0113. 路径总和 II

ITCharge 大约 2 分钟

- 标签：树、深度优先搜索、回溯、二叉树
- 难度：中等

题目链接

- 0113. 路径总和 II - 力扣

题目大意

描述：给定一棵二叉树的根节点 `root` 和一个整数目标 `targetSum` 。

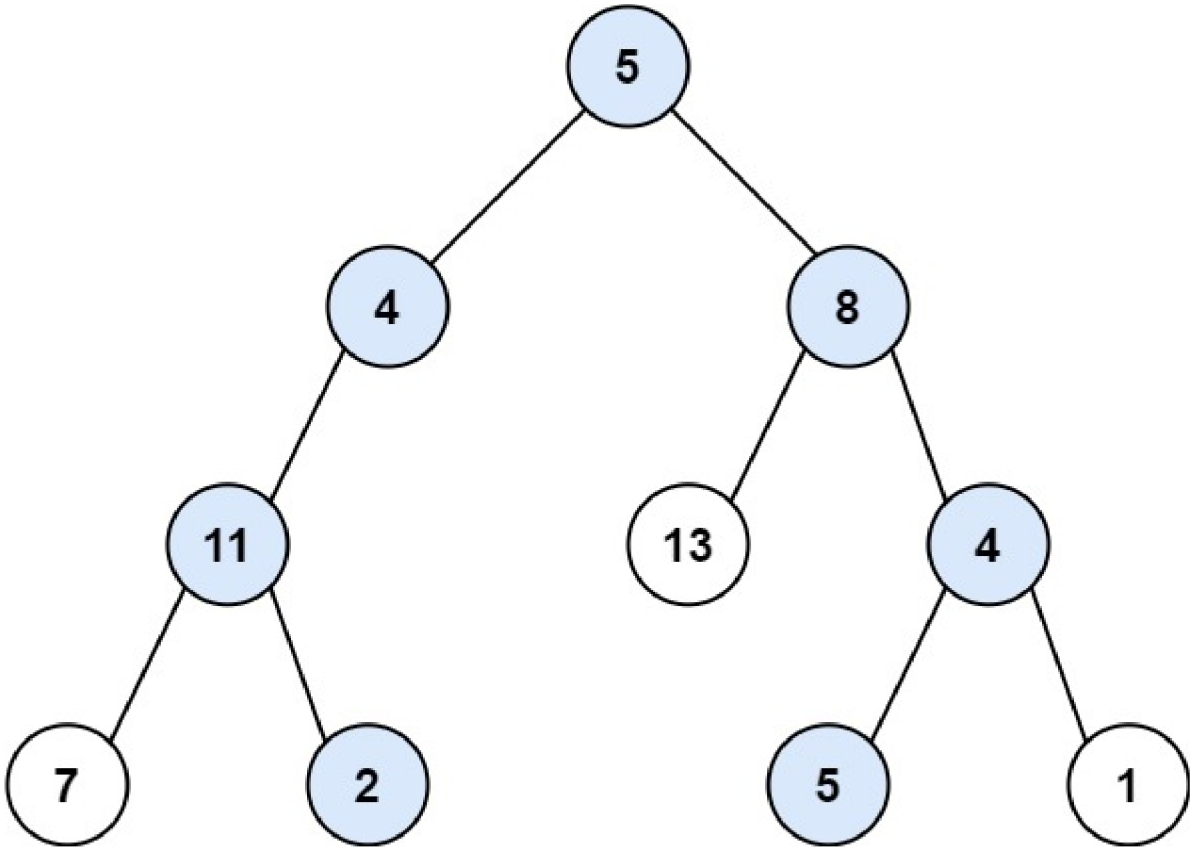
要求：找出「所有从根节点到叶子节点路径总和」等于给定目标和 `targetSum` 的路径。

说明：

- **叶子节点**：指没有子节点的节点。
- 树中节点总数在范围 $[0, 5000]$ 内。
- $-1000 \leq Node.val \leq 1000$。
- $-1000 \leq targetSum \leq 1000$。

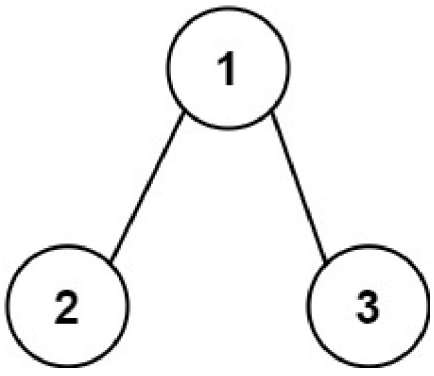
示例：

- 示例 1：




```py
输入：root = [5,4,8,11,null,13,4,7,2,null,null,5,1], targetSum = 22
输出：[[5,4,11,2],[5,8,4,5]]
```

- 示例 2：




```py
输入：root = [1,2,3], targetSum = 5
输出：[]
```

解题思路

思路 1：回溯

在回溯的同时，记录下当前路径。同时维护 `targetSum`，每遍历到一个节点，就减去该节点值。如果遇到叶子节点，并且 `targetSum == 0` 时，将当前路径加入答案数组中。然后递归遍历左右子树，并回退当前节点，继续遍历。

具体步骤如下：

1. 使用列表 `res` 存储所有路径，使用列表 `path` 存储当前路径。
2. 如果根节点为空，则直接返回。
3. 将当前节点值添加到当前路径 `path` 中。
4. `targetSum` 减去当前节点值。
5. 如果遇到叶子节点，并且 `targetSum == 0` 时，将当前路径加入答案数组中。
6. 递归遍历左子树。
7. 递归遍历右子树。
8. 回退当前节点，继续递归遍历。

思路 1：代码

py

```
class Solution:
    def pathSum(self, root: TreeNode, targetSum: int) -> List[List[int]]:
        res = []
        path = []

        def dfs(root: TreeNode, targetSum: int):
            if not root:
                return
            path.append(root.val)
            targetSum -= root.val
            if not root.left and not root.right and targetSum == 0:
                res.append(path[:])
            dfs(root.left, targetSum)
```

```
            dfs(root.right, targetSum)
            path.pop()

        dfs(root, targetSum)
        return res
```

思路 1：复杂度分析

- **时间复杂度**：$O(n^2)$，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 $O(n)$。