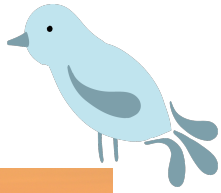
TREES 1



Good Evening

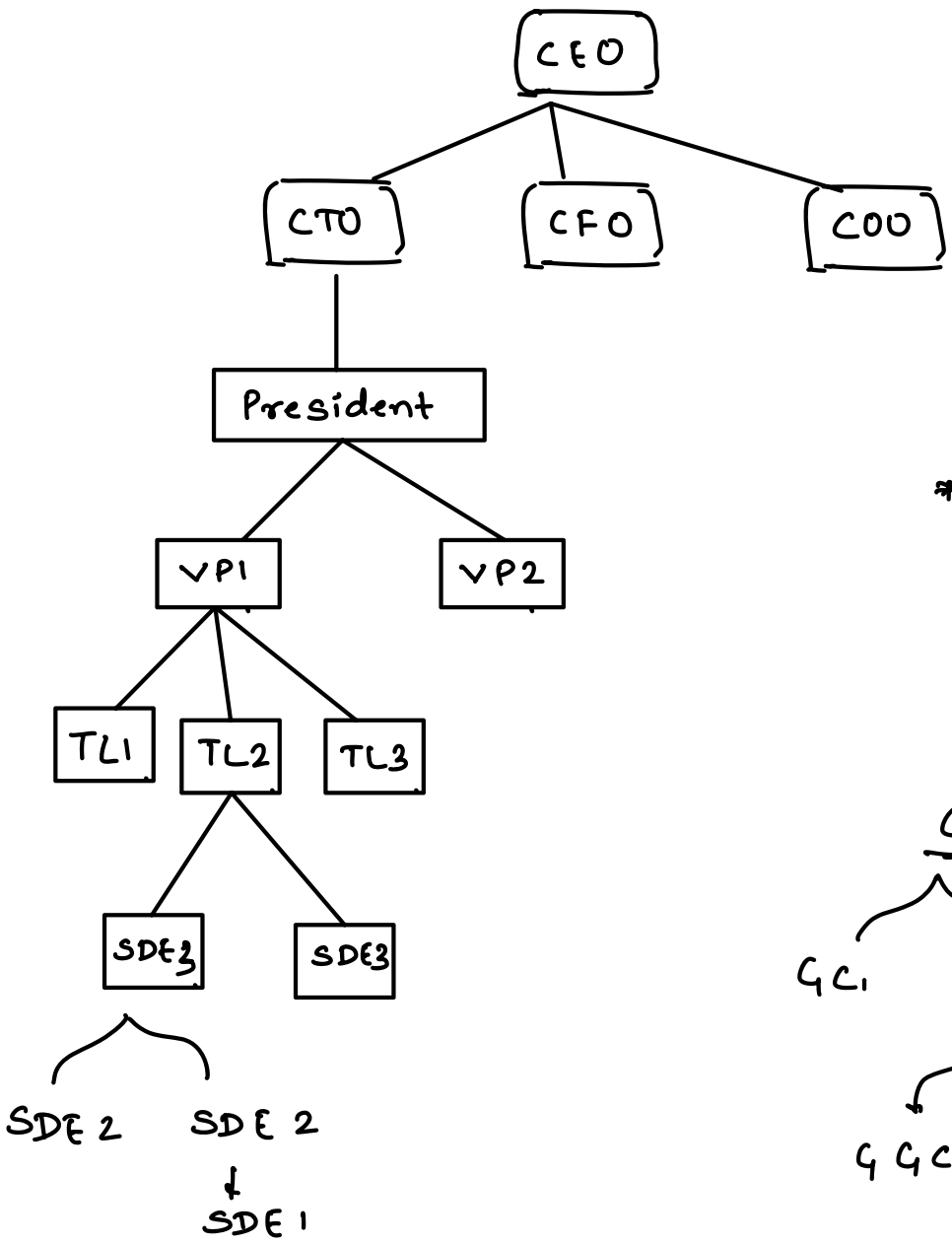
Today's content

- → Trees Intro
- → Naming convention
- → Trees traversal
- → Iterative inorder
- → Construct tree with pre[] & in[]

