Trees 1 : Structure & Traversal

TODAY'S CONTENT

- Binary Tree (BT)
- Traversal
- Iterative traversal
- construct BT from pre-order & In-order

DSA Certification → (DSA) → Company

↓

Mock Interview of DSA → After class class / 30 days

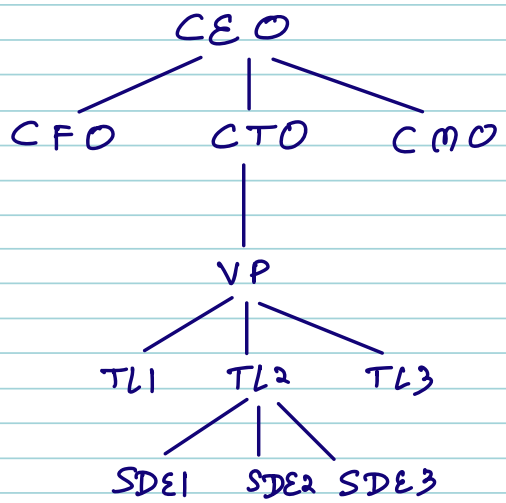
+

contest of DSA

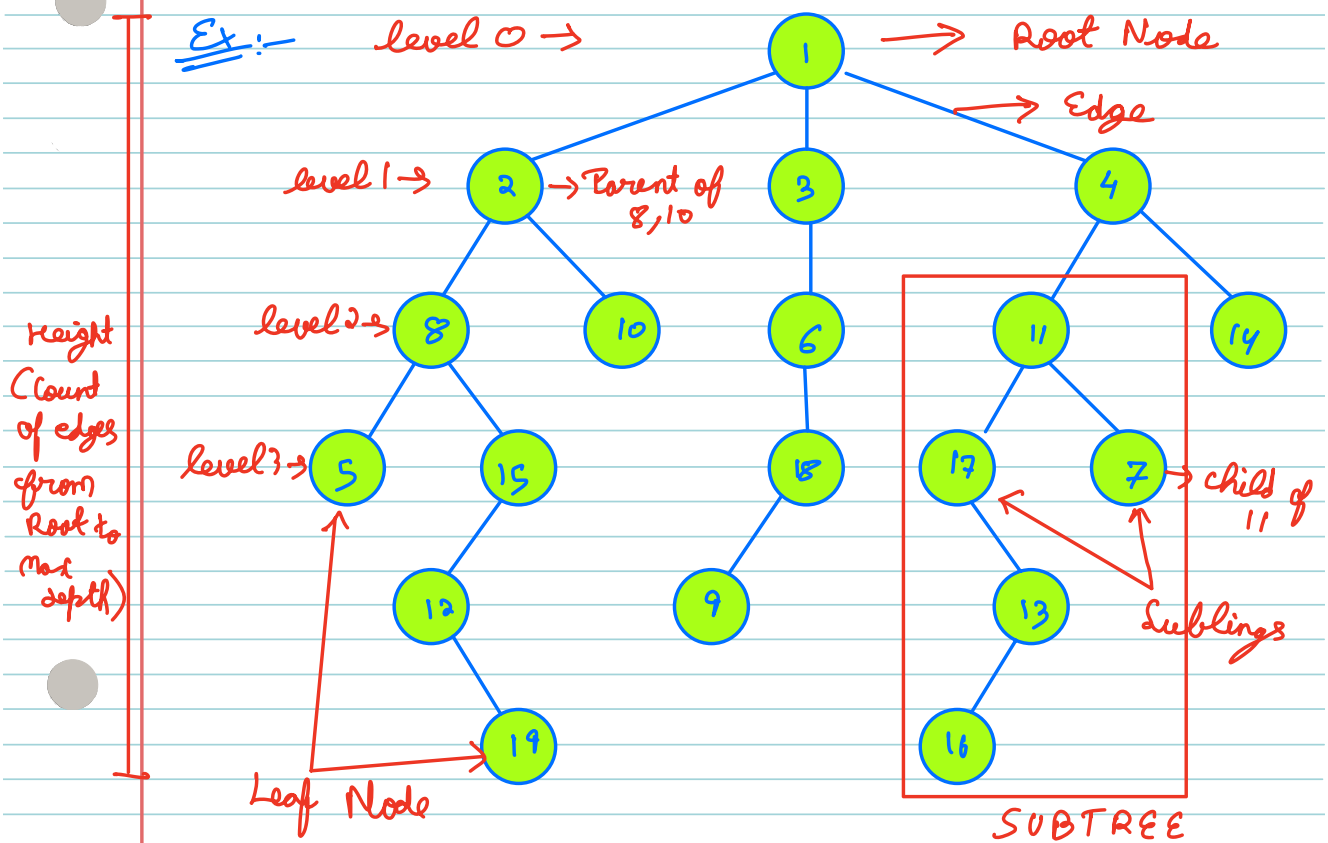
Trees Basics & Terminologies

// Linear data Structure → Arrays, Stacks, LL.

// Hierarchical data Structure



↳ To Store this kind of Data tree is there.



Root → It is a node with No parent.

Leaf → It is a node with No child.

Quiz1 :- Can leaf Node also be a Subtree?

