

# Problem 105: Construct Binary Tree from Preorder and Inorder Traversal

## Problem Information

Difficulty: **Medium**

Acceptance Rate: 67.86%

Paid Only: No

Tags: Array, Hash Table, Divide and Conquer, Tree, Binary Tree

## Problem Description

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

**Example 1:**



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

**Example 2:**

**Input:** `preorder = [-1]`, `inorder = [-1]` **Output:** `[-1]`

**Constraints:**

\* `1 <= preorder.length <= 3000` \* `inorder.length == preorder.length` \* `-3000 <= preorder[i], inorder[i] <= 3000` \* `preorder` and `inorder` consist of **unique** values. \* Each value of `inorder` also appears in `preorder`. \* `preorder` is **guaranteed** to be the preorder traversal of the tree. \* `inorder` is **guaranteed** to be the inorder 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>& preorder, vector<int>& inorder) {

    }
};
```

## 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[] preorder, int[] inorder) {

    }
}
```

### 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, preorder: List[int], inorder: List[int]) ->
Optional[TreeNode]:
```