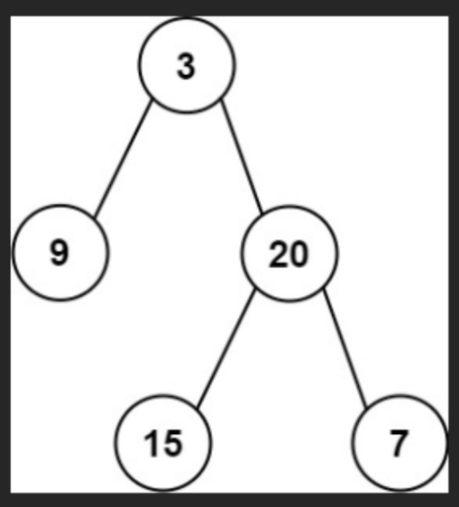
106. Construct Binary Tree from Inorder and Postorder Traversal

Medium | Topics | Companies

Given two integer arrays `inorder` and `postorder` where `inorder` is the inorder traversal of a binary tree and `postorder` is the postorder traversal of the same tree, construct and return *the binary tree*.

Example 1:




```
Input: inorder = [9,3,15,20,7], postorder = [9,15,7,20,3]
Output: [3,9,20,null,null,15,7]
```

Example 2:

```
Input: inorder = [-1], postorder = [-1]
Output: [-1]
```

inorder :

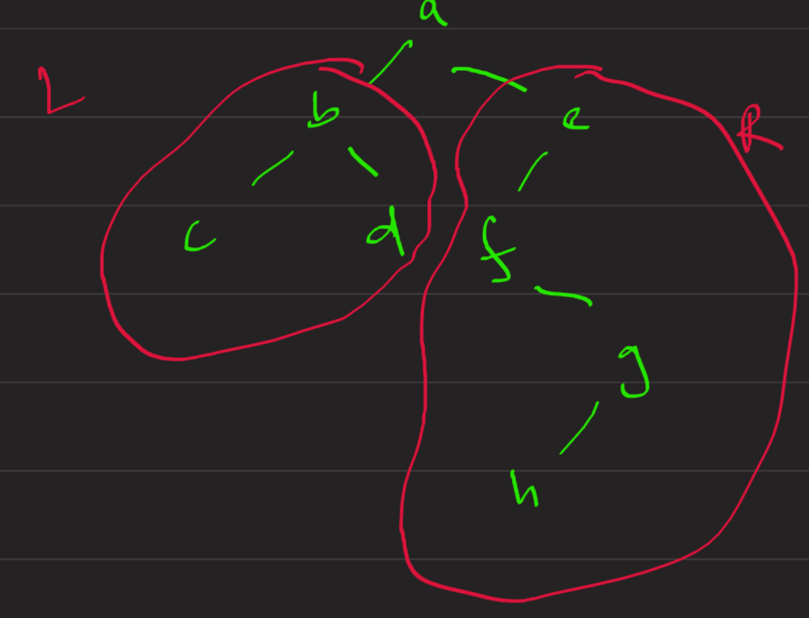
(9, 3, 15, 20, 7)

(L, Root, R)

(9, 15, 7, 20, 3)

(L, R, Root)

* whenever you see postOrder, the last element will be our root...



→ first we traverse (L)

next (R)

then f root → (a)

