105. 从前序与中序遍历序列构造二叉树

难度 中等 619

根据一棵树的前序遍历与中序遍历构造二叉树。

注意:
你可以假设树中没有重复的元素。

例如，给出

```
前序遍历 preorder = [3,9,20,15,7]
中序遍历 inorder = [9,3,15,20,7]
```

返回如下的二叉树：

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

解题思路：

前序遍历 preorder = [3,9,20,15,7]
中序遍历 inorder = [9,3,15,20,7]

0	1	2	3	4
3（root）	9	20	15	7
9	3（root）	15	20	7

左子树： 9（确定位置递归结束）
右子树：20，15，7 继续递归

0	1	2
20（root）	15	7
15	20（root）	7

左子树：15（确定位置递归结束）
右子树：7（确定位置递归结束）

树的结构图：

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

可以看出，这是一个递归的过程，根据前序遍历的第一个数是根，从中序遍历中找到对应的索引，索引前面的数是左子树序列，索引后后面在从前序遍历序列中找到左子树的序列，和右子树的序列
这样可以得到，左子树的前序遍历和中序遍历的序列，右子树的前序遍历和中序遍历的序列
左右子树，就相当于一个新树，在树里再去根据前序遍历序列第一个数是根，根据根从中序遍历序列中找到对应索引，索引前是左子树序列

根据索引，可以知道，中序遍历中，左子树的中序遍历序列是:inorder[0,mix_index]., 右子树的中序遍历序列是:inorder[mix_index+1:].
那么对应的前序遍历的序列是：左子树：preorder[1,mix_index+1]（0-->root），右子树：preorder[mix_index+1]

```python
def buildTree(self, preorder, inorder):
        """
        :type preorder: List[int]
        :type inorder: List[int]
        :rtype: TreeNode
        """
        if not (preorder and inorder):
            return None
```

```
9
10          root = TreeNode(preorder[0])
11          mid_index = inorder.index(preorder[0])
12
13          root.left = self.buildTree(preorder[1:mid_index+1], inorder[:mid_index])
14          root.right = self.buildTree(preorder[mid_index+1:], inorder[mid_index+1:])
15          return root
```

```
def buildTree(self, preorder, inorder):
        """
        :type preorder: List[int]
        :type inorder: List[int]
        :rtype: TreeNode
        """
        def build(stop):
    # 变量 pre 保存当前要构造的树的 root
    # 变量 in 保存 inorder 数组中可以成为 root 的数字们的开头那个
     # 对于当前要构造的树，有一个停止点 stop ，inorder 数组中第 in 项到第 stop 项是要构造的树的节点值们
     # 每次递归调用，都会确定出一个停止点，它告诉了子调用在哪里停止，把自己的根节点值作为左子树调用的停止点，
     # 自己的（父调用给下来的）停止点作为右子树的停止点
            if inorder and inorder[-1] != stop:
                root = TreeNode(preorder.pop())
                root.left = build(root.val)
                inorder.pop()
                root.right = build(stop)
                return root
        preorder.reverse()
        inorder.reverse()
        return build(None)

```

我没看懂这个答案～～～