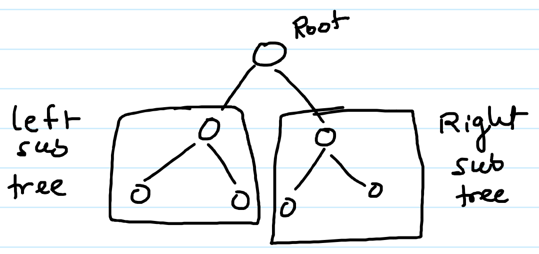
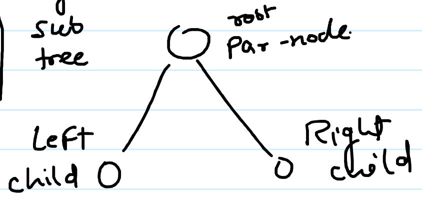
# Day19

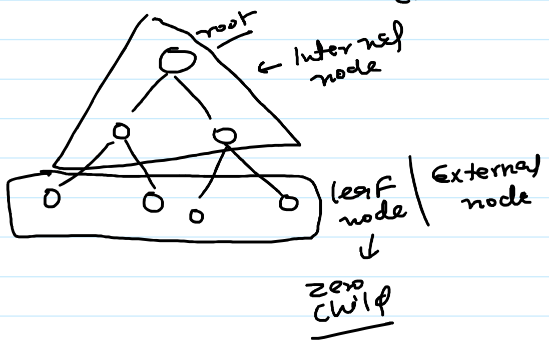
08 April 2025 08:34 AM

**Tree** Non Linear DS

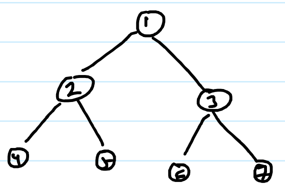
Linear DS.
- Array
- List
- Stack
- Queue

