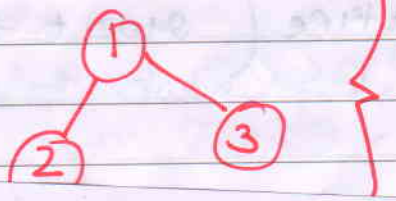
Code →

```
#include <vector>
using namespace std;

Class TreeNode {
  Public:
  int data;
  Vector <TreeNode*> Childern;
  Treenode (int element) {
    this → data = data;
  }
};
```

Self Referential Structure

DATE

```
int main () {
  // Creating treenodes.
Treenode * root = new treenode (1); — (1)

Treenode * node1 = new treenode (2); — (2)

Treenode * node2 = new treenode (3); — (3)

  // Connecting Nodes

root -> Childern.push_back (node1);
root -> Childern.push_back (node2);

}
```



till now we have Made this tree

10 May 2022

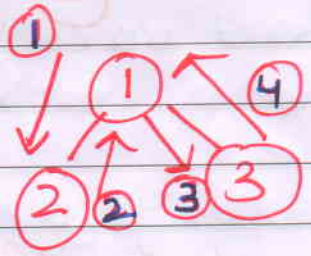
DATE

Taking Tree Input & Printing Tree

★ PREE ORDER TRAVERSAL

```
Void printtree(Treenode * root)
{
    Cout << root -> data << endl;

    for( int i=0; i < root -> Childerw.size()
    {
        printtree( root -> Children[i] );
    }
}
```

DRY RUN →



Step 1 → we printed root data → 1

Step 2 → Our Root has two childerw & loop will work twice

Now we Called print tree for root first Child ie 2

Since 2 has NO Childen. Now loop will not Move further it will return

DATE

Now since we have returned back Now root will call Another child and same happens

Output → 1
2
3

After seeing output we cannot say who is the child of whom So Just by doing some Modification →

Copy code

```
Void printtree ( treeNode * root )
{
  Cout << root → data << ":";
  for ( int i=0; i < root → Children.size(); i++ )
  {
    Cout << root → Children[i] → data << ",";
  }
  Cout << endl;
  for ( int i=0; i < root → Children.size(); i++ )
  {
    printtree ( root → Children[i] );
  }
}
```

Output → 1 : 2 , 3 ,
2 :
3 :

classmate

PAGE

DATE

Note → ~~Abhi~~ In the above code I have not written base case

because we are using for loop it will only call till Children.size() so no further calls will be done

12 May

Take input

Noob - Approach

```
Treenode * takeInput() {
    // Creating tree node
    int rootdata; cout << "enter data";
    cin >> rootdata;
Treenode * root = new treenode(rootdata);
// Asking no of Children of root
int n;
cout << enter no of Children of << root data
cin >> n;
// Calling function again for each Children.
for (int i=0; i<n; i++)
{
Treenode *child = takeInput()
}
for (int i=0; i<n; i++)
{
    Treenode * child = takeInput();
    root → children.push_back(child);
}
}
```

DATE

Input / output →

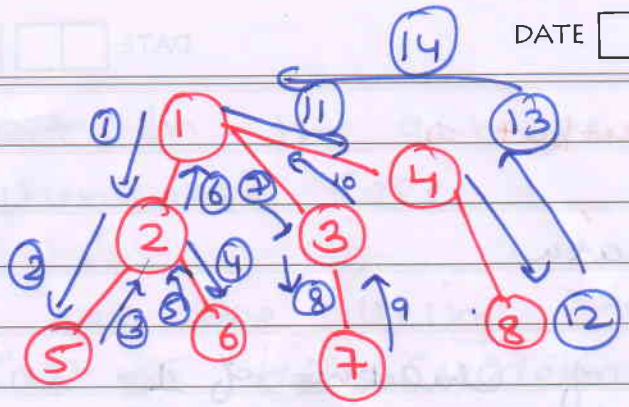
enter data
1
enter no of Children of 1
3
enter data
2
enter no of Children of 2
2
enter data
5
enter no of Children of 5
0
enter data
6
enter no of Children of 6
0
enter no of Children of 3
1
enter data
7
enter no of Children of 7
0
enter data
4
enter no of Children of 4
1
enter data
8
enter no of Children of 8

output →
1 : 2, 3, 4
2 : 5, 6
5 :
6 :
3 : 7
7 :
4 : 8
8 :

output →
1 : 2, 3, 4
2 : 5, 6
3 :
4 :
5 :
6 :
7 :
8 :

