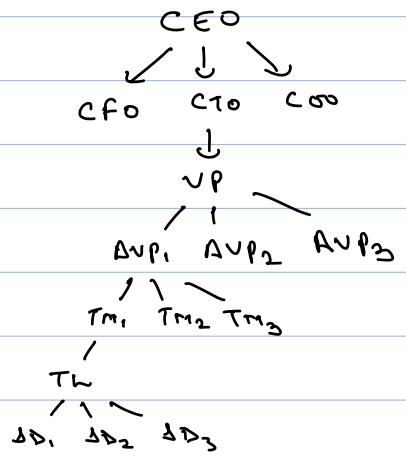
Today's Content

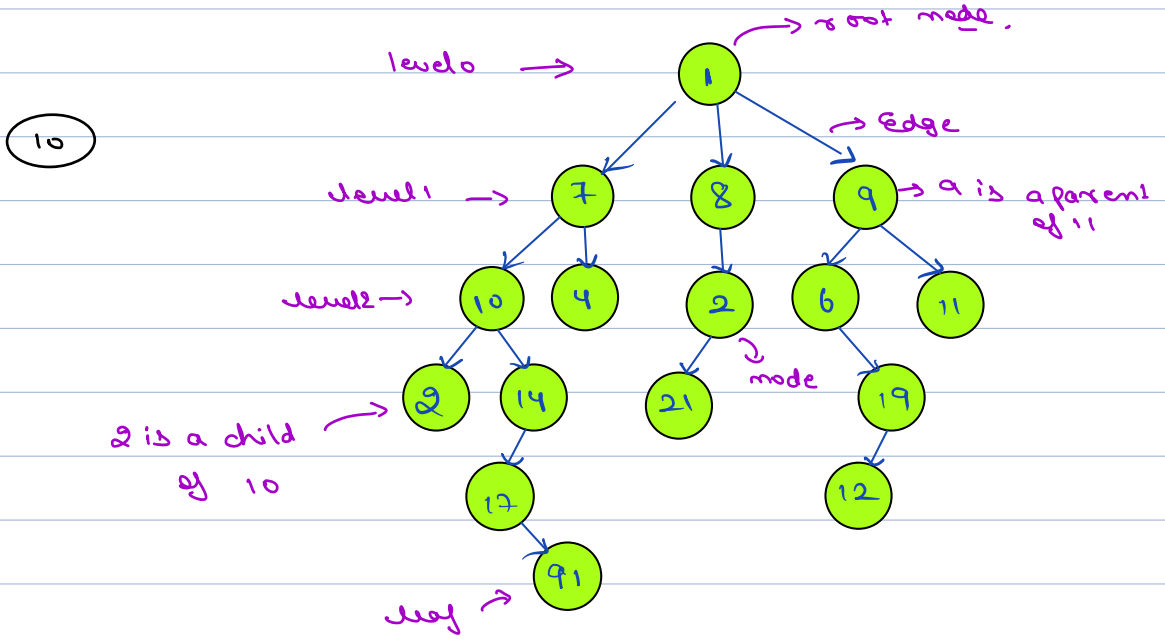
- → Binary Tree
- → Traversal
- → Iterative traversals
- → Construct B.T from Preorder to Inorder

Trees Basics & terminologies

// Linear data structures → arrays, stacks, LL.

// Hierarchical :-



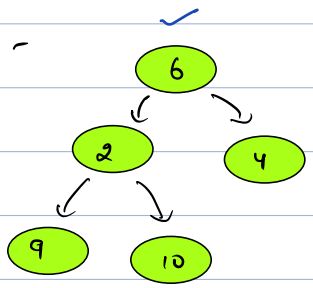


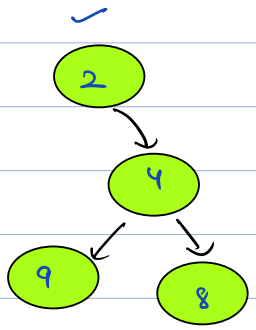
root → It is a node with no parent.

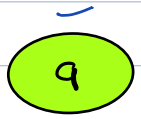
leaf → It is a node with no children.

Binary Tree :- for every node, no. of children <= 2.

