0105. 从前序与中序遍历序列构造二叉树

ITCharge 大约 2 分钟

- 标签：树、数组、哈希表、分治、二叉树
- 难度：中等

题目链接

- 0105. 从前序与中序遍历序列构造二叉树 - 力扣

题目大意

描述：给定一棵二叉树的前序遍历结果 `preorder` 和中序遍历结果 `inorder` 。

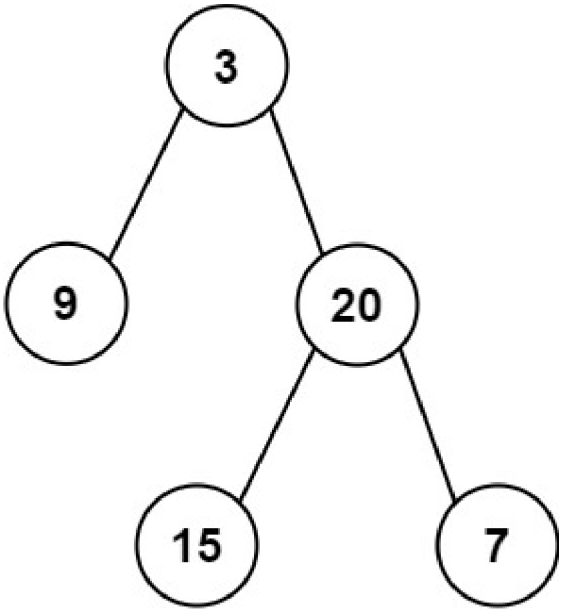
要求：构造出该二叉树并返回其根节点。

说明：

- $1 \leq preorder.length \leq 3000$。
- $inorder.length == preorder.length$。
- $-3000 \leq preorder[i], inorder[i] \leq 3000$。
- `preorder` 和 `inorder` 均无重复元素。
- `inorder` 均出现在 `preorder` 。
- `preorder` 保证为二叉树的前序遍历序列。
- `inorder` 保证为二叉树的中序遍历序列。

示例：

- 示例 1：



img

```py
输入: preorder = [3,9,20,15,7], inorder = [9,3,15,20,7]
输出: [3,9,20,null,null,15,7]
```

- 示例 2:

```py
输入: preorder = [-1], inorder =     |
输出: [-1]
```

解题思路

思路 1：递归遍历

前序遍历的顺序是：根 -> 左 -> 右。中序遍历的顺序是：左 -> 根 -> 右。根据前序遍历的顺序，可以找到根节点位置。然后在中序遍历的结果中可以找到对应的根节点位置，就可以从根节点位置将二叉树分割成左子树、右子树。同时能得到左右子树的节点个数。此时构建当前节点，并递归建立左右子树，在左右子树对应位置继续递归遍历进行上述步骤，直到节点为空，具体操作步骤如下：

1. 从前序遍历顺序中当前根节点的位置在 `postorder[0]` 。
2. 通过在中序遍历中查找上一步根节点对应的位置 `inorder[k]` ，从而将二叉树的左右子树分隔开，并得到左右子树节点的个数。
3. 从上一步得到的左右子树个数将前序遍历结果中的左右子树分开。

4. 构建当前节点，并递归建立左右子树，在左右子树对应位置继续递归遍历并执行上述三步，直到节点为空。

思路 1：代码

```py
class Solution:
    def buildTree(self, preorder: List[int], inorder: List[int]) -> TreeNode:
        def createTree(preorder, inorder, n):
            if n == 0:
                return None
            k = 0
            while preorder[0] != inorder[k]:
                k += 1
            node = TreeNode(inorder[k])
            node.left = createTree(preorder[1: k+1], inorder[0: k], k)
            node.right = createTree(preorder[k+1:], inorder[k+1:], n-k-1)
            return node
        return createTree(preorder, inorder, len(inorder))
```

思路 1：复杂度分析

- **时间复杂度**：$O(n)$，其中 n 是二叉树的节点数目。
- **空间复杂度**：$O(n)$。递归函数需要用到栈空间，栈空间取决于递归深度，最坏情况下递归深度为 n，所以空间复杂度为 $O(n)$。