↳ yes

Quiz2 :- Do all nodes have a parent Node?

↳ No

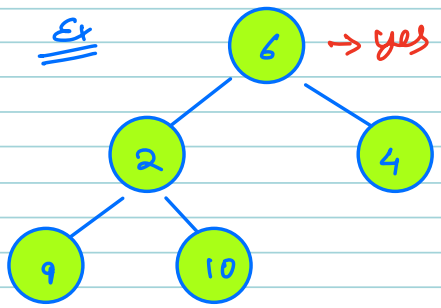
Quiz3 :- what is height of the Leaf Node in any Tree?

↳ 0

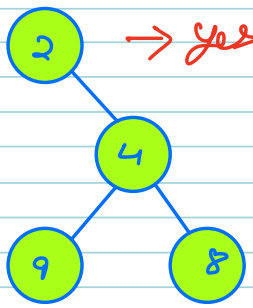
BINARY TREE (B.T.)

↳ For each Node, No of children <=2

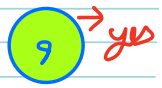
Ex



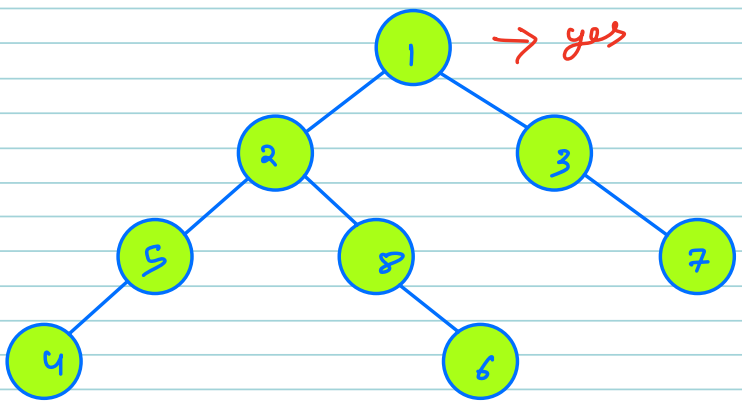
→ yes



→ yes



→ yes



→ yes

⇒ STRUCTURE :- At each Node, we need to store data & two child.

```
class Node {
    int data
    Node left ;
    Node Right;
    Public Node ( int x ) {
        data = x;
        Left = Null;  Right = null;
    }
}
```