So, a will come at last

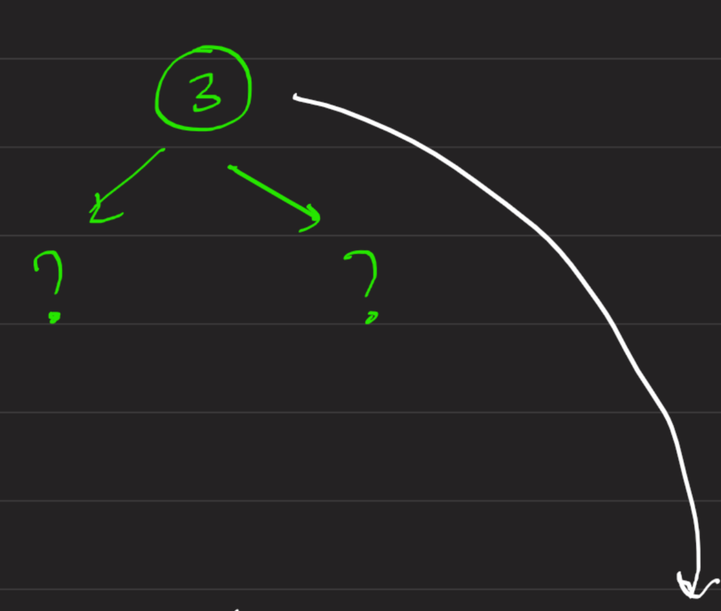
a is our root

InStart ↓ (9, 3, 15, 20, 7) ↓ InEnd

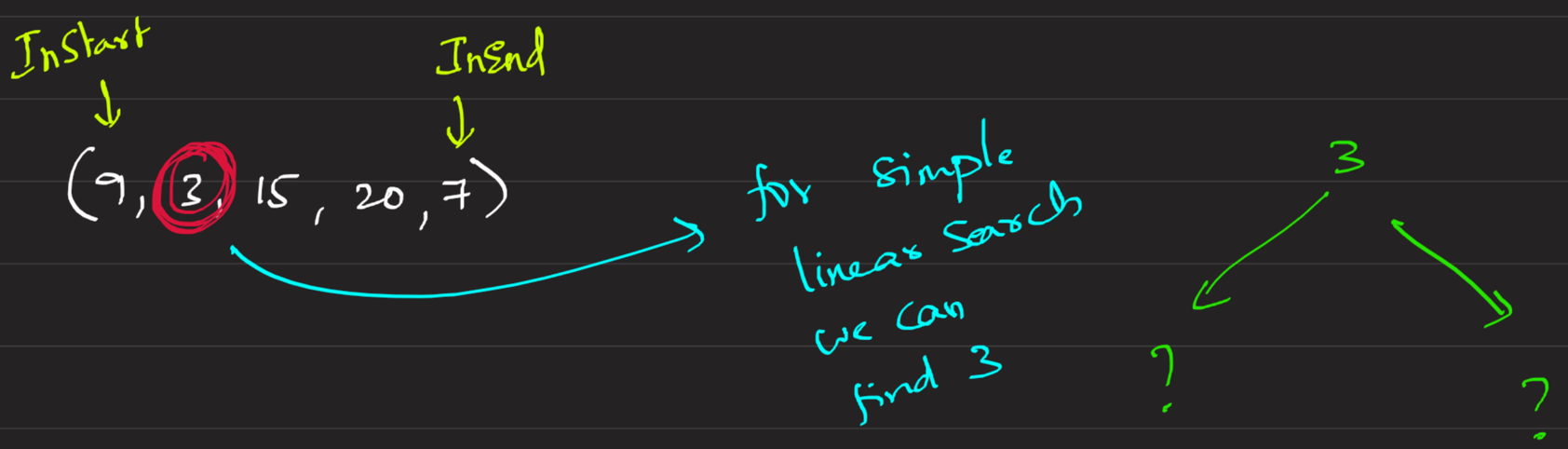
* root = postOrder[postEnd]

