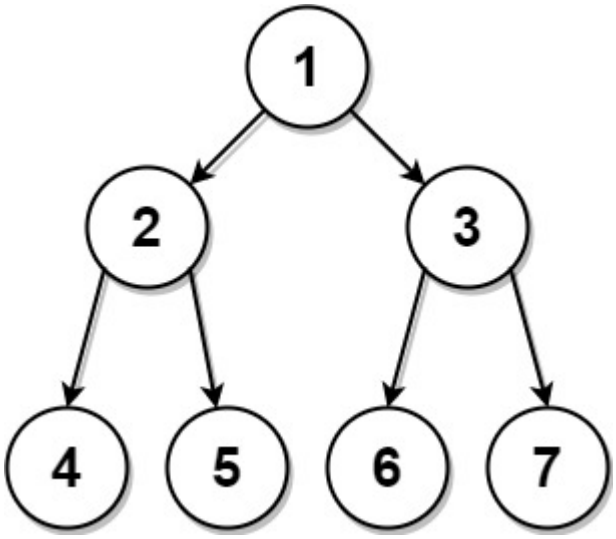
889. Construct Binary Tree from Preorder and Postorder Traversal

Given two integer arrays, preorder and postorder where preorder is the preorder traversal of a binary tree of **distinct** values and postorder is the postorder traversal of the same tree, reconstruct and return *the binary tree*.

If there exist multiple answers, you can **return any** of them.

Example 1:



Input: preorder = [1,2,4,5,3,6,7], postorder = [4,5,2,6,7,3,1]

Output: [1,2,3,4,5,6,7]

Example 2:

Input: preorder = [1], postorder = [1]

Output: [1]

Constraints:

- 1 <= preorder.length <= 30
- 1 <= preorder[i] <= preorder.length
- All the values of preorder are **unique**.
- postorder.length == preorder.length

- 1 <= postorder[i] <= postorder.length
- All the values of postorder are **unique**.
- It is guaranteed that preorder and postorder are the preorder traversal and postorder traversal of the same binary tree.

```
# Definition for a binary tree node.
# class TreeNode(object):
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution(object):
    def constructFromPrePost(self, preorder, postorder):
        """
        :type preorder: List[int]
        :type postorder: List[int]
        :rtype: TreeNode
        """
        if len(preorder) == 0:
            return None
        root = TreeNode(preorder[0])
        stack, i = [root], 0

        for n in preorder[1:]:
            if stack[-1].val != postorder[i]:
                stack[-1].left = left = TreeNode(n)
                stack.append(left)
            else:
                while stack and stack[-1].val ==
postorder[i]:
                    cur = stack.pop()
                    i+=1
                stack[-1].right = right = TreeNode(n)
                stack.append(right)

        return root
```