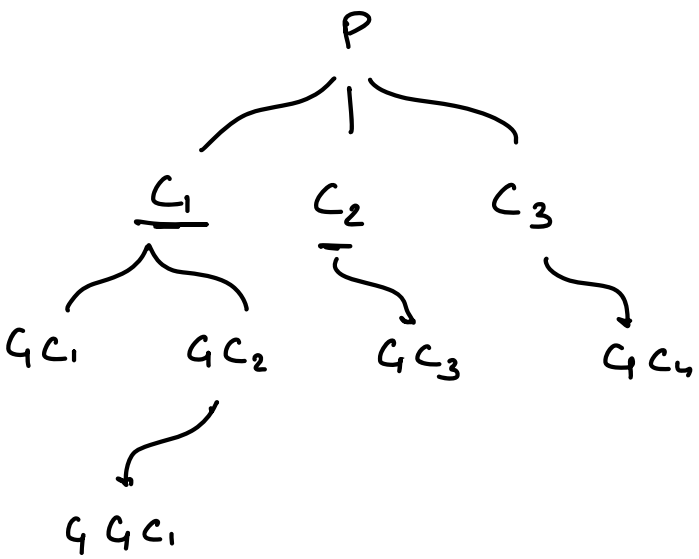
Linear Data Structure

01. Array
02 Stack , queues
03 Linkedlist
04 String

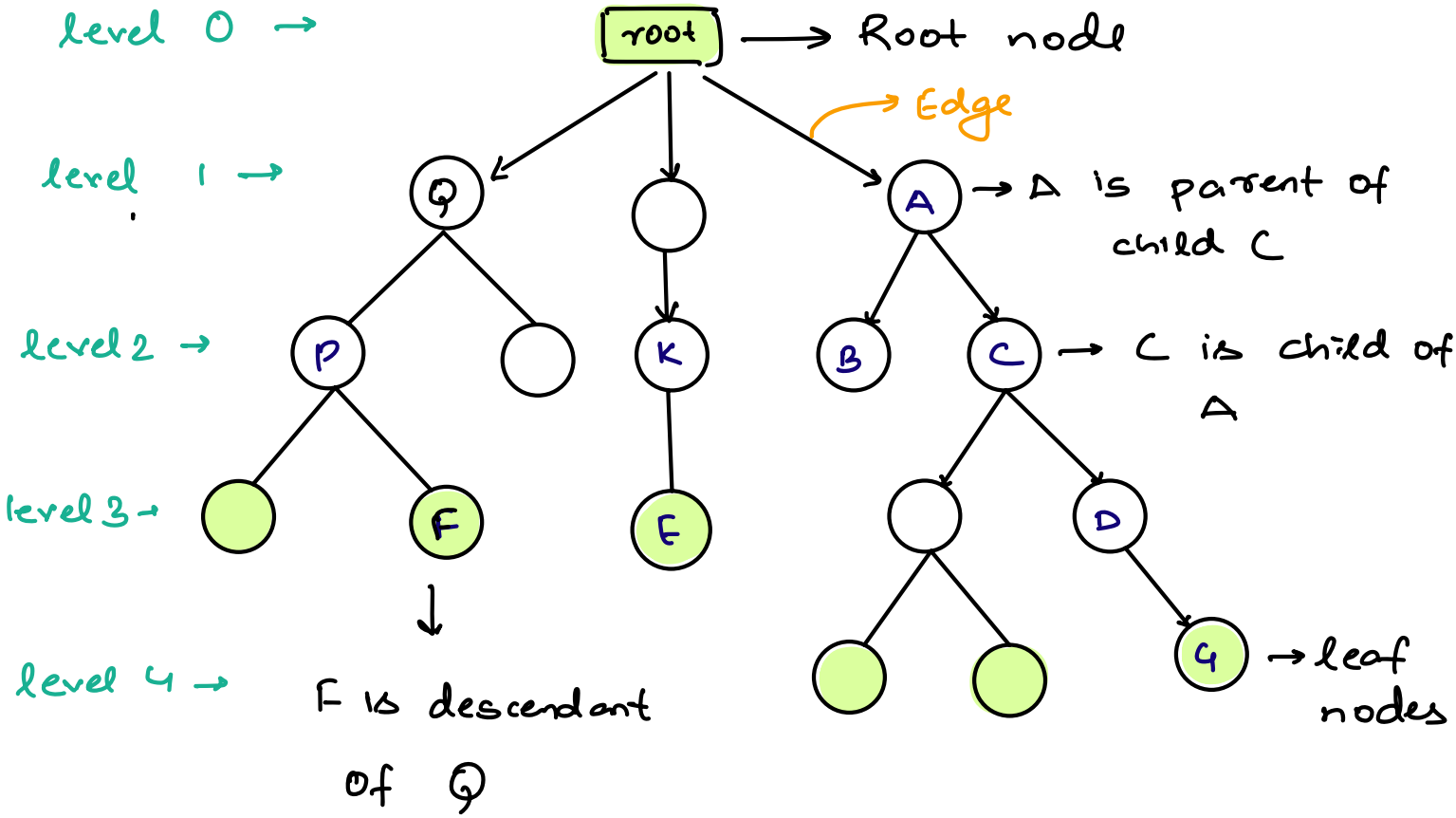
Hierarichal Data Structure



* Govt
* Family tree



