Problem 106: Construct Binary Tree from Inorder and Postorder Traversal

Problem Information

Difficulty: Medium
Acceptance Rate: 67.41%
Paid Only: No
Tags: Array, Hash Table, Divide and Conquer, Tree, Binary Tree

Problem Description

Given two integer arrays `inorder` and `postorder` where `inorder` is the inorder traversal of a binary tree and `postorder` is the postorder traversal of the same tree, construct and return _the binary tree_.

Example 1:

![](https://assets.leetcode.com/uploads/2021/02/19/tree.jpg)

Input: inorder = [9,3,15,20,7], postorder = [9,15,7,20,3] **Output:** [3,9,20,null,null,15,7]

Example 2:

Input: inorder = [-1], postorder = [-1] **Output:** [-1]

Constraints:

* `1 <= inorder.length <= 3000` * `postorder.length == inorder.length` * `-3000 <= inorder[i], postorder[i] <= 3000` * `inorder` and `postorder` consist of **unique** values. * Each value of `postorder` also appears in `inorder`. * `inorder` is **guaranteed** to be the inorder traversal of the tree. * `postorder` is **guaranteed** to be the postorder traversal of the tree.

Code Snippets

C++:

```
/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode() : val(0), left(nullptr), right(nullptr) {}
* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
* TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),
right(right) {}
* };
*/
class Solution {
public:
TreeNode* buildTree(vector<int>& inorder, vector<int>& postorder) {

}
};
```

Java:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode() {}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class Solution {
public TreeNode buildTree(int[] inorder, int[] postorder) {

}
}
```

Python3:

```
# Definition for a binary tree node.
# class TreeNode:
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution:
def buildTree(self, inorder: List[int], postorder: List[int]) ->
Optional[TreeNode]:
```