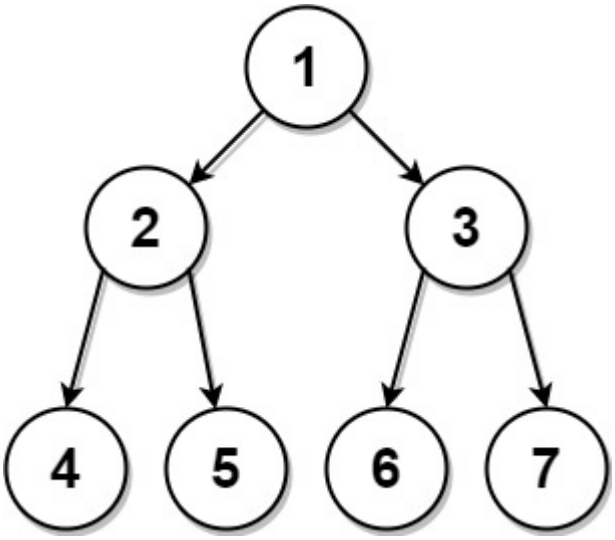
889. Construct Binary Tree from Preorder and Postorder Traversal

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.

```
class Solution:
    def constructFromPrePost(self, preorder: List[int], postorder:
List[int]) -> Optional[TreeNode]:

        return self.constructFromPerPostHelper(preorder,postorder)


    def constructFromPerPostHelper(self,preorder,postorder):
        if len(preorder)==0:
            return None
        if len(preorder)==1:
            return TreeNode(preorder[0])


        node = TreeNode(preorder[0])
        item  = preorder[1]
        idx = postorder.index(item)
        preleft = preorder[1:idx+2]
        preright = preorder[idx+2:]
        postleft = postorder[:idx+1]
        postright = postorder[idx+1:-1]
        node.left = self.constructFromPerPostHelper(preleft,postleft)
        node.right = self.constructFromPerPostHelper(preright,postright)
        return node
```