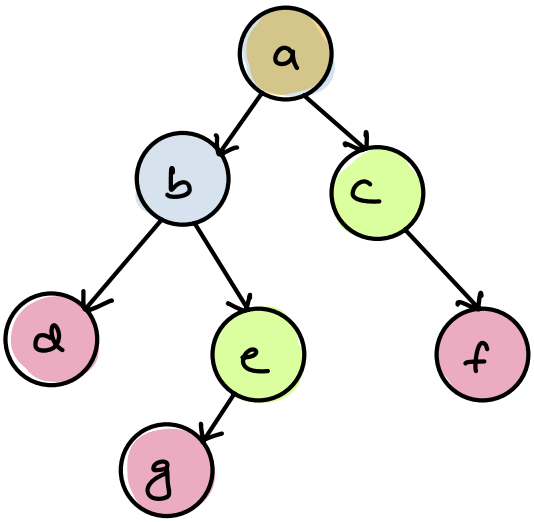
Tree

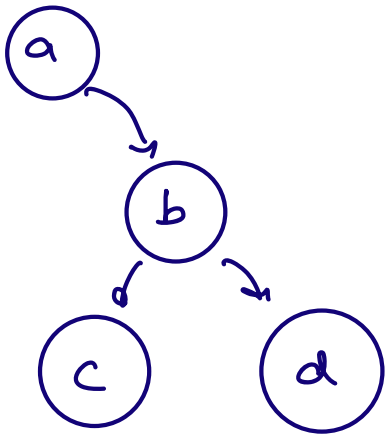


* Height (root) = 5 in terms of nodes

= 4 in terms of edges

Binary Tree :- for every node, we can have atmax 2 children

