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

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

For example, given

preorder = [3,9,20,15,7]
inorder = [9,3,15,20,7]
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[] preorder, int[] inorder) {

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

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

    private TreeNode buildTreeHelper(int[] preorder, int rootIndexOfPreOrder,
                                     int[] inorder, int startIndexOfInOrder, int
                                     endIndexOfInOrder) {

        if (startIndexOfInOrder > endIndexOfInOrder) {
            return null;
        }

        TreeNode root = new TreeNode(preorder[rootIndexOfPreOrder]);

        int rootIndexOfInOrder = findRootIndexOfInOrder(preorder[rootIndexOfPreOrder],
                inorder, startIndexOfInOrder, endIndexOfInOrder);

        root.left = buildTreeHelper(preorder, rootIndexOfPreOrder + 1,
                inorder, startIndexOfInOrder, rootIndexOfInOrder - 1);

        root.right = buildTreeHelper(preorder, rootIndexOfInOrder -
        startIndexOfInOrder + rootIndexOfPreOrder + 1,
                inorder, rootIndexOfInOrder + 1, endIndexOfInOrder);

        return root;
    }

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

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

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

```
71          }
72
73          return -1;
74      }
75  }
```