DATE

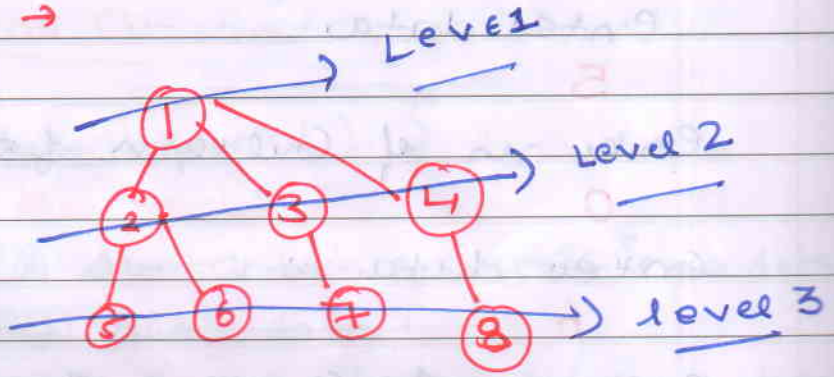
Recursion TREE →



13 May, 2022

Take Input Level Wise

Meaning →

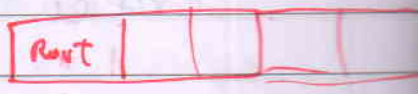


First take (1) as input
then all Children of (1) → (2), (3), (4)
then Children of (2) → (5), (6)
and so on.

Jis order mei chize ayegi uss order mei laga dena hai

A perfect use case of Queue

working → Input Root put in Queue



take root out and ask for Children
put children in queue and link to root
then take out and process continuously
until queue gets empty.

classmate

PAGE

15 May | 2022

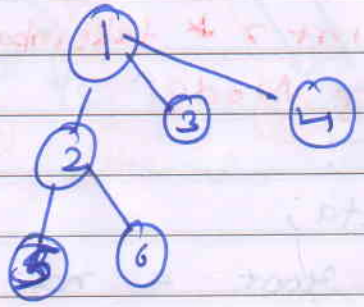
Code →

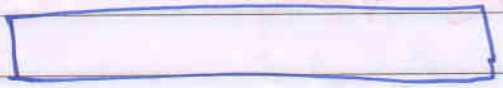
```
TreeNode<int>* takeInputLevelWise() {
    // Creating Node
    int data;
    cin >> data;
    TreeNode<int>* root = new treenode(data);
    // Creating queue.

    queue<TreeNode*> pendingnodes;
    // Pushed Root in queue
    pendingnodes.push(Root);

    while(pendingnodes()!=
    // While queue is not empty
    while(pendingnodes.size() != 0) {
        // taken root out
        TreeNode* front = pendingnodes.front();
        Pendingnodes.pop();
        // Asked for no of Children of root
        Cout << "enter num of childern of " << front->data << endl;
        int numofchild;
        cin >> num of Child;
        // Pushed all Children of root and linked to root
        for(int i=0; i<numofchild; i++) {
            int ChildData;
            Cout << "Enter " << i << "th child of " << front->data << endl;
            Cin >> ChildData;
            TreeNode* Child = New TreeNode(Childdata);
            front->Childern.push_back(Child);
            pendingnodes.push_back(Child);
        }
    }
}
```

PAGE

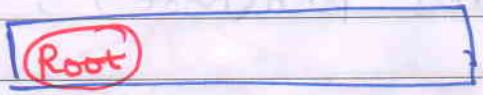
DATE

DRY RUN

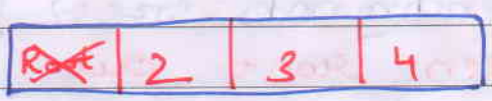


queue → 

Pushed Root

queue → 

Removed root and asked for Childern

queue → 

Now Childern are pushed in queue and
do well as linked (font → Childern push_back

Again go to while loop since queue is not empty

Taken (2) out and asked for Childern

:

Same happens till queue gets empty.

DATE

Input / output →

Enter root data
1
Enter no of childern of 1
3
enter 0th child of 1
2
enter 1th child of 1
3
enter 2th child of 1
4
enter no of childern of 2
2
enter 0th child of 2
5
enter 1th child of 2
6
enter no of childern of 3
0
enter no of childern of 4
0
enter no of childern of 5
0
enter no of childern of
0

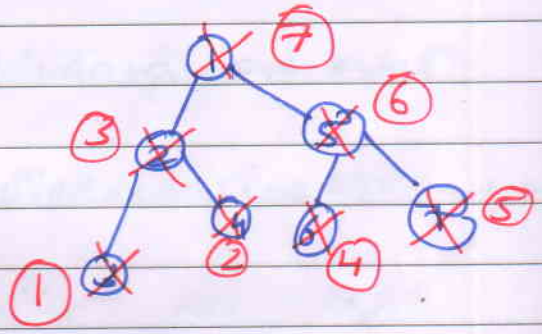
18, MAY, 22

DATE

Distructor

- To delete a whole tree

```
Void deletetree (Treenode * root)
{
  for(int i=0; i < root->Children.size(); i++)
  {
    deletetree(root->Children[i]);
  }
  delete root;
}
```



" Root will be deleted in the end

Root bolega → Sabse Pehle Mujhe nai Mere Children ko delete karo.

- Alternate way → Use distructor

```
~Treenode() {
  for(int i=0; i < root->Childen.size(); i++)
  {
    delete Children[i];
  }
```