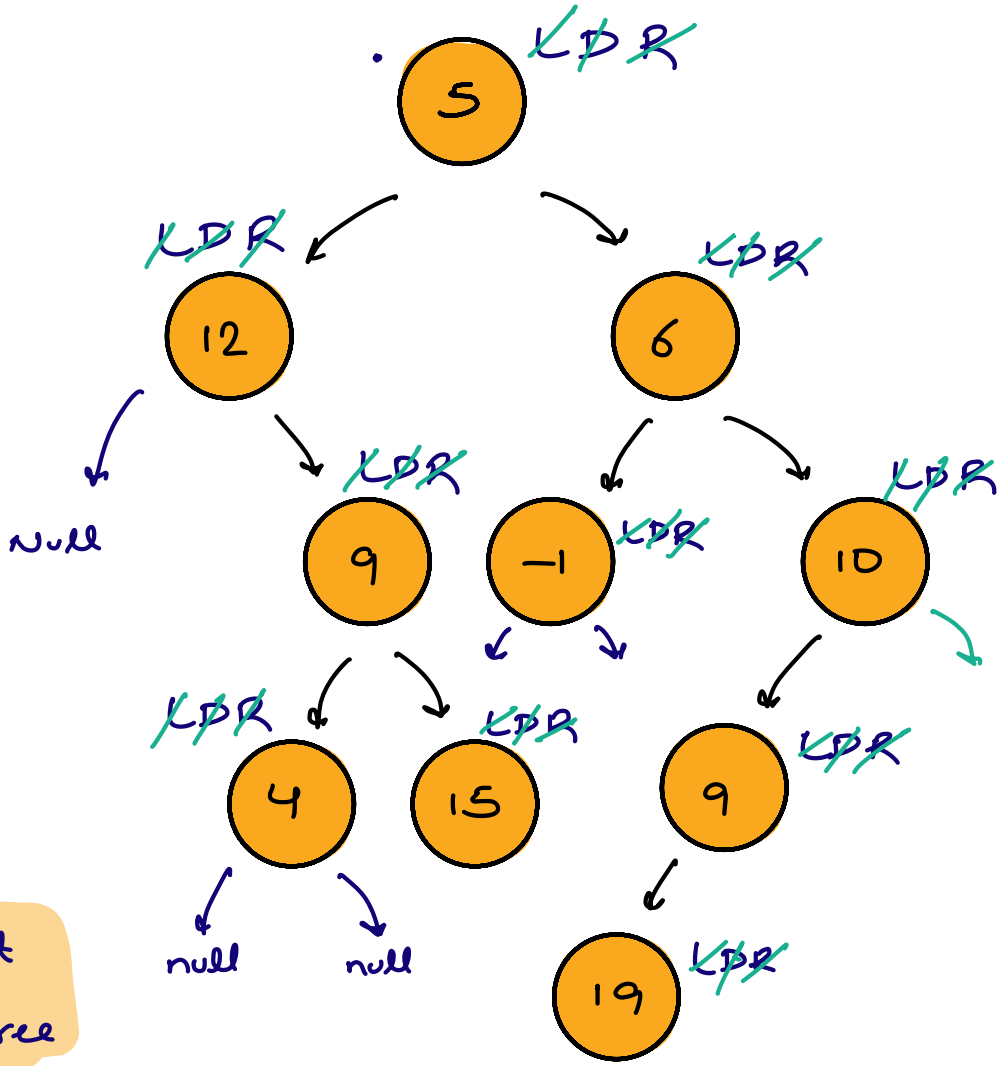





```
class Node {
    int val;
    Node left
    Node right;

    Node (x)
    {
        val = x;
        left = null;
        right = null;
    }
}
```

