

```

1  /* Given inorder and postorder traversal of a tree, construct the binary tree.
2
3  Note:
4  You may assume that duplicates do not exist in the tree.
5
6  For example, given
7
8  inorder = [9,3,15,20,7]
9  postorder = [9,15,7,20,3]
10 Return the following binary tree:
11
12     3
13    /\
14   9 20
15  /\  /\
16 15 7 */
17
18 class TreeNode {
19     int val;
20     TreeNode left;
21     TreeNode right;
22
23     TreeNode(int x) {
24         val = x;
25     }
26 }
27
28
29 class Solution {
30     public TreeNode buildTree(int[] inorder, int[] postorder) {
31
32         if (inorder == null || postorder == null
33             || inorder.length == 0 || postorder.length == 0) {
34             return null;
35         }
36
37         return buildTreeHelper(inorder, 0, inorder.length - 1, postorder,
38                                 postorder.length - 1);
39
40     private TreeNode buildTreeHelper(int[] inorder, int startIndexOfInOrder, int
41                                     endIndexOfInOrder,
42                                     int[] postorder, int rootIndexOfPostOrder) {
43
44         if (startIndexOfInOrder > endIndexOfInOrder) {
45             return null;
46         }
47
48         TreeNode root = new TreeNode(postorder[rootIndexOfPostOrder]);
49
50         int rootIndexOfInOrder = findRootIndexOfInOrder(inorder,
51                                                         startIndexOfInOrder, endIndexOfInOrder,
52                                                         postorder[rootIndexOfPostOrder]);
53
54         root.left = buildTreeHelper(inorder, startIndexOfInOrder, rootIndexOfInOrder
55                                     - 1,
56                                     postorder, rootIndexOfPostOrder - 1 - (endIndexOfInOrder -
57                                     rootIndexOfInOrder));
58         root.right = buildTreeHelper(inorder, rootIndexOfInOrder + 1,
59                                     endIndexOfInOrder,
60                                     postorder, rootIndexOfPostOrder - 1);
61
62         return root;
63     }
64
65     private int findRootIndexOfInOrder(int[] inorder, int startIndexOfInOrder, int
66                                     endIndexOfInOrder,
67                                     int rootValue) {
68
69         for (int i = startIndexOfInOrder; i <= endIndexOfInOrder; i++) {
70
71             if (inorder[i] == rootValue) {
72                 return i;
73             }
74         }
75     }
76 }

```

```
67  .....}
68  .....}
69
70  .....return -1;
71  .....}
72  }
```