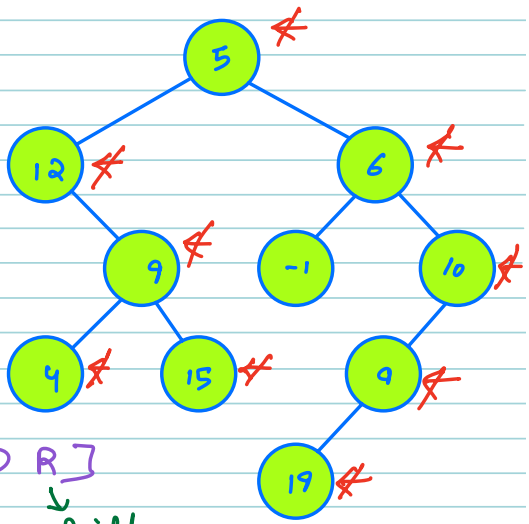
3

TREE TRAVERSAL

→ Inorder

→ Pre-order

→ Post-order



1:> INORDER

↳ Traversal order :- [L D R]

Data

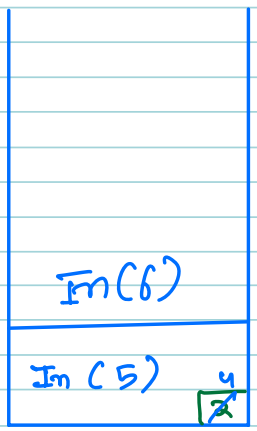
Left

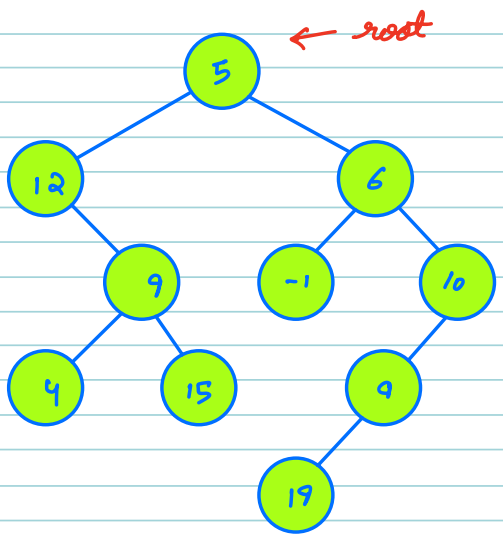
Right

output :- 12 , 4 , 9 , 15 , 5 , -1 , 6 , 19 , 9 , 10

PSEUDO CODE

```
void Inorder ( Node root) {
1   if ( root == Null) { return ;}
2   Inorder ( root. Left);
3   Print (root . data);
4   Inorder ( root . right);
}
```





output = 12, 4, 9, 15, 5, ...

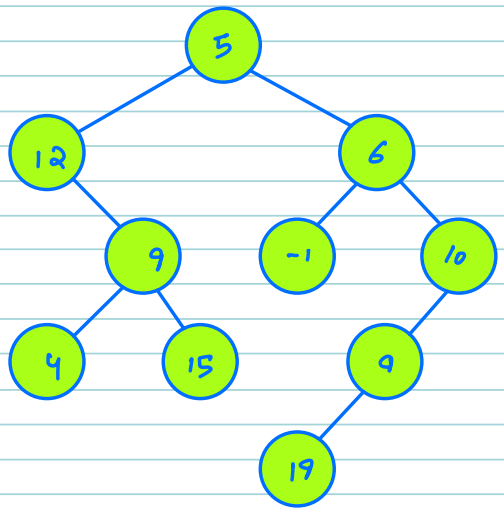
TC — O(N)
SC — O(H)

→ we are touching each node thrice

SC → O(H)

- → O(N) [worst case] → Skewed tree
- → O(logN) [Best case] → Complete Tree [each Node have 2 childs]

