Construct Binary Tree from Preorder and Inorder Traversal

Problem Statement

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 leq text{preorder.length} leq 3000)
- (text{inorder.length} == text{preorder.length})
- (-3000 leq text{preorder[i]}, text{inorder[i]} leq 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.

Solution

To solve this problem, we need to reconstruct the binary tree using the given preorder and inorder traversal arrays. The approach involves the following steps:

1. **Identify the Root Node:**
   - The preorder array's first element is always the tree's root.

2. **Find the Root in the Inorder Array:**
   - Locate the root node in the inorder array. Elements to the left of the root node in the inorder array belong to the left subtree, and elements to the right belong to the right subtree.

3. **Recursively Construct Subtrees:**
   - Using the identified left and right subtrees from the inorder array, recursively construct the left and right subtrees.

4. **Repeat Until Tree is Constructed:**
   - Continue this process recursively for each subtree until the entire tree is constructed.

Explanation of the Code

1. Class Definitions:

- TreeNode is defined as the basic structure of a node in a binary tree.

2. Base Case:

- If either the preorder or inorder list is empty, return None.

3. Root Node Initialization:

- The first element of the preorder list is assigned as the root node.

4. Finding the Root in Inorder List:

- The index of the root node in the inorder list is found. This index helps in splitting the inorder list into left and right subtrees.

5. Recursive Calls:

- The left subtree is built using the elements before the root in inorder list and the corresponding elements in preorder list.
- The right subtree is built using the elements after the root in inorder list and the corresponding elements in preorder list.

6. Return Constructed Tree:

- The constructed root node, with its left and right subtrees, is returned.

This solution effectively reconstructs the binary tree by leveraging the properties of preorder and inorder traversals.