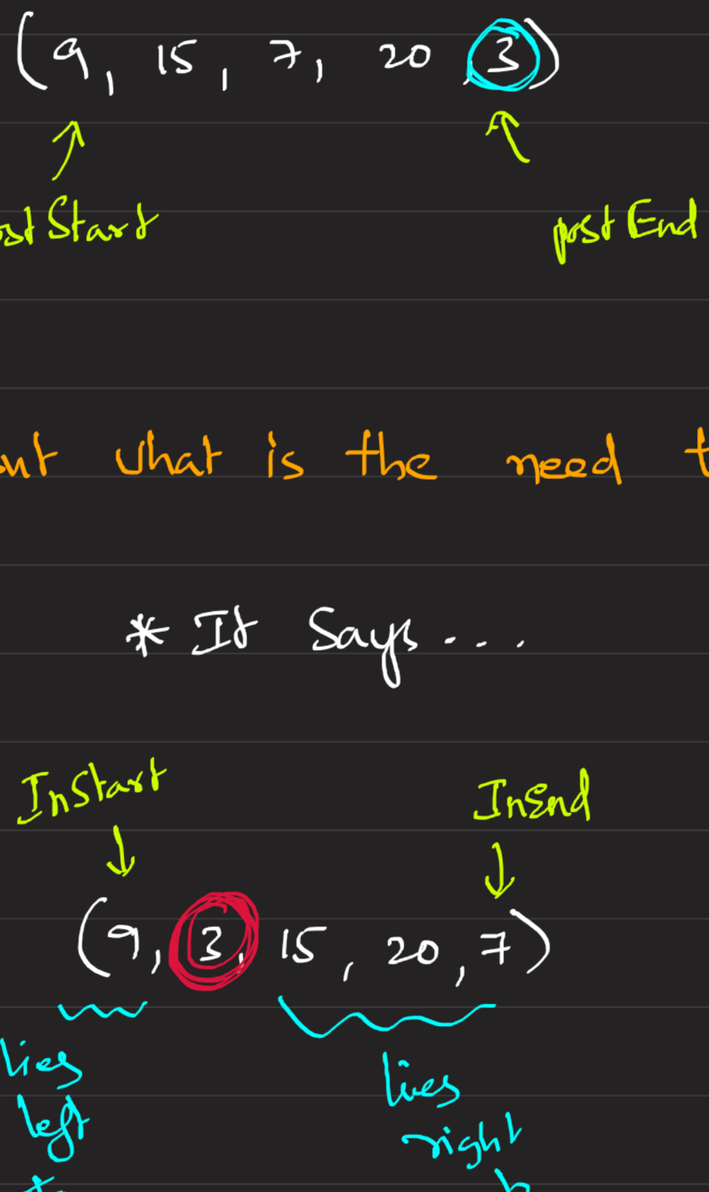
(9, 15, 7, 20, 3)

postStart ↑ postEnd ↑



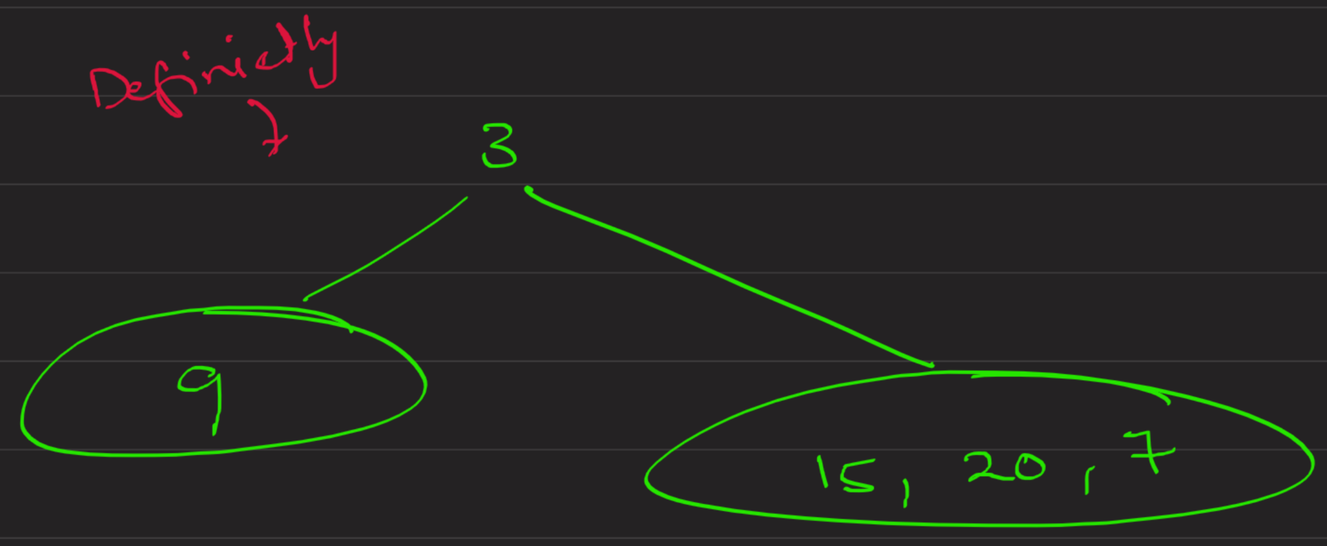
* Now Question ourselves than, wher did this (3) located in inorder....





But what is the need to find (3) in inorder...

* It Says...