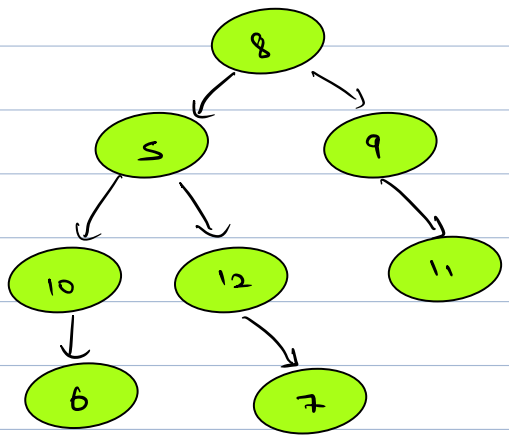
Ex 1 :-





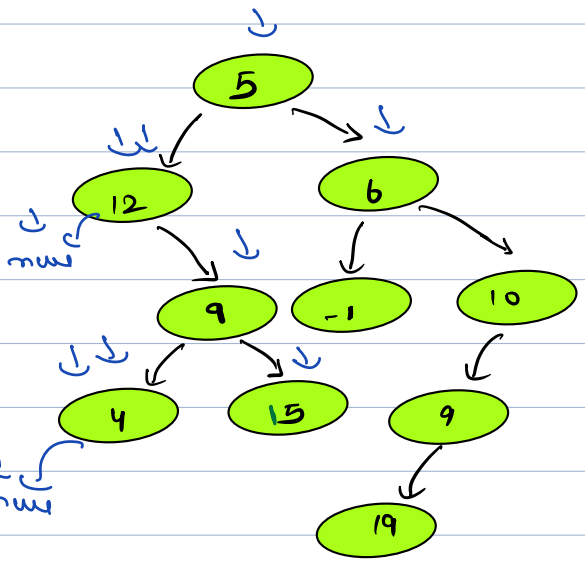



```
class Node {
    int data;
    Node left;
    Node right;
    Node(x) {
        data = x;
        left = null;
        right = null;
    }
}
```



Tree Traversals :-

- → Inorder
- → Preorder
- → Postorder



1) Inorder (L → R)

12 4 9 15 5

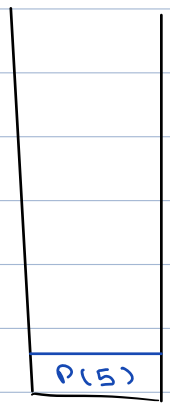
-1 6 19 9 10

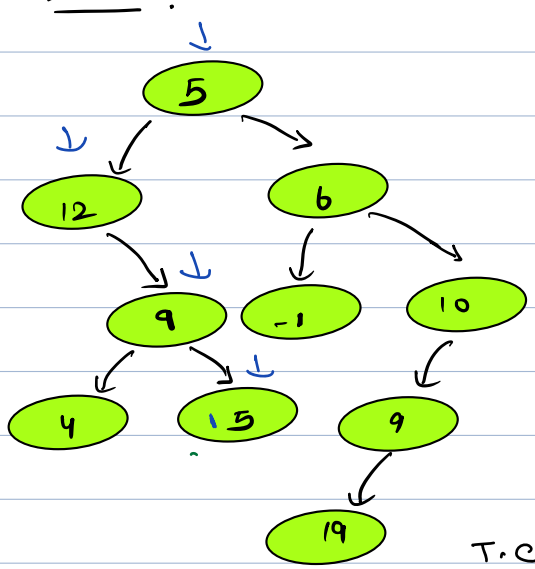
T.C → O(N)

S.C → O(H)

```
void inorder (Node root) {
    if (root == null) {return;}
    inorder (root.left);
    Print (root.data);
    inorder(root.right);
}
```

2) PreOrder D L R





T.C → O(n).

5 12 9 4 15 6 -1 10 9 19

S.C → O(H).