Tree Traversal

01. Inorder
02. Preorder
03 Postorder



* Inorder Traversal

Left Subtree | Data | Right Subtree

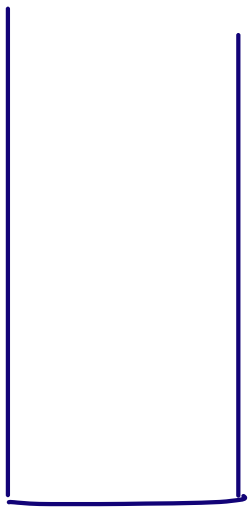
Inor = 12 4 9 15 5 -1 6 19 9 10

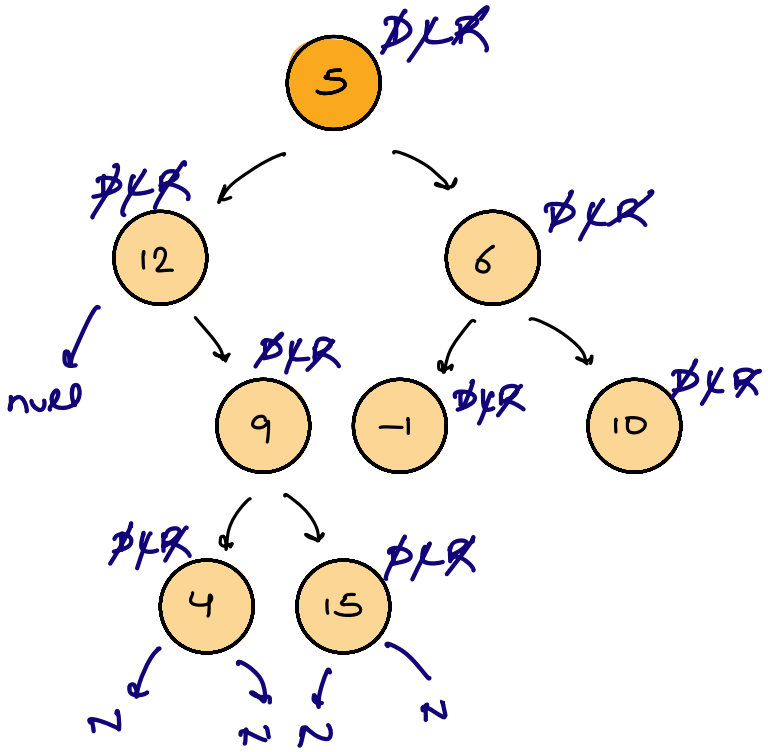
```
void inorder (root)
{
    if (root == null) return;
    inorder (root.left)
    print (root.val);
    inorder (root.right);
}
```

TC: O(N)
SC: O(ht)

* Preorder Traversal

Data	left subtree	Right subtree





Output = 5 12 9 4 15 6 -1 10

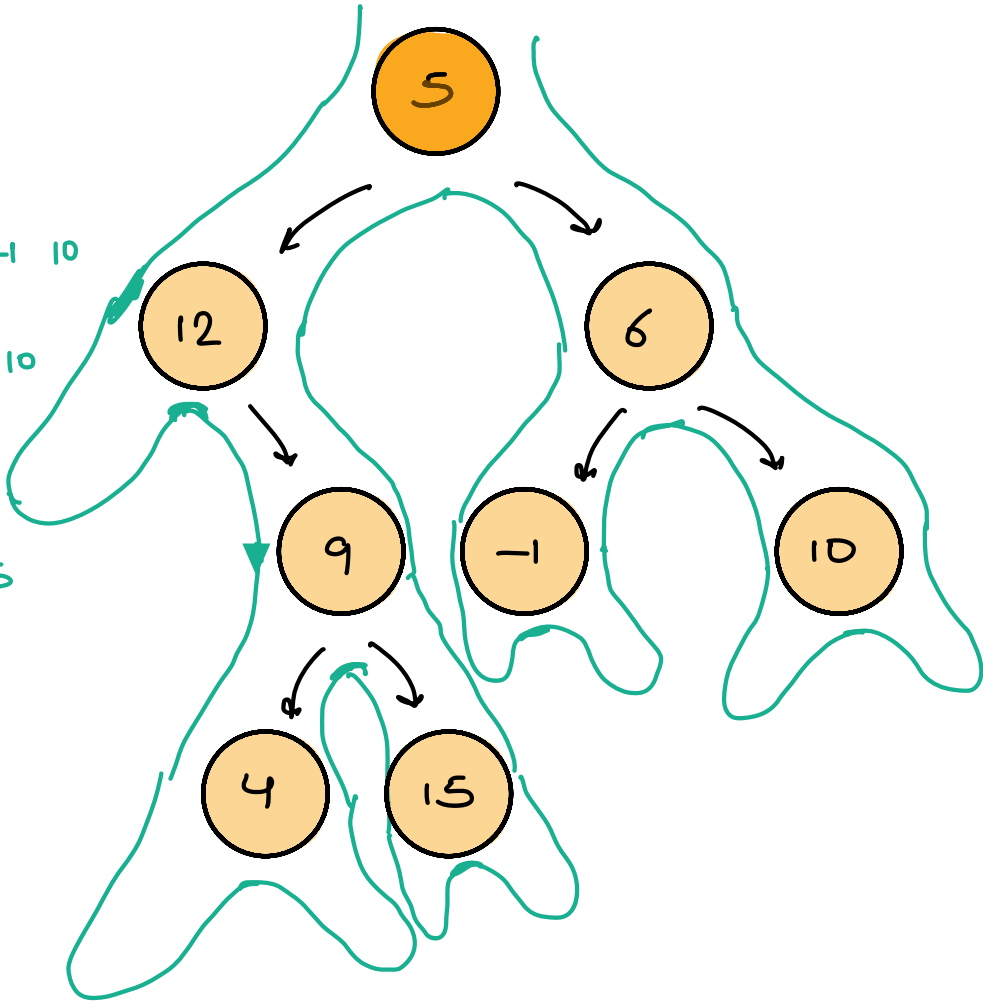
```
void preorder (root)
    if (root == null) return;

      print (root.val);
    preorder (root.left)
    preorder (root.right);
}
```

