```java
/*

    Time complexity: O(N^2)
    Space complexity: O(H)

    where N is the number of nodes in the input tree
    and H is the height of the input tree

 */

public class Solution {

        private static BinaryTreeNode<Integer> buildTreeHelper(int[] postOrder, int postStart, int
postEnd, int[] inOrder, int inStart, int inEnd) {
            if (postStart > postEnd || inStart > inEnd) {
                return null;
            }

            int rootVal = postOrder[postEnd];
            BinaryTreeNode<Integer> root = new BinaryTreeNode<>(rootVal);

            // Find parent element index from inOrder array
            int k = 0;
            for (int i = inStart; i <= inEnd; i++) {
                if (rootVal == inOrder[i]) {
                    k = i;
                    break;
                }
            }

            root.left = buildTreeHelper(postOrder, postStart, postStart + k - inStart - 1, inOrder,
inStart, k - 1);
            root.right = buildTreeHelper(postOrder, postStart + k - inStart, postEnd - 1, inOrder, k +
1, inEnd);

            return root;
        }

        public static BinaryTreeNode<Integer> buildTree(int[] postOrder, int[] inOrder) {
            int n = postOrder.length;

            int postStart = 0;
            int postEnd = n - 1;
            int inStart = 0;
            int inEnd = n - 1;

            return buildTreeHelper(postOrder, postStart, postEnd, inOrder, inStart, inEnd);
        }

}
```