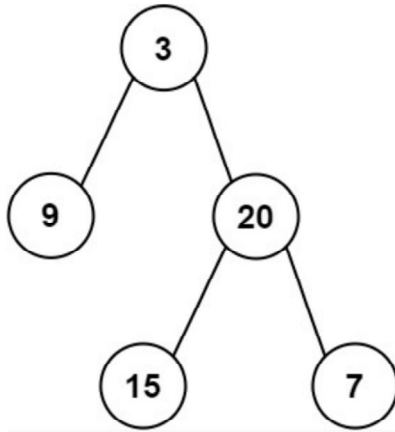


106. Construct Binary Tree from In order and Post order Traversal

21 November 2021 11:06 AM

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:



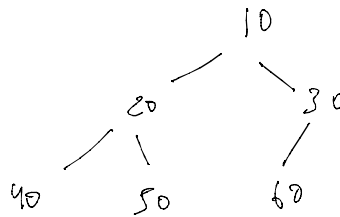
Input: `inorder = [9,3,15,20,7]`, `postorder = [9,15,7,20,3]`
Output: `[3,9,20,null,null,15,7]`

Example 2:

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

Inorder $\rightarrow [40, 20, 50, 10, 60, 30]$

Post Order $\rightarrow [40, 50, 20, 60, 30, 10]$

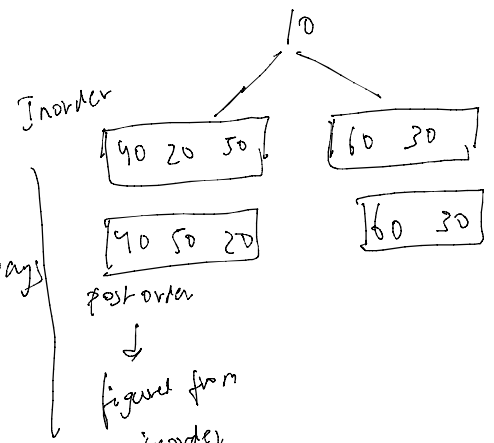


Inorder $\rightarrow [40, 20, 50, (10), 60, 30]$

(left Root Right)

Post Order $\rightarrow [40, 50, 20, 60, 30, (10)]$

(left Right Root)



Left Right Root post order of L post order of R

figure from
inorder

L ← R → R
10 40 (20) 50
10 40 50 (20)
R

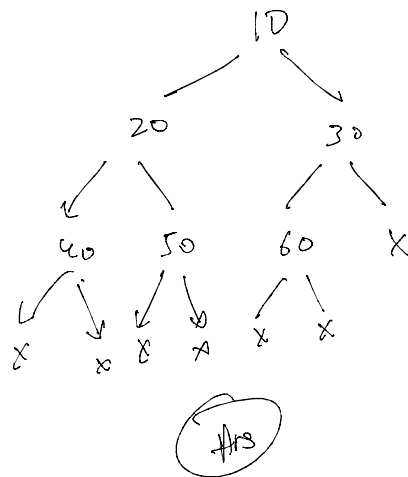
10 40 50 PD
10 40 50 PD

10 40 (40) R
L R, Root

10 (40) R
L R, Root

10 (50) Root

10 (50) Root



L ← R → Right
10 30 10

10 (30) PD
R Root

10 (50) → L

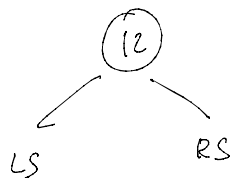
10 (60) Root

Code

Inorder → [X] (12) [Y]

post order → [X] [Y] (12)
P.O of left subtree P.O of right subtree Root

map



```
class Solution {
    public TreeNode buildTree(int[] inorder, int[] postorder) {
        if (inorder == null || postorder == null || inorder.length != postorder.length)
            return null;
        HashMap<Integer, Integer> hm = new HashMap<Integer, Integer>();
        for (int i=0; i<inorder.length; ++i)
            hm.put(inorder[i], i);
        return buildTreePostIn(inorder, 0, inorder.length-1, postorder, 0,
                               postorder.length-1, hm);
    }
}
```

```
private TreeNode buildTreePostIn(int[] inorder, int is, int ie, int[] postorder, int ps, int pe,
                                  HashMap<Integer, Integer> hm){
    if (ps > pe || is > ie) return null;
    TreeNode root = new TreeNode(postorder[pe]);
    int ri = hm.get(postorder[pe]);
    TreeNode leftchild = buildTreePostIn(inorder, is, ri-1, postorder, ps, ps+ri-is-1, hm);
    TreeNode rightchild = buildTreePostIn(inorder, ri+1, ie, postorder, ps+ri-is, pe-1, hm);
    root.left = leftchild;
    root.right = rightchild;
    return root;
}
```