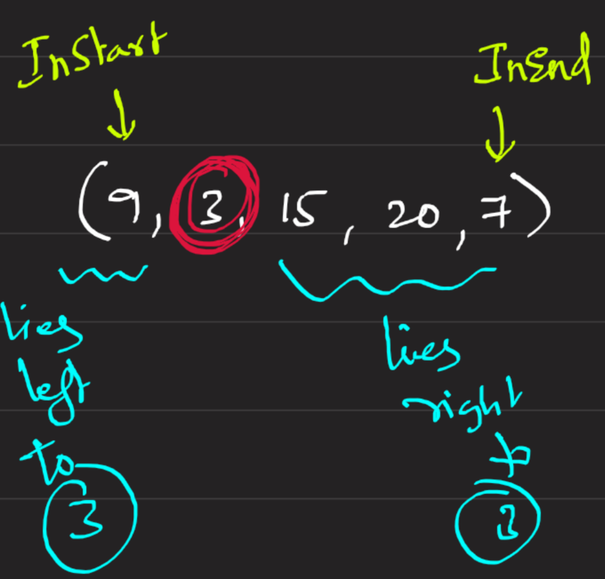


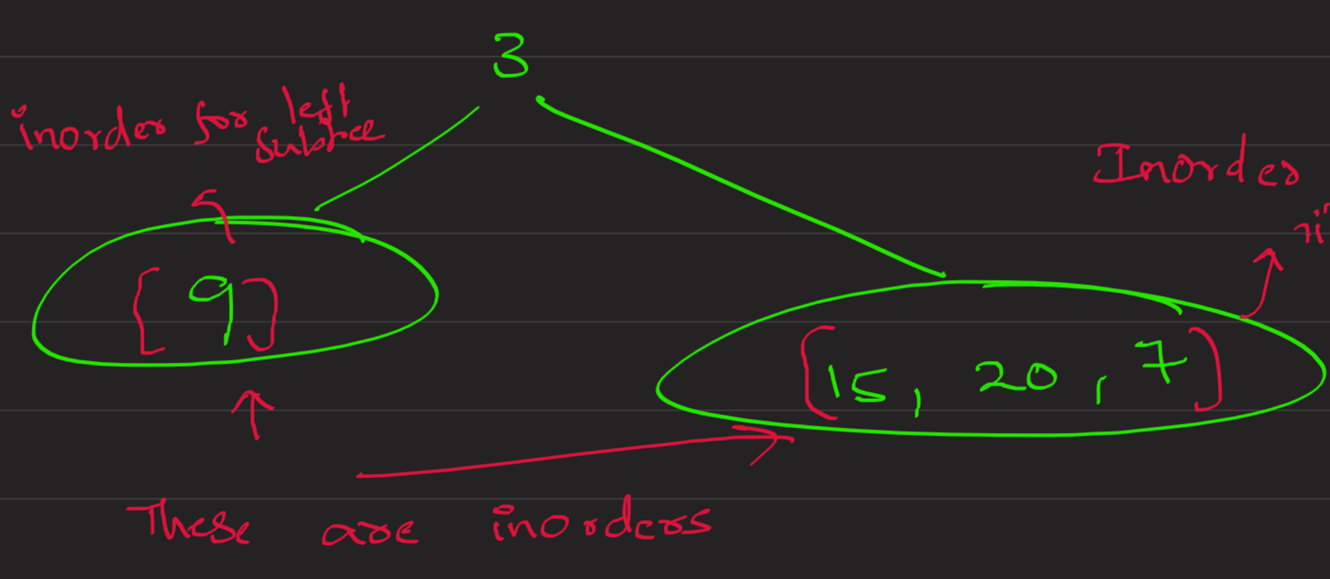
* A key point to remember here is,

We are solving this in recursion ⟶ so like we solved for 1st level we needed both inOrder & postOrder to do that

