

```

1  /* Given preorder and inorder 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  preorder = [3,9,20,15,7]
9  inorder = [9,3,15,20,7]
10 Return the following binary tree:
11
12     3
13    /\
14   9 20
15  /\  /\
16 15 7 */
17
18
19 class TreeNode {
20     int val;
21     TreeNode left;
22     TreeNode right;
23
24     TreeNode(int x) {
25         val = x;
26     }
27 }
28
29
30 class Solution {
31     public TreeNode buildTree(int[] preorder, int[] inorder) {
32
33         if (preorder == null || inorder == null
34             || preorder.length == 0 || inorder.length == 0) {
35             return null;
36         }
37
38         return buildTreeHelper(preorder, 0,
39             inorder, 0, inorder.length - 1);
40     }
41
42     private TreeNode buildTreeHelper(int[] preorder, int rootIndexOfPreOrder,
43                                     int[] inorder, int startIndexOfInOrder, int
44                                         endIndexOfInOrder) {
45
46         if (startIndexOfInOrder > endIndexOfInOrder) {
47             return null;
48         }
49
50         TreeNode root = new TreeNode(preorder[rootIndexOfPreOrder]);
51
52         int rootIndexOfInOrder = findRootIndexOfInOrder(preorder[rootIndexOfPreOrder],
53             inorder, startIndexOfInOrder, endIndexOfInOrder);
54
55         root.left = buildTreeHelper(preorder, rootIndexOfPreOrder + 1,
56             inorder, startIndexOfInOrder, rootIndexOfInOrder - 1);
57
58         root.right = buildTreeHelper(preorder, rootIndexOfInOrder +
59             startIndexOfInOrder + rootIndexOfPreOrder + 1,
60             inorder, rootIndexOfInOrder + 1, endIndexOfInOrder);
61
62         return root;
63     }
64
65     private int findRootIndexOfInOrder(int rootValue,
66                                     int[] inorder, int startIndexOfInOrder, int
67                                         endIndexOfInOrder) {
68
69         for (int i = startIndexOfInOrder; i <= endIndexOfInOrder; i++) {
70
71             if (inorder[i] == rootValue) {
72                 return i;
73             }
74         }
75     }
76 }

```

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