LeetCode 105

https://leetcode.com/problems/construct-binary-tree-from-preorder-and-inorder-traversal/description/

Yifeng Zeng

Description

105. Construct Binary Tree from Preorder and Inorder Traversal

Given preorder and inorder traversal of a tree, construct the binary tree.

Note:
You may assume that duplicates do not exist in the tree.

For example, given

preorder = [3,9,20,15,7]

inorder = [9,3,15,20,7]

Return the following binary tree:

```
    3
   / \
  9  20
    /  \
   15   7
```

Idea Report

We can come up an example so it's easier to explain, I will just use the above example.
For the preorder array, the first number [3] is always the root. Then we can find this root number

[3] in the inorder array. The left part of [3] is [9], this is the whole left children of root [3], the right part of [3] in inorder array is [15,20,7], these are the whole right children of root [3]. By counting how many left childre, and right children of root [3], we would know where those children are in the preorder array. Which the index is 1 [9], and 3,4,5 [20,15,7] in the preorder array. Because the first index of the subarray is the child root, we can know index 1[nodex 9] is our left child, index 3[node 20] is our right child. By passing the start and end index of the inorder array, and the child root index in the preorder array, we can recursivly build up this tree.

Code

```
class Solution {
    public TreeNode buildTree(int[] preorder, int[] inorder) {
        return helper(preorder, inorder, 0, 0, inorder.length - 1);
    }

    private TreeNode helper(int[] preorder, int[] inorder,
                            int preIndex, int inStart, int inEnd) {
        if (inStart > inEnd) {
            return null;
        }
        TreeNode root = new TreeNode(preorder[preIndex]);
        int index = inStart;
        while (index <= inEnd) {
            if (inorder[index] == preorder[preIndex]) {
                break;
            }
            index++;
        }

        root.left = helper(preorder, inorder, preIndex + 1, inStart, index - 1);
        root.right = helper(preorder, inorder, preIndex
                            + (index - inStart) + 1, index + 1, inEnd);
        return root;
    }
}
```

Summary

- Divide and conquer, build up the root at the current level, and recursively get the left child and right child.
- Sometimes using a helper() function with the start and end index may do the trick.
- Follow up is the same idea just a different order (from nums.length - 1 to 0).

Follow up

LeetCode 106. Construct Binary Tree from Inorder and Postorder Traversal

Basically the same as LC 105, because postorder reversed is the root-right-left preorder, so just start from the end of the int[] postorder array, and postIndex - 1 is the right child index, and calculate the left child index in the postorder array.

```
class Solution {
    public TreeNode buildTree(int[] inorder, int[] postorder) {
        return helper(inorder, postorder, postorder.length - 1,
                      0, inorder.length - 1);
    }

    private TreeNode helper(int[] inorder, int[] postorder,
                            int postIndex, int inStart, int inEnd) {
        if (inStart > inEnd || postIndex < 0) {
            return null;
        }

        TreeNode root = new TreeNode(postorder[postIndex]);
        int index = inStart;
        while (index <= inEnd) {
            if (inorder[index] == postorder[postIndex]) {
                break;
            }
            index++;
        }
        root.right = helper(inorder, postorder, postIndex - 1,
                            index + 1, inEnd);
        root.left = helper(inorder, postorder, postIndex - (inEnd - index) - 1,
                           inStart, index - 1);
        return root;
    }
}
```