TC: O(N)

SC: O(H)

Postorder

Left Subtree | Right Subtree | Data



Pre = 5 12 9 4 15 6 -1 10

In = 12 4 9 15 5 -1 6 10

Postorder = 4 15 9 12 -1 10 6 5

```
void postorder (root)
    if (root == null) return;

    postorder (root.left)
    postorder (root.right);
    print (root.val);
```

3

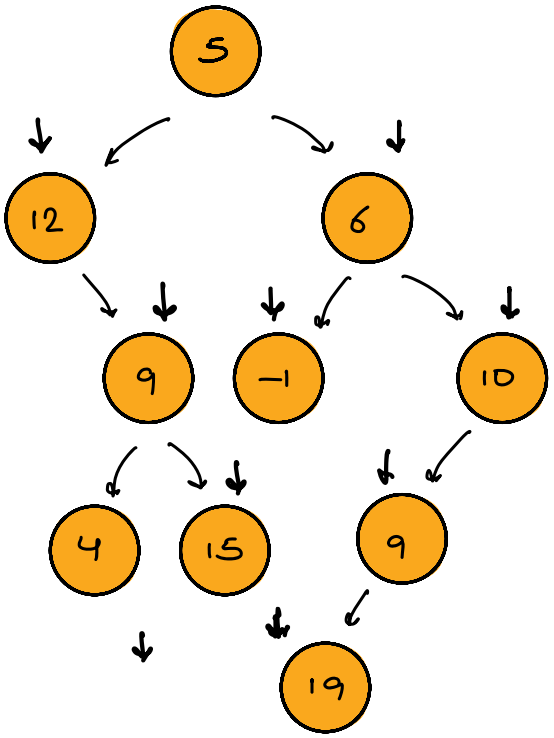
9:53 pm → 10:03 pm

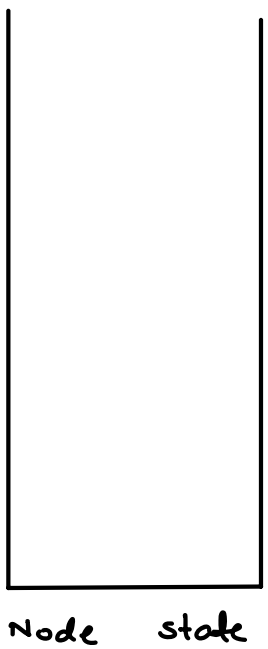
* Inorder Iterative (LDR)

01. Call left child ✓

02. print data .

03. call right child .