2.> PRE-ORDER [D L R]



output:- 5, 12, 9, 4, 15, 6, -1, 10, 9, 19

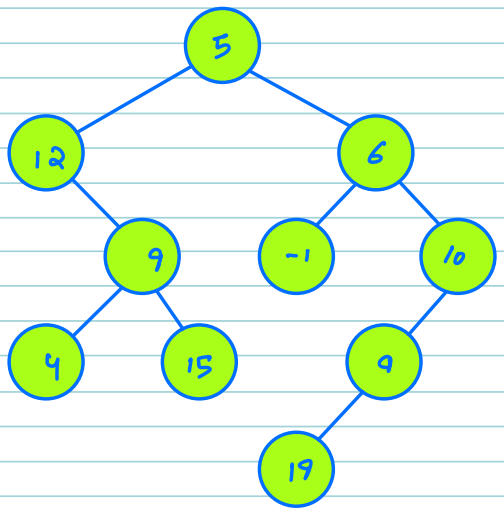
PSEUDO CODE

```
void PreOrder ( Node root) {
1   if ( root == Null) { return; }
2   Print (root.data);
3   PreOrder ( root.Left);
4   PreOrder ( root.right);
}
```

TC → O(N)
SC → O(H)

↳ Recursive Dry Run as H.W.

3> POST ORDER (LRD)



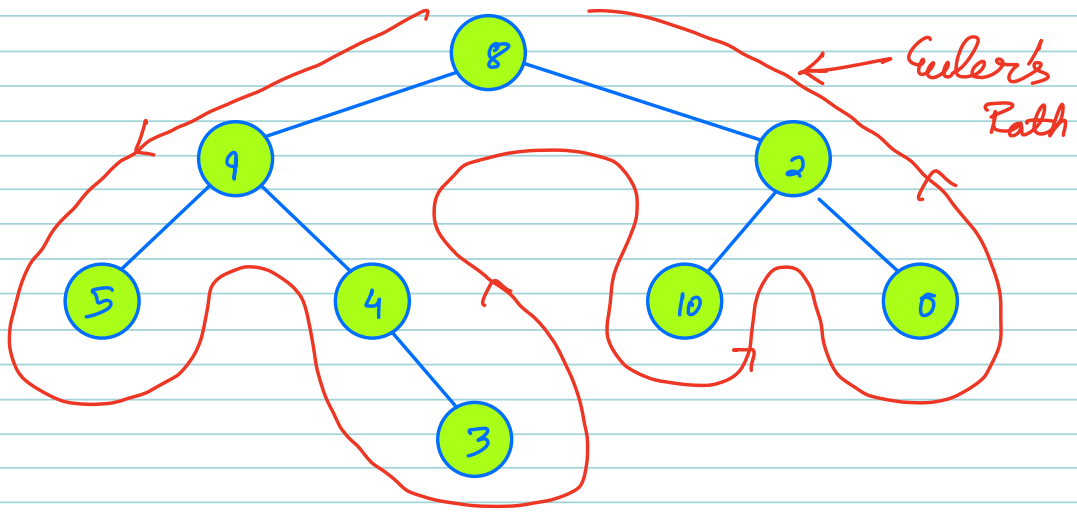
OUTPUT:- 4, 15, 9, 12, -1
19, 9, 10, 6, 5

PSEUDO CODE

```
void PostOrder (Node root) {
1   if (root == Null) { return; }
2   PostOrder (root.Left);
3   PostOrder (root.right);
4   Print (root.data);
}
```

TC — O(N)
SC — O(H)

EULER'S PATH



↳ Magical way of doing travels we discussed above.

INORDER :- 5, 9, 4, 3, 8, 10, 2, 0

↑ Print when you are touching Node as middle

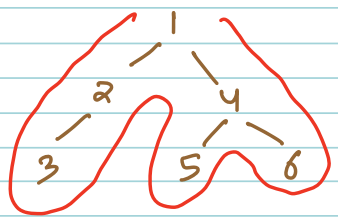
Pre Order :- 8, 9, 5, 4, 3, 2, 10, 0

↑ Print when you are going toward the Node

POST Order = 5, 3, 4, 9, 10, 0, 2, 8

↑ Print when moving away from Node.