Similarly one every level we need both inOrders & postOrders

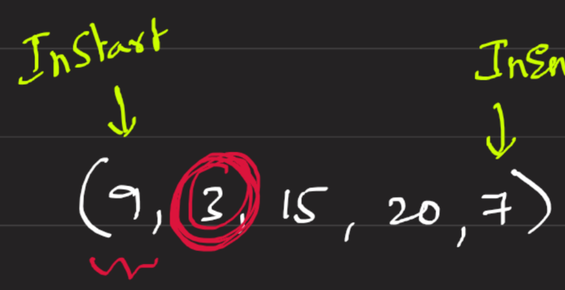
* whatever we have splitted above ↴



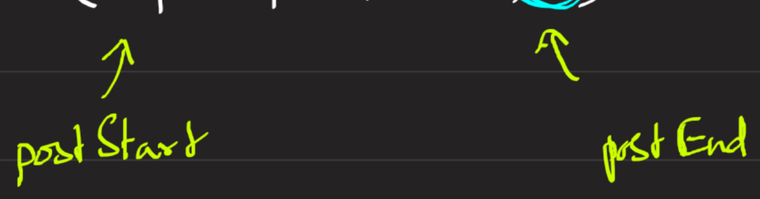


similarly, we need postOrder's to pass in our recursion

let's observe

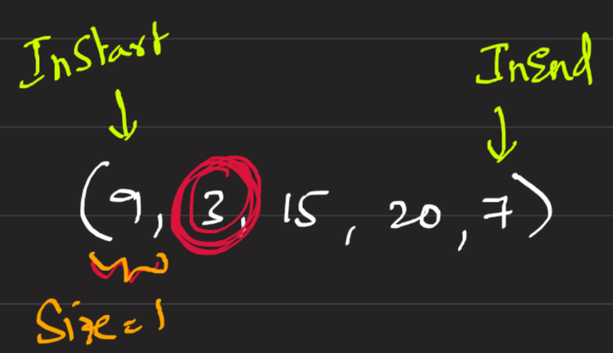


Inorder for left : [9]
right : [15, 20, 7]



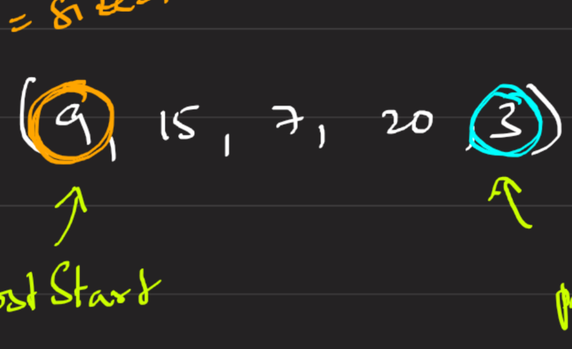
what will be postOrder for left = ?
right = ?

* Count sizes of left tree & right trees of InOrder... add that many nodes to our postOrder...



Inorder for left : [9]
right : [15, 20, 7]

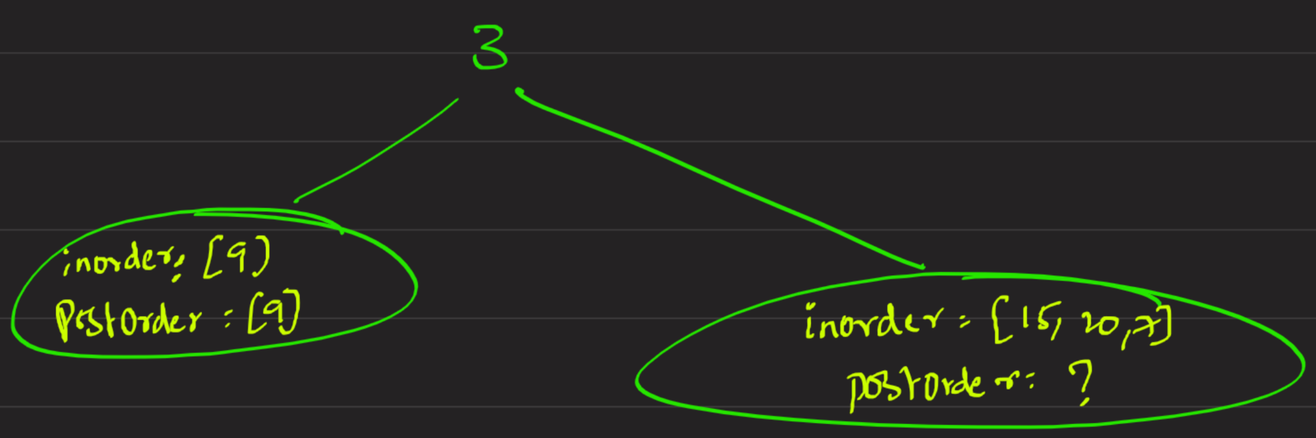
Count = size=1

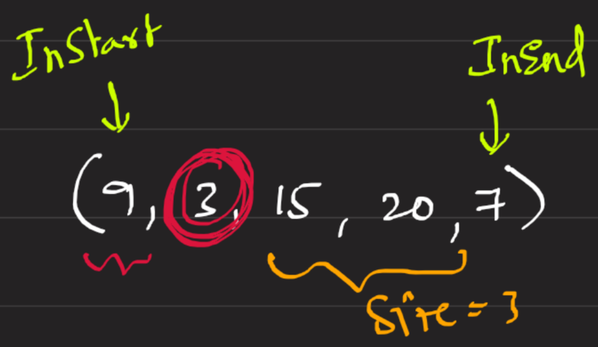


for postOrder's left array :