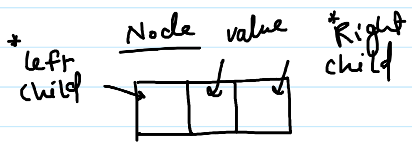




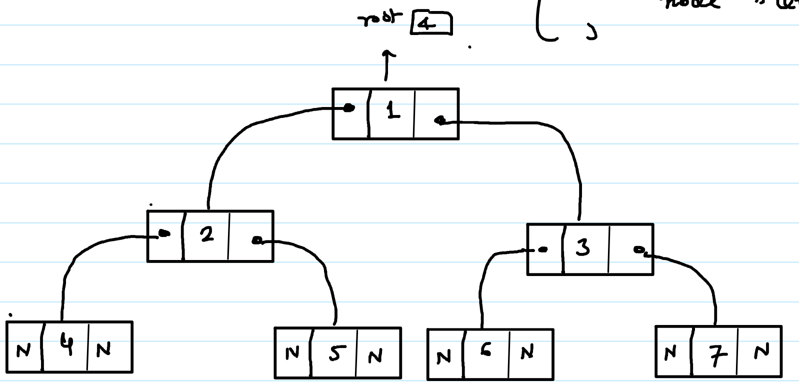


**Binary tree** → * max child - 2





```
class node
{
        int value;
        node *left, * right;
}
```

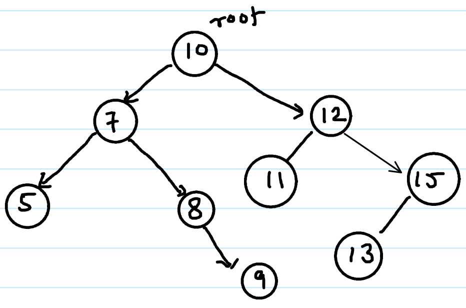


① BST → **Binary Search tree** →

- * max child - 2
- * left child < root < right child

**Draw BST** :-

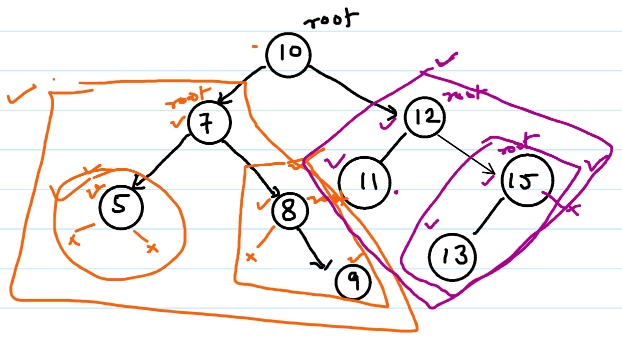
10, 7, 5, 8, 9, 12, 15, 11, 13



Traverse

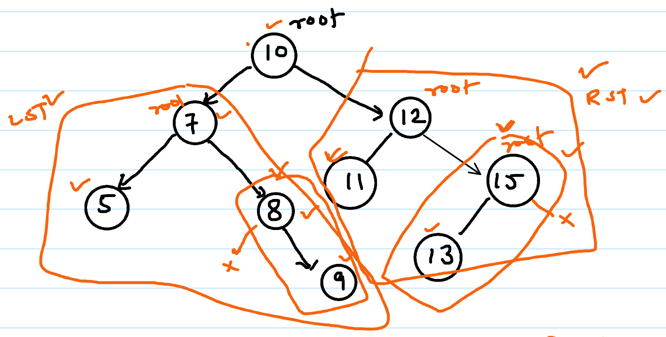
① Inorder [LST, Root, RST]
② postorder [LST, RST, Root]
③ Preorder [Root, LST, RST]

Inorder traverse [LST, Root, RST]



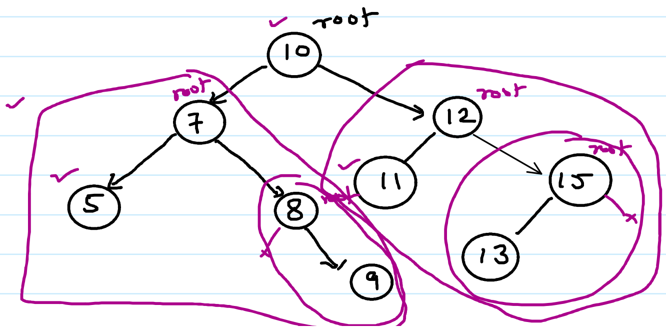
$\boxed{5, 7, 8, 9, 10, 11, 12, 13, 15}$

Postorder . [LST, RST, Root]

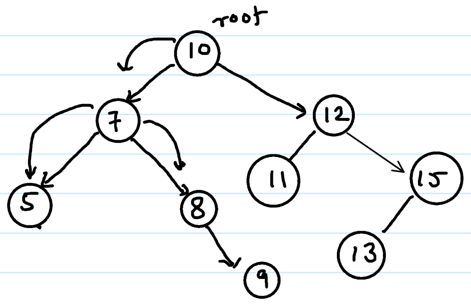


5, 9, 8, 7, 11, 13, 15, 12, 10

Preorder [Root, LST, RST]



10, 7, 5, 8, 9, 12, 11, 15, 13



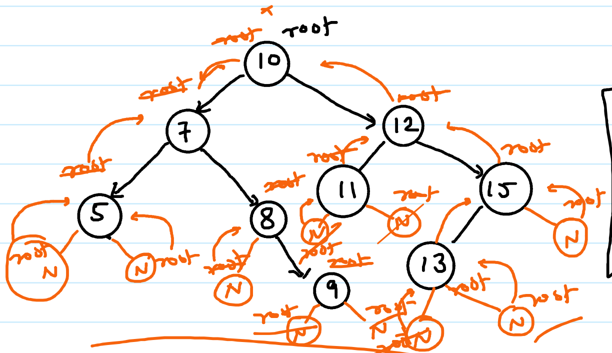
cout<< root -> left -> val

PTR
~~root~~ = root -> left

~~root~~ = root -> right
PTR

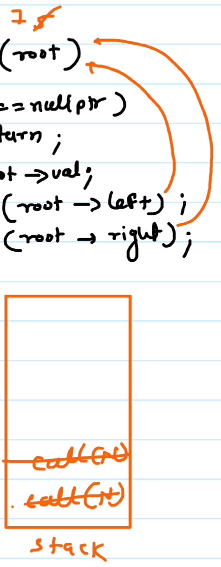
## Pre order

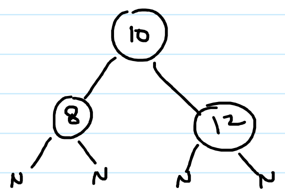
```
void  Preorder (root)
{
      if (root == nullptr)
           return ;
      cout << root -> val;
      Preorder (root -> left) ;
      preorder (root -> right);
}
```