12 4 9 15 5 -1 6 19 9 10

state = 1 → add left child

state = 2 → print (root.data)
add right child ←

State = 3 → remove yourself

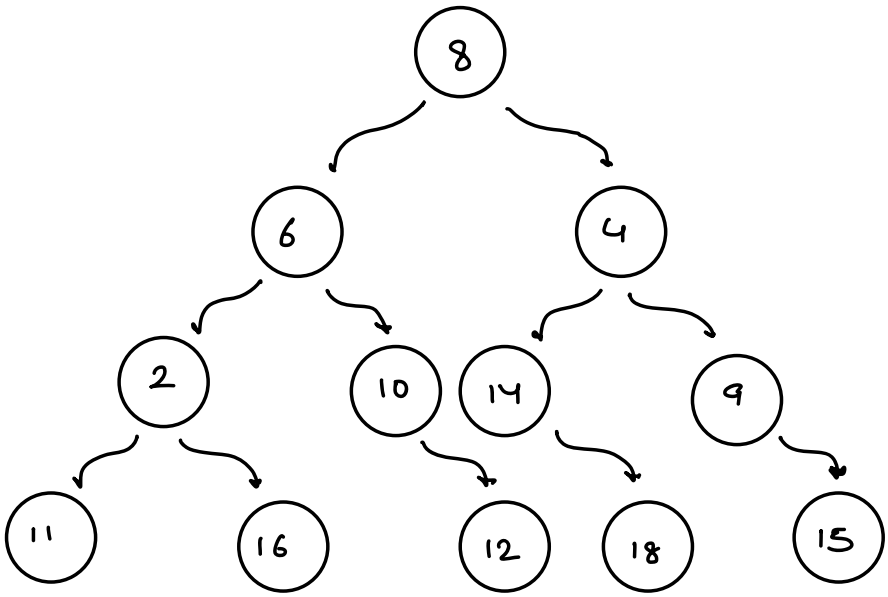
```
class pair {
    Node node;
    int state;
}

public void iterativeinorder (Node root)
    Stack <pair> st;
    Pair p = new pair (root, 1);
        st.push(p);
        while (st.size () > 0)
            pair top = st.peek();
            if (top.state == 1)
                top.state ++;
                if (top.node.left != null) {
                    Pair c = new pair (top.node.left, 1)
                    st.push(c)
                }
            else (top.state == 2)
```

```
        top.state ++;
        print ( top.node.val );
        if (top.node.right != null){
            Pair c = new Pair( top.node.right, 1);
            st.push(c);
        }
    }
    else (top.state == 3)
    {
        st.pop();
    }
}
```

TC : O(n)

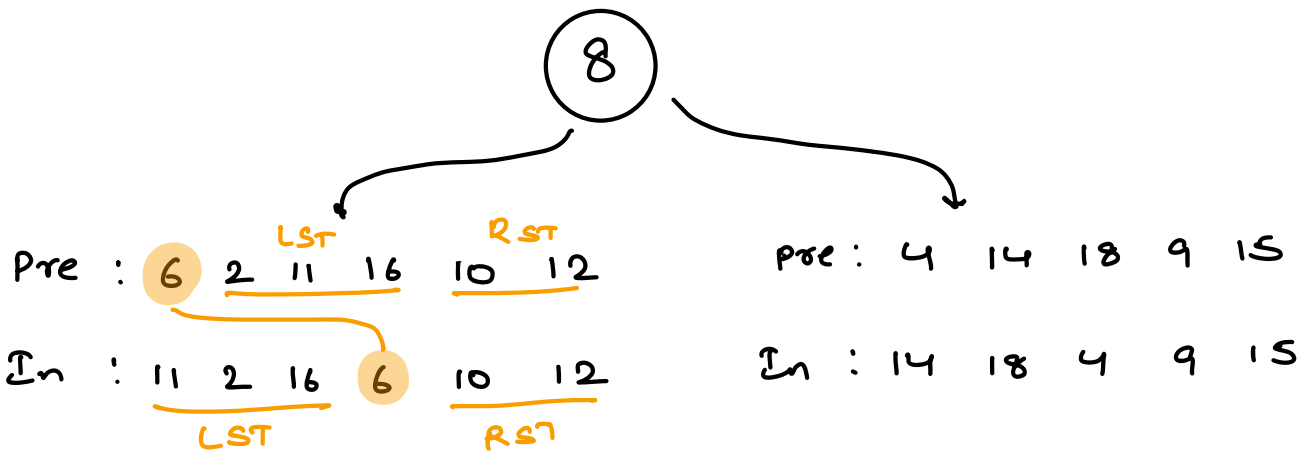
SC : O(h)