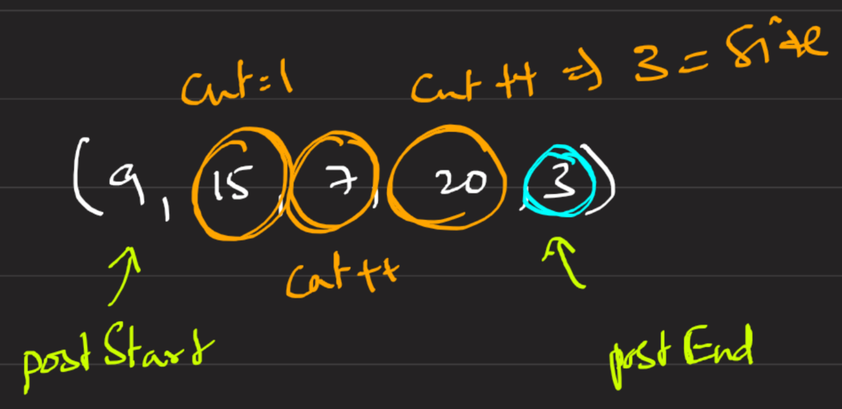
① take inOrder's (left) : [9] → size = 1

take 1 node in postOrder & pass it to recursion as (left) subtree





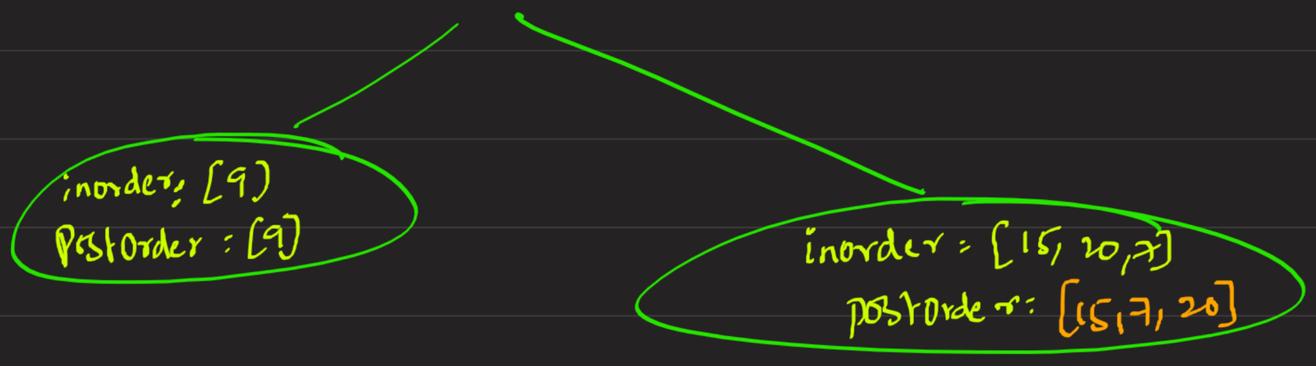
Inorder for left : [9]
right : [15, 20, 7]



for postOrder's right array :

① take inOrders right : [15,20,7] ⇒ Size = 3

take 3 nodes in postOrder & pass it to recursion as (right subtree)

