0129. 求根节点到叶节点数字之和

ITCharge 大约 2 分钟

- 标签：树、深度优先搜索、二叉树
- 难度：中等

题目链接

- 0129. 求根节点到叶节点数字之和 - 力扣

题目大意

描述：给定一个二叉树的根节点 `root` ，树中每个节点都存放有一个 `0` 到 `9` 之间的数字。每条从根节点到叶节点的路径都代表一个数字。例如，从根节点到叶节点的路径是 `1 -> 2 -> 3` ，表示数字 `123` 。

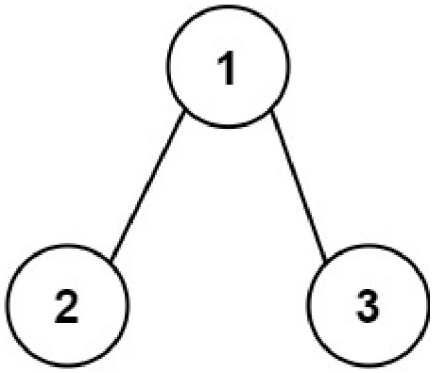
要求：计算从根节点到叶节点生成的所有数字的和。

说明：

- **叶节点**：指没有子节点的节点。
- 树中节点的数目在范围 $[1, 1000]$ 内。
- $0 \leq Node.val \leq 9$。
- 树的深度不超过 10。

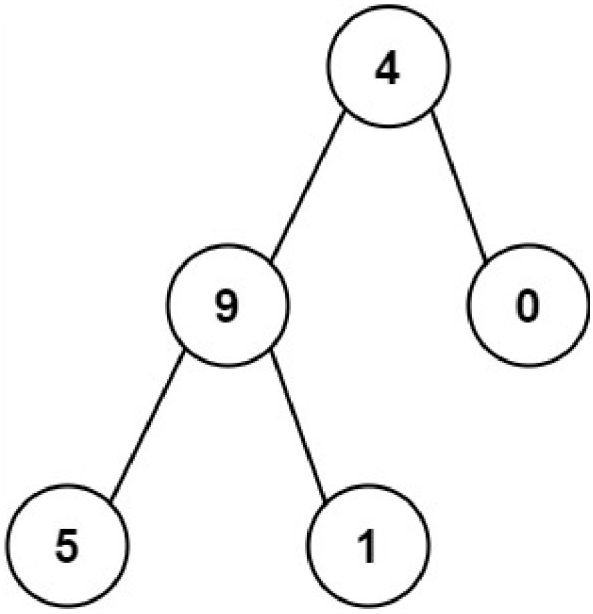
示例：

- 示例 1：



```py
输入：root = [1,2,3]
输出：25
解释：
从根到叶子节点路径 1->2 代表数字 12
从根到叶子节点路径 1->3 代表数字 13
因此，数字总和 = 12 + 13 = 25
```

- 示例 2：




```py
输入：root = [4,9,0,5,1]
输出：1026
解释：
从根到叶子节点路径 4->9->5 代表数字 495
从根到叶子节点路径 4->9->1 代表数字 491
从根到叶子节点路径 4->0 代表数字 40
因此，数字总和 = 495 + 491 + 40 = 1026
```

解题思路

思路 1：深度优先搜索

1. 记录下路径上所有节点构成的数字，使用变量 `pre_total` 保存下当前路径上构成的数字。
2. 如果遇到叶节点，则直接返回当前数字。

3. 如果没有遇到叶节点，则递归遍历左右子树，并累加对应结果。

思路 1：代码

```py
class Solution:
    def dfs(self, root, pre_total):
        if not root:
            return 0
        total = pre_total * 10 + root.val
        if not root.left and not root.right:
            return total
        return self.dfs(root.left, total) + self.dfs(root.right, total)

    def sumNumbers(self, root: Optional[TreeNode]) -> int:
        return self.dfs(root, 0)
```

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 的栈空间，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 $O(n)$。