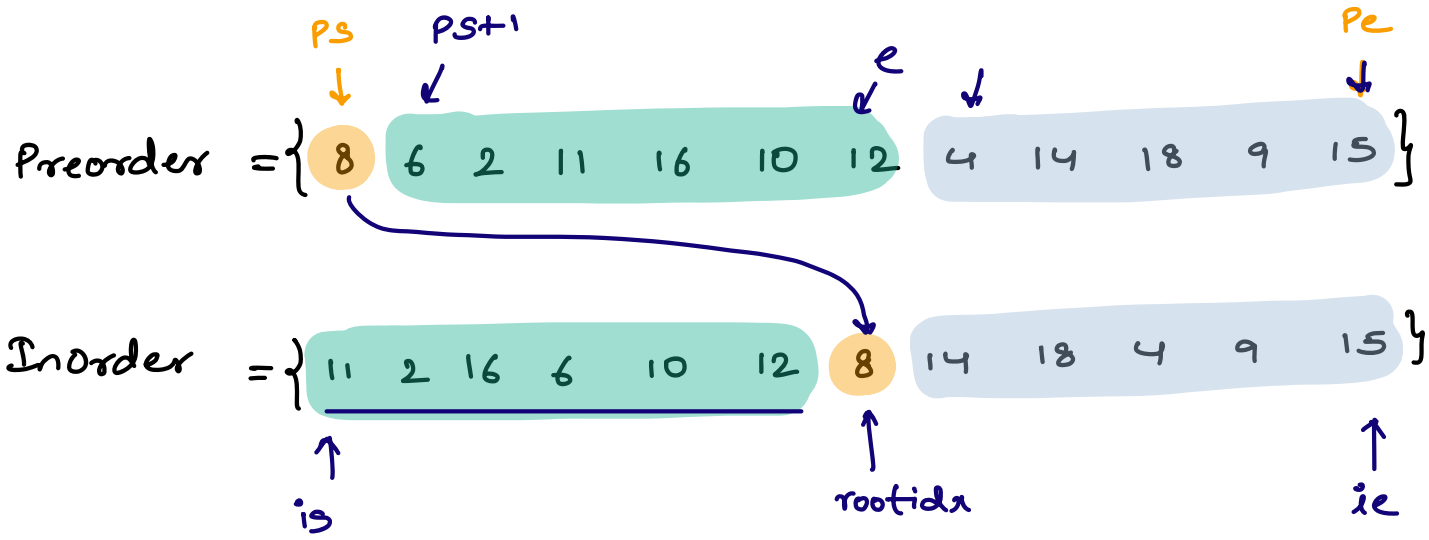
Q Given preorder[] & inorder[] of Binary Tree with distinct values. Construct BT



Preorder = { 8 | 6 2 11 16 10 12 | 4 14 18 9 15 }
LST RST

Inorder = { 11 2 16 6 10 12 | 8 | 14 18 4 9 15 }
LST RST






```
Node create ( int[] pre , ps, pe , int[] in, is, ie)

    if (ps > pe) return null;

      int root data = pre[ps]

    Node root = new Node (root.data)

    int rootidx = find ( in[], is, ie, rootdata)

    int elementsLST = rootidx - is

  root.left = create ( pre, ps+1 , ps+elementsLST , in , is,
                                                    rootidx-1 )

  root.right = create (pre, ps+elemLST+1 , pe , in ,
                                          rootidx+1 , ie )

return root;
```

$a \quad b = b - a + 1$

elements in LST $(is \quad rootidx - 1)$

Elements LST = $rootidx - 1 - is + 1$

Elements LST = $rootidx - is$

(a) (b)

$(ps+1 \quad e)$ = elements in LST

$e - (ps+1) + 1$

$e - ps - 1 + 1$ = elements in LST

e = elements LST + ps