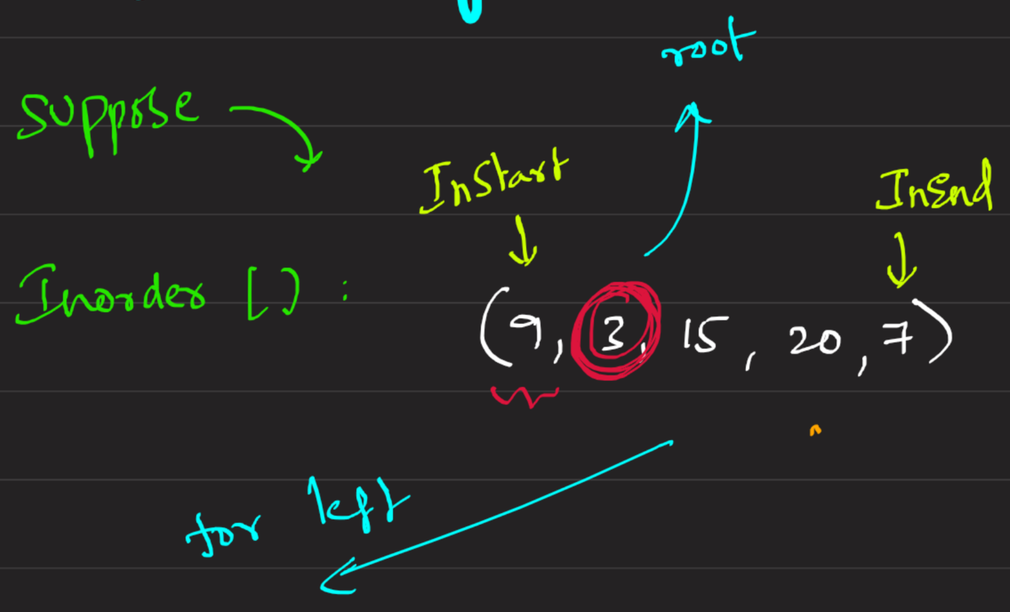


Now, solve these subproblem recursively....

* The main question here is, Ok, i understood we need to partition the given array according to our needs, to solve the inorder, postOrder

∴ But how to partition

Simply... By indexing



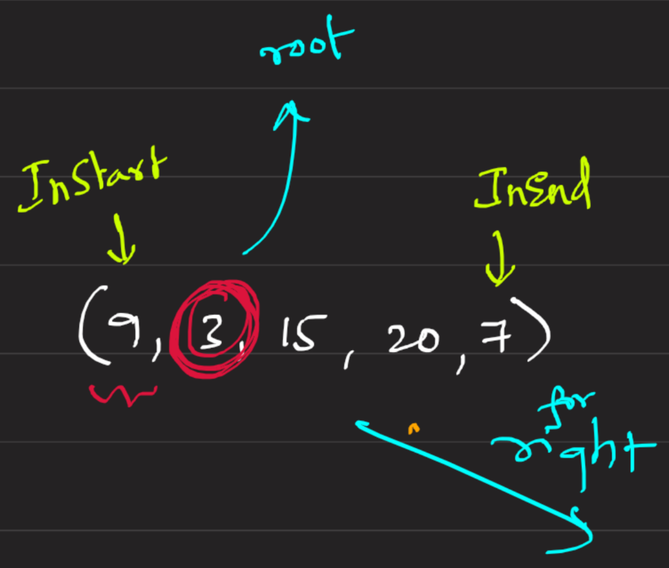
InStart → stays at InStart

InEnd → Becomes the node just behind root (if root's index is (i))

InEnd = i - 1

Suppose →

Inorder [ ] :

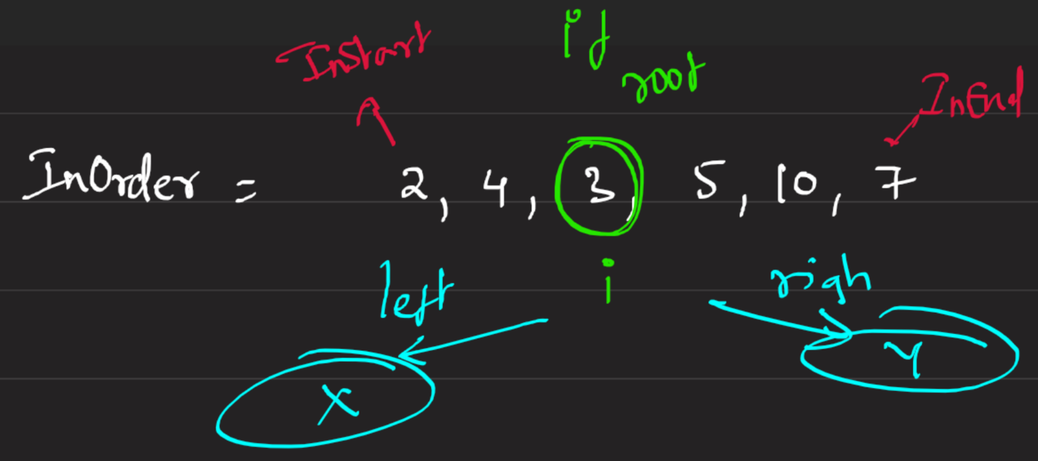


InStart → Becomes the node just after root (if root's index is (i))