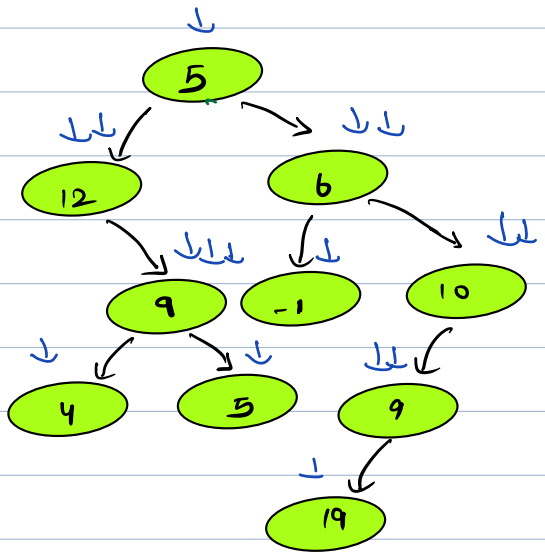
```
void preorder (node root) {
    if (root == null) { return; }

    Print(root.data);
    Preorder(root.left);
    Preorder(root.right);
}
```

Note :- (Each node is touched 1 times).

3) Post Order :- LRD

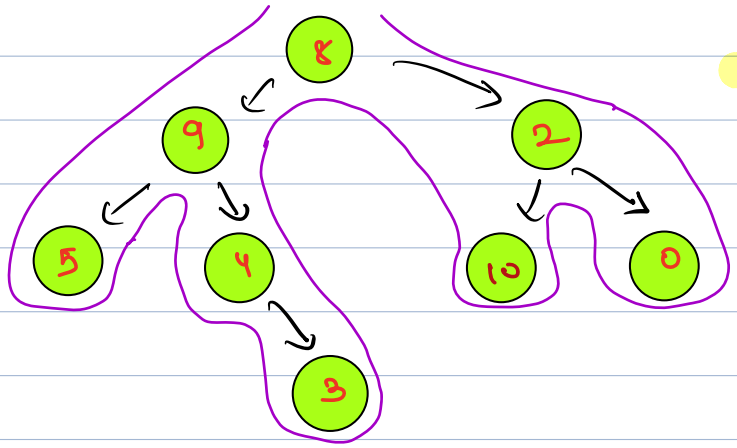
4 5 9 12 -1 19 9

10 6 5




```
void PostOrder (Node root) {
    if (root == null) return;
    PostOrder (root.left);
    PostOrder (root.right);
    Print (root.data);
}
```

T.C → O(n)
S.C → O(H)



Euler Path

(LDR) Inorder 5 9 4 3 8 10 2 0

(DLR) Preorder 8 9 5 4 3 2 10 0

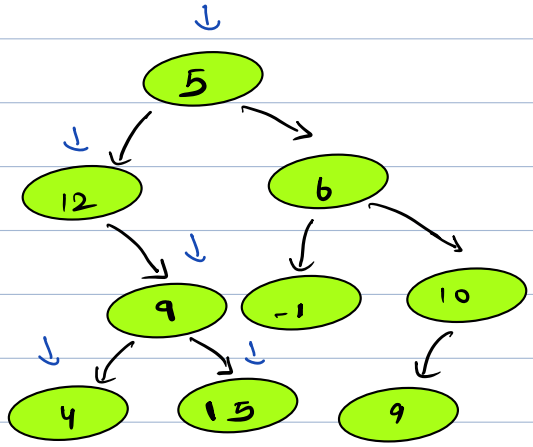
(LRD) PostOrder 5 3 4 9 10 0 2 8

Break 8:20 Am - 8:30 Am

Inorder Iterative :- (L D R).

Steps involved

1) call left child
2) print data
3) call right child .