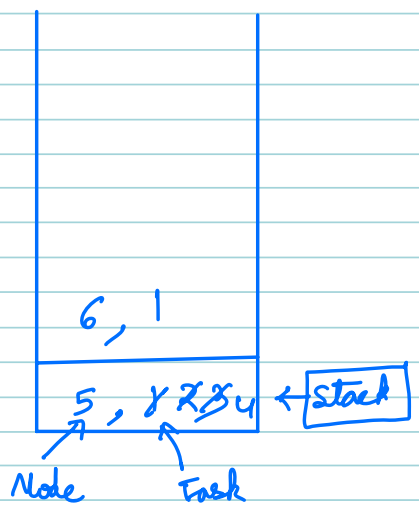
Quiz 4 :- what is the inorder traversal sequence of the below tree ?

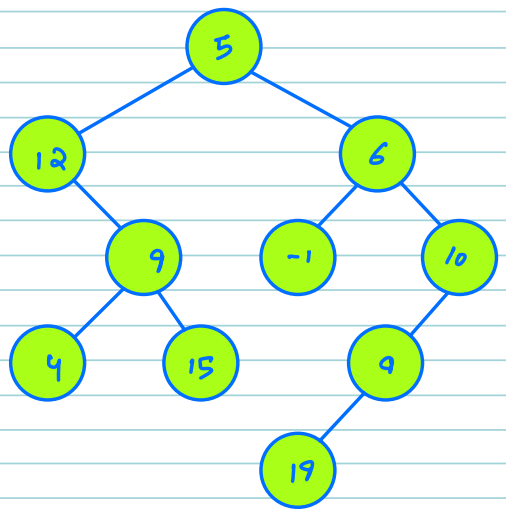


⇒ Inorder = 3, 2, 1, 5, 4, 6,

INORDER ITERATIVE (LDR)

Tasks :- ① Call left child
② Print data
③ Call Right child.





output :- 12, 4, 9, 15, 5

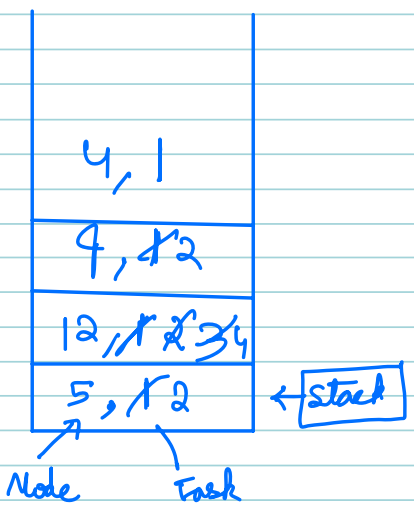
* when Task = 1, then Insert Left child of it & Increament task
* when Task = 2, then Print & Increament task
* when Task = 3, then insert Right Node & Increament task
* when Task = 4, Simply Pop from Stack.