InStart = i + 1

IEnd → Stays at InEnd

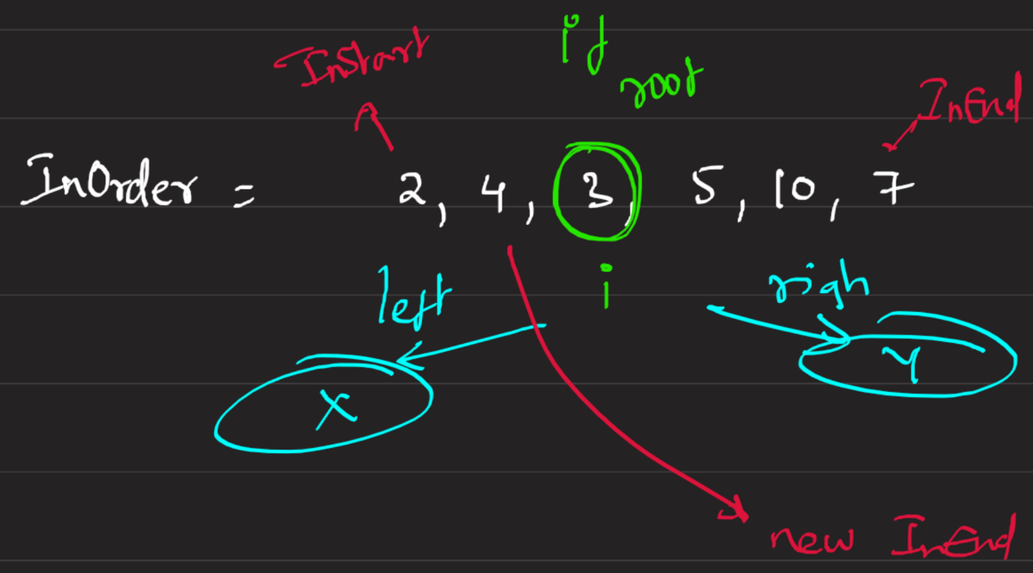
```java
class Solution {
    public TreeNode solve(int[] inorder,int postorder[],int inStart,int inEnd,int postStart,int postEnd){
        //we know that our root will be last index value in our postorder in every recursive call
        if(inStart>inEnd)return null;
        TreeNode root = new TreeNode(postorder[postEnd]);

        //traverse in inorder and check at which index, our root present in inorder
        int i;
        for( i = inStart;i<=inEnd;i++){
            if(inorder[i] == root.val)break;
        }
        //Now i is at the root node for this current level...
        //we keep trust in recursion that answer for my self....i solved

        //now recursion we give me answers for ledt subtree and right subtree

        //left-subtree answer

        //to do that...we need size of left inorder tree and right as well
        int leftSize = i-inStart;
        int rightSize = inEnd-i;
        root.left = solve(inorder,postorder,inStart,i-1,postStart,postStart+leftSize-1);
        root.right = solve(inorder,postorder,i+1,inEnd,postStart+leftSize,postEnd-1);
        return root;
    }
    public TreeNode buildTree(int[] inorder, int[] postorder) {
        return solve(inorder,postorder,0,inorder.length-1,0,postorder.length-1);
    }
}
```




```java
root.left = solve(inorder,postorder,inStart,i-1,postStart,postStart+leftSize-1);
root.right = solve(inorder,postorder,i+1,inEnd,postStart+leftSize,postEnd-1);
```

to Solve X, we need inOrder array & preOrder array

inorder will be left half of (3)



InStart of left = InStart

InEnd of left = i − 1

InStart of (right) = i + 1

InEnd of right = InEnd

Similarly, to find PostOrder of left & right

* We need to count no of nodes present in left inorder

& count same no. of nodes from 0th index in postOrder &

pass it as recursion

Similarly in right as well

If leftSize of inOrder left tree = 1

rightSize = 3