12 4 9 15 5

node test

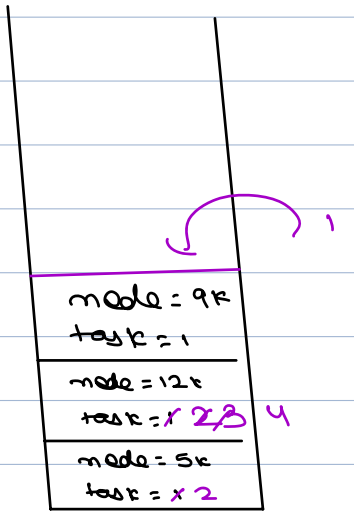
class Pair {

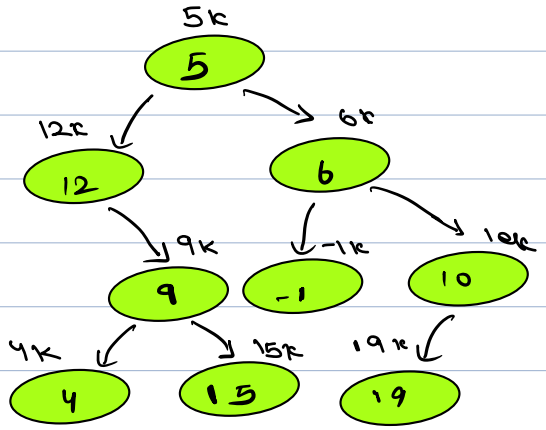
Node node
int task;
Pair (Node t) {
node = t
task = 1
}
}

T.C → O(n)

S.C → O(H)