10, 7, 5, 8, 9, 12, 11, 15, 13

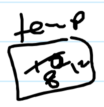
```
void  Preorder (root)
{
      if (root == nullptr)
           return ;
      cout << root -> val;
      Preorder (root -> left) ;
      preorder (root -> right);
}
```

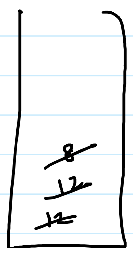




## Preorder

```
stack< TreeNode * >  STK
STK. push (root)
while (!stk. empty ())
{
        temp = stk. top ()
cout<< temp. val
~~matra~~ stk. push (temp -> ~~left~~ right)
```





$\boxed{\cancel{10}\ 8}$

$\boxed{12, 8, 12}$

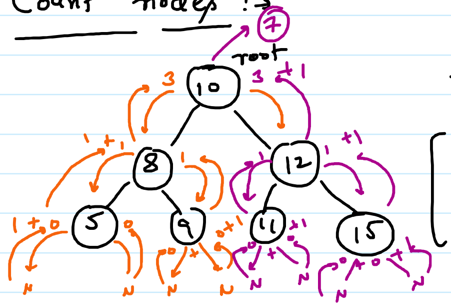
$\boxed{\cancel{8}\ \cancel{12}\ \cancel{12}}$

```
cout<< temp.val
                  right
not  { stk.push (temp->~~left~~)
null { stk.push (temp->~~right~~)
                        left
```

3

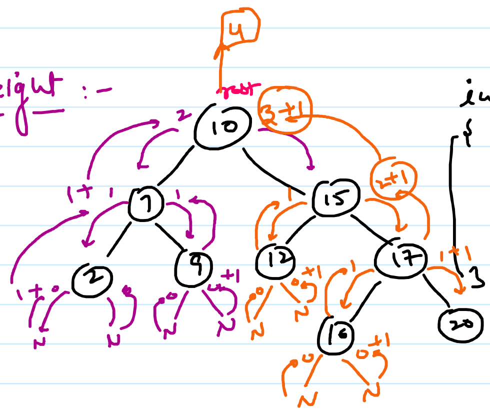
Leetcode - 144
 - 145

Count nodes :→



```
int  CountNode (root)
{
    if (root==nullptr)
        return 0;
    return CountNode (root->left) + CountNode(root->right) + 1;
}
```

height :-



```
int  height (root)
{
    if (root ==nullptr)
        return 0;
    return max (height(root->left), height (root->right)) +1;
}
```

Searching BST

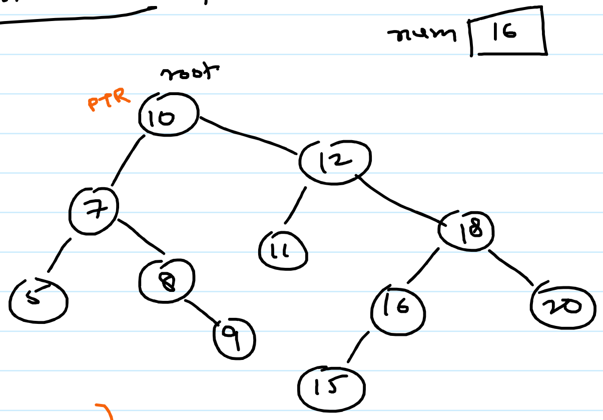
Leetcode 111

```
① if (root ==null)
        return 0;
② if (root -> left == null)
        return mindepth (root->right) +1;
③ if (root -> right == null)
        return min_ (root->left) +1;
④
return min(mindepth (root->left)  mindepth (root->right) + 1;
```

