```java
/* Given inorder and postorder traversal of a tree, construct the binary tree.

Note:
You may assume that duplicates do not exist in the tree.

For example, given

inorder = [9,3,15,20,7]
postorder = [9,15,7,20,3]
Return the following binary tree:

    3
   / \
  9  20
    /  \
   15   7 */

class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode(int x) {
        val = x;
    }
}


class Solution {
    public TreeNode buildTree(int[] inorder, int[] postorder) {

        if (inorder == null || postorder == null
                || inorder.length == 0 || postorder.length == 0) {
            return null;
        }

        return buildTreeHelper(inorder, 0, inorder.length - 1, postorder, 
        postorder.length - 1);
    }

    private TreeNode buildTreeHelper(int[] inorder, int startIndexOfInOrder, int 
    endIndexOfInOrder,
                                     int[] postorder, int rootIndexOfPostOrder) {

        if (startIndexOfInOrder > endIndexOfInOrder) {
            return null;
        }

        TreeNode root = new TreeNode(postorder[rootIndexOfPostOrder]);

        int rootIndexOfInOrder = findRootIndexOfInOrder(inorder, 
        startIndexOfInOrder, endIndexOfInOrder,
                postorder[rootIndexOfPostOrder]);

        root.left = buildTreeHelper(inorder, startIndexOfInOrder, rootIndexOfInOrder 
        - 1,
                postorder, rootIndexOfPostOrder - 1 - (endIndexOfInOrder - 
                rootIndexOfInOrder));
        root.right = buildTreeHelper(inorder, rootIndexOfInOrder + 1, 
        endIndexOfInOrder,
                postorder, rootIndexOfPostOrder - 1);

        return root;
    }

    private int findRootIndexOfInOrder(int[] inorder, int startIndexOfInOrder, int 
    endIndexOfInOrder,
                                       int rootValue) {

        for (int i = startIndexOfInOrder; i <= endIndexOfInOrder; i++) {

            if (inorder[i] == rootValue) {
                return i;
                }
            }

            return -1;
    }
}
```