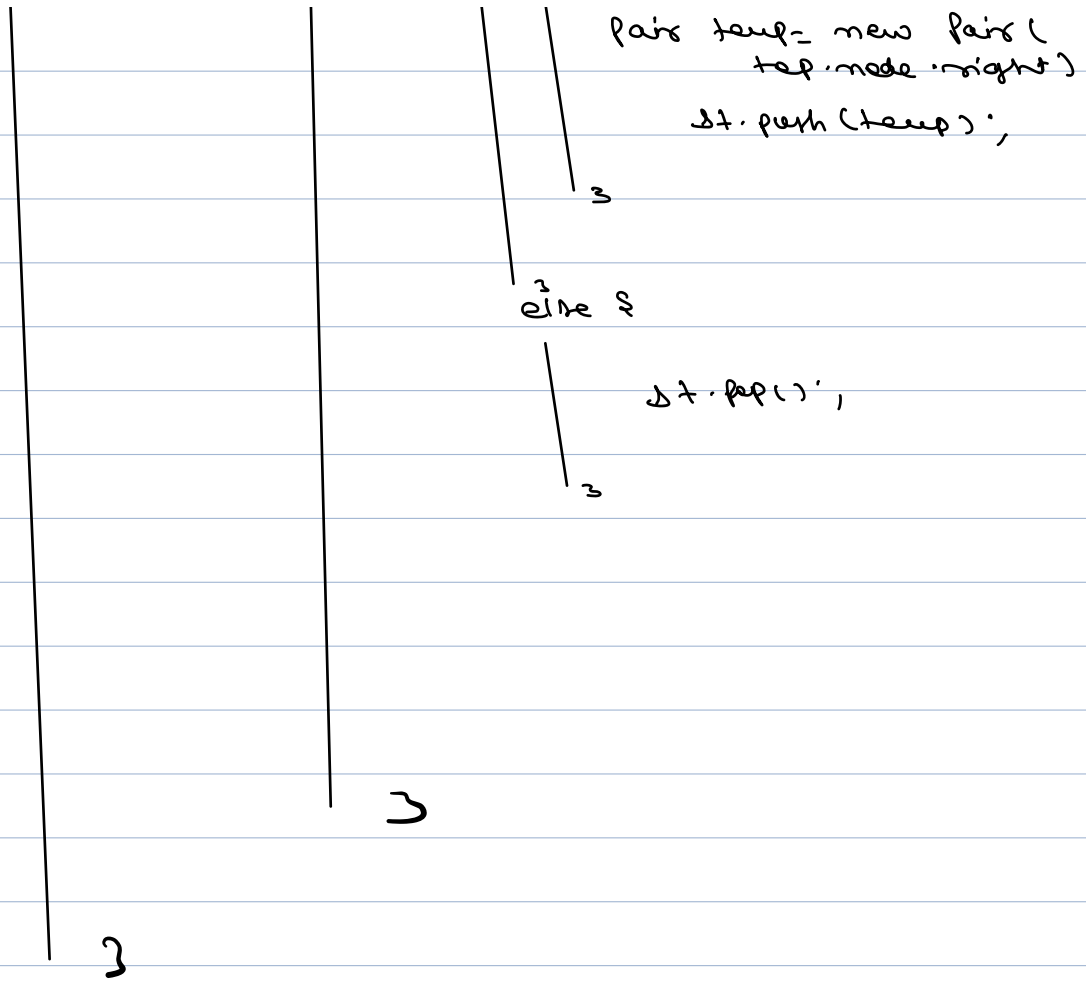
12





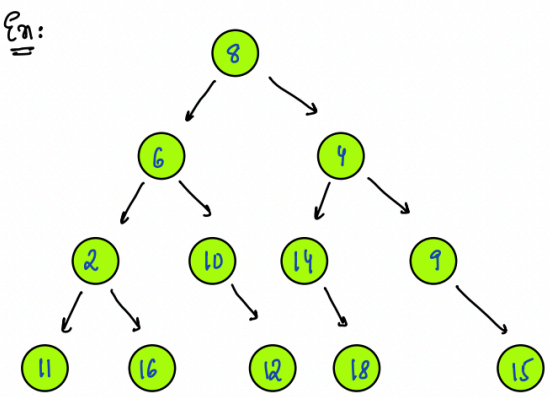
5k
2

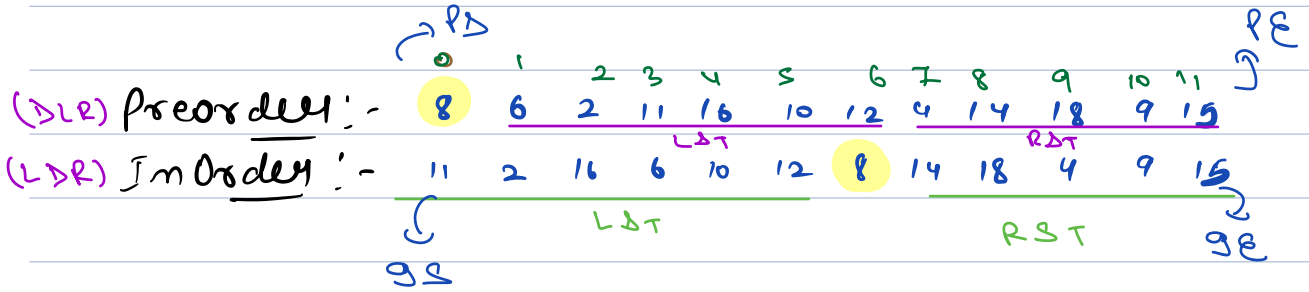
```
public void inorder ( Node root ) {
    Stack <pair > st;
    pair p= new pair (root);
    st.push(p);
    while (st.size()>0) {
        Pair top = st.Peek();
        if (top.task == 1) {
            top.task++;
            if (top.node.left != null) {
                Pair temp= new Pair(
                    top.node.left);
                st.push(temp);
            }
        }
        else if(top.task == 2){
            top.task++;
            Print (top.node.data);
        }
        else if (top.task==3) {
            top.task++;
            if (top.node.right != null) {
```