Search in BST :→

① Non Recursive method

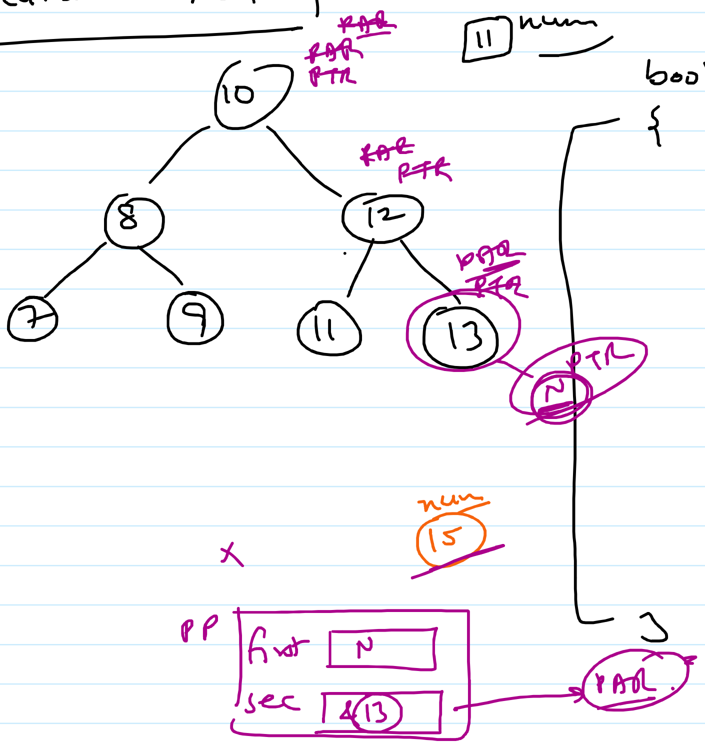
① Non Recursive method



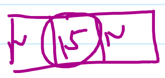
$o(\log_2 n)$

```
bool Search (Node* root, num)
{
    node* PTR = root
    while (PTR != nullptr)
    {
        if (PTR->val == num)
        {
            return true;
        }
        if (num < PTR->val)
            PTR = PTR->left;
        else
            PTR = PTR->right;
    }
    return false;
}
```

② Recursive method



```
bool Search (root, num)
{
    if (root == nullptr)
        return false;
    if (root->value == num)
        return true;
    if (num < root->val)
        return search (root->left);
    else
        return search (root->right);
}
```



```
#include<iostream>
using namespace std;
class node{
    public:
        int val;
        node *left,*right;
        node(int x)
        {
            val=x;
```

```
                left=right=nullptr;
            }
    };
    class BST{
        node*root;
        pair<node*,node*> pp;
        public:
            BST()
            {
                root=nullptr;
            }
            bool search(int num)
            {
                node *ptr = root;
                node *par = nullptr;
                while(ptr != nullptr)
                {
                    if(num == ptr->val)
                    {
                        pp=pair<node*, node*>(ptr,par);
                        return true;
                    }
                    par = ptr;
                    if(num < ptr->val)
                        ptr = ptr->left;
                    else
                        ptr = ptr->right;
                }
                pp=pair<node*, node*>(ptr,par);
                return false;
            }
            void addBST(int num)
            {
                if(search(num))
                {
                    cout<<"Duplicate element\n";
                    return;
                }
                node* par = pp.second;
                if(par == nullptr)
                {
                    root= new node(num);
                }
                else
                {
                    if(num < par->val)
                        par->left = new node(num);
                    else
                        par->right=new node(num);
                }
            }
            void inorder(node* root)
            {
                if(root==nullptr)
                    return;
                inorder(root->left);
                cout<<root->val<<" ";
                inorder(root->right);
            }
            void traverse()
            {
                inorder(root);
                cout<<endl;
            }
    };
    int main()
    {
        BST tree;
        tree.addBST(10);
        tree.addBST(8);
        tree.addBST(9);
        tree.addBST(12);
        tree.addBST(11);
        tree.addBST(13);
        tree.addBST(7);
        tree.traverse();

        return 0;
    }
```

H.W

Leetcode → 100

Leetcode → 101