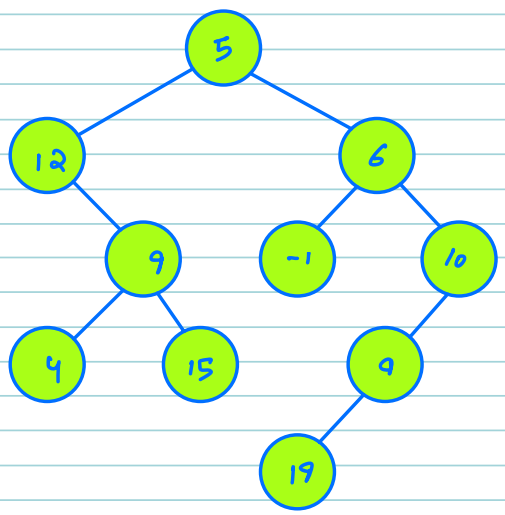
PSEUDO CODE

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

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()
}
}
```





output = 12, ...

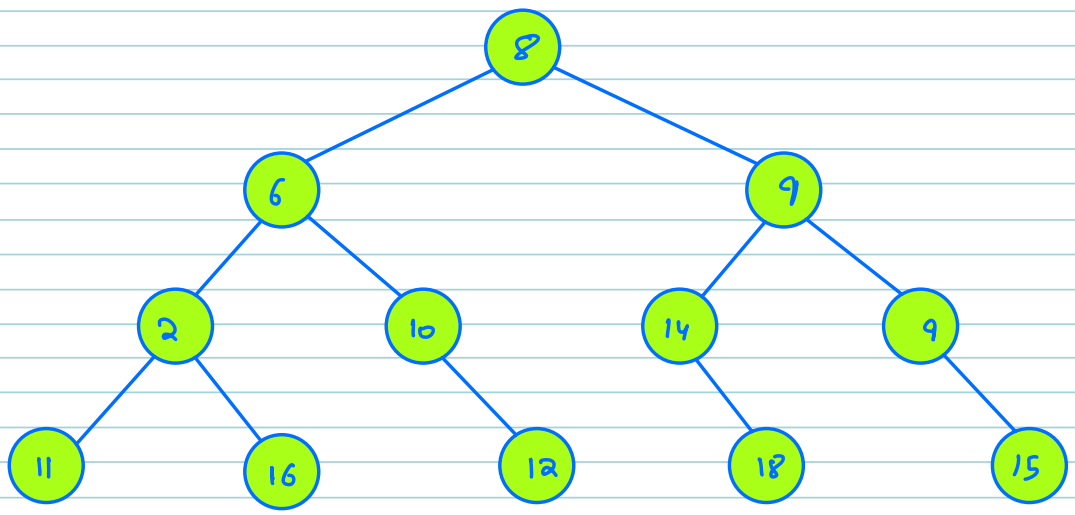
TC - O(N)

SC - O(H)

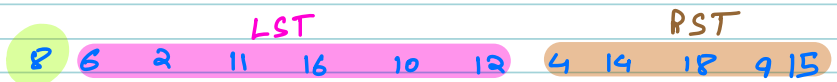
* Same solution can be modified for Pre order & Post order. Try as H.W.

Question :- Construct a binary Tree from its inorder & Pre order.

Note :- Elements will be having distinct values.



Pre Order :- (DLR)



Inorder :- (LDR)

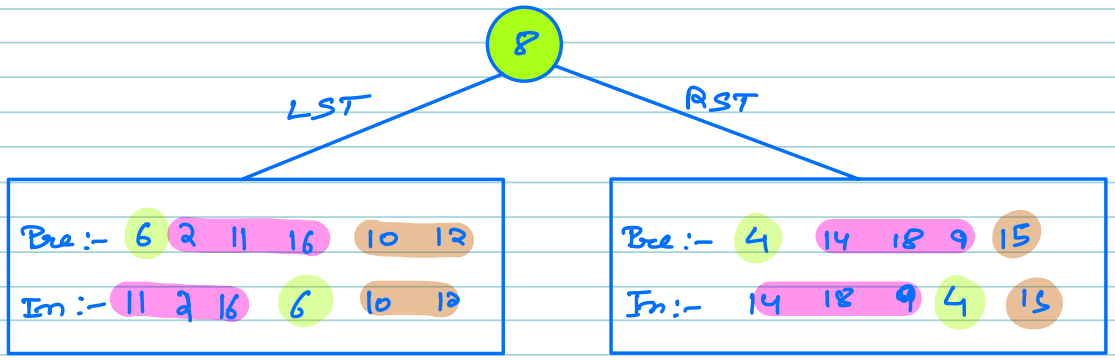
Q Can we say first Node of Pre-order is root of Tree

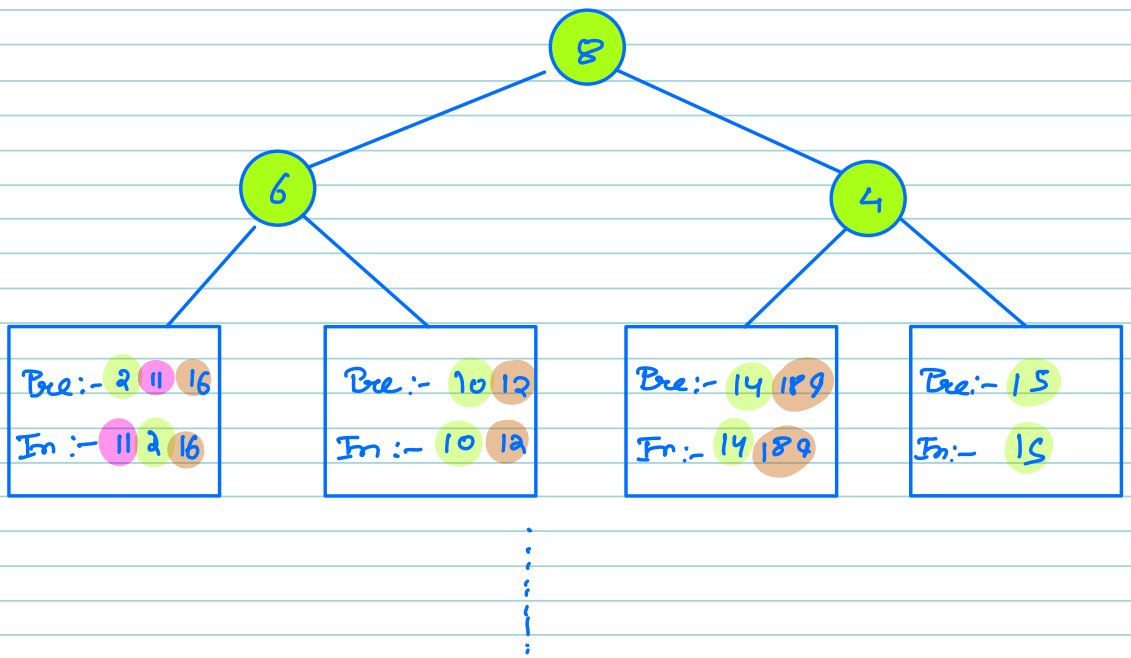
↳ yes

Q Can we say Nos. on Left of 8 in Inorder will be Left Sub Tree

↳ yes

* Let Count of element on Left in Inorder be C, then we can say C elements ahead of 8 in pre-order is Left





⇒ This is nothing just Recursion.

Q How we will split?

↳ Math to Split.

is ie

PSEUDO CODE

```
Node Create ( int Ps, int Pe, int is, int ie ) {

    if ( Ps > Pe || is > ie ) {
        return Null;
    }
    root data = Pre [Ps];
    Node root = new Node (root data);
    int rootIdx = find ( is, ie, root data);
                  ↳ map ( element Vs index) can be used.
    int elementinLST = rootIdx - is;
    root.Left = Create ( Ps+1, (LST) , is, rootIdx-1);
                                ↳ element in LST + Ps
    root.Right = Create ( (RST) , Pe, rootIdx+1, ie);
                                ↳ element in LST + Ps + 1
    return root;
}
```

CLDR) Left Sub tree CLST Right Sub tree (RST)

is rootIdx ie

Maths

① element in CST = [is , rootIdx − 1]

= rootIdx − is

= rootIdx − is

② element in LST of Pre order

[PS + 1 , x) = elements in LST

⇒ x − PS = element in LST

⇒ x = element in LST + PS

Quiz 5 :- The inorder traversal sequence [4, 2, 5, 1, 6, 3] and the post order traversal sequence [4, 5, 2, 6, 3, 1]. what is the root of the binary tree ?

↳ 1

POST ORDER [LRD]