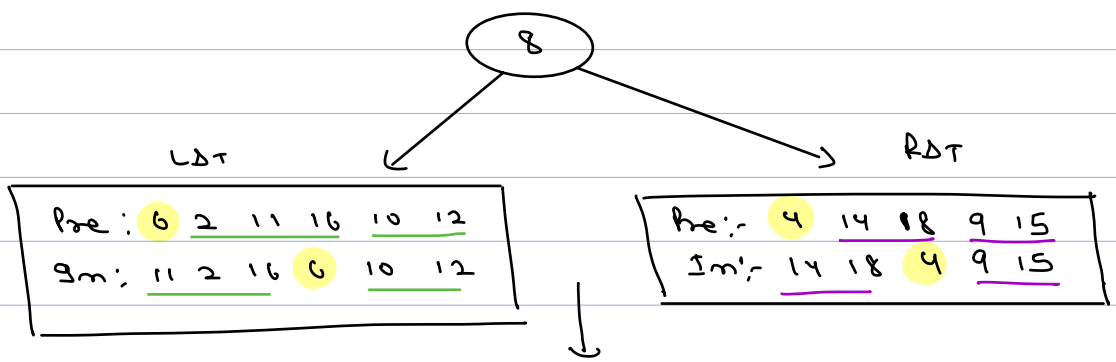

Pair temp = new Pair(
top.node.right)
st.push(temp);
}
else {
st.pop();
}
}

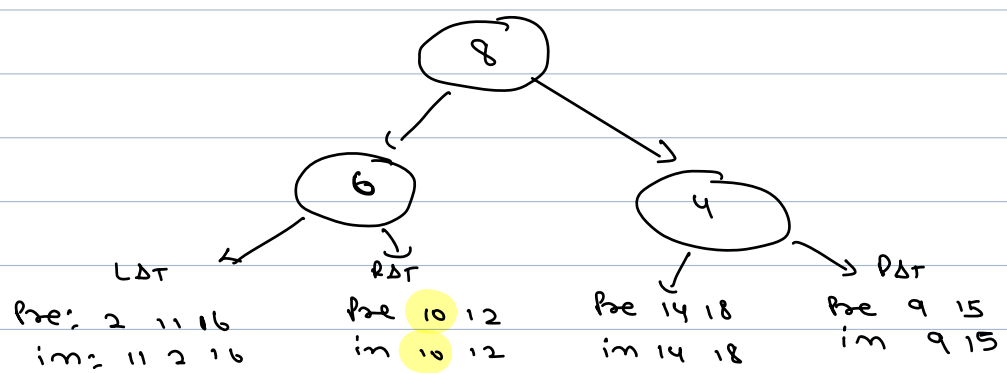
Ques Given Preorder & inorder of a B.T. with distinct values. create it.

Eg:







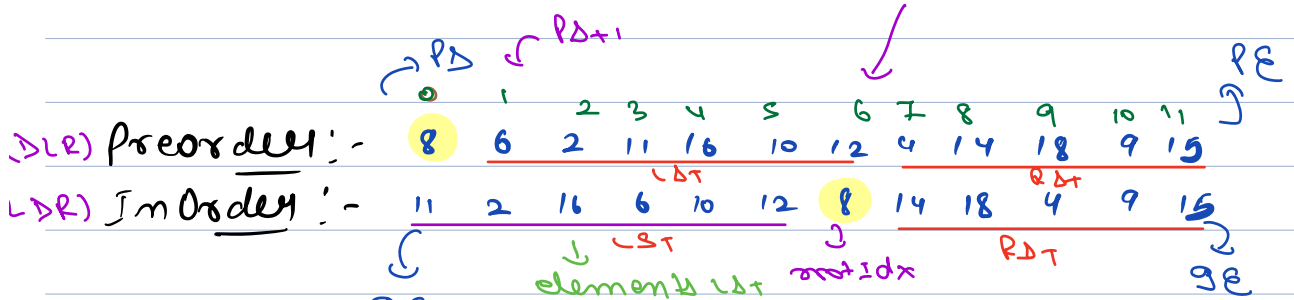


elementsLST = (gs, rootIdx-1); => rootIdx - x - gs + 1 } => rootIdx - gs.

(ps+1, e) = elementsLST

=> e - ps - 1 + 1 = elementsLST

=> e = elementsLST + ps



Pseudo code :- // pre [], in[].

```
Node create (int ps, int pe, int is, int ie) {
    if (ps > pe || is > ie) { return null }

    int rootData = pre[ps];
    Node root = new Node(rootData);
    int rootIdx = find(is, ie, rootData);   -> (HashMap)
    int elementsLST = rootIdx - gs

                                  elementsLST + ps
    root.left = create(ps+1, ↓, gs, rootIdx-1);
    root.right = create(↑, pe, rootIdx+1, ie);

                elementsLST + ps + 1

    return root;
```

3

T.C → O(n)

S.C → O(n+1)

5 → 1 2 3 4 5

3 = 2

map

a → 2
b → 1 2 d → 1
c → 1

~~a b d c~~

a a b b d

a b a d b c

a a b b d d.

a b a d b c

a a b b d d

map

a → x 2
b → x 2
c → 1
d → 1

~~a~~ ~~b~~ d c

---

map

a b c a b c

a a a b c #

a → x 2
b → x 2
c → x 2

~~a~~ ~~b~~ ~~c~~

---

a b b a d b c

a a a #

a → x 2
b → x 2

~~a~~ ~~b~~