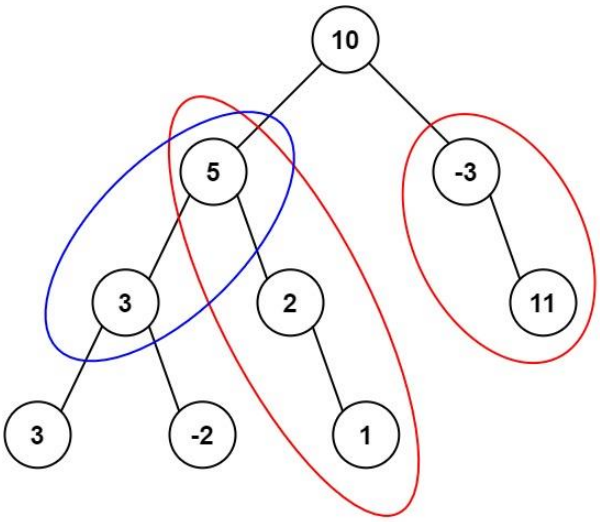
437. Path Sum III

Given the root of a binary tree and an integer targetSum, return *the number of paths where the sum of the values along the path equals* targetSum.

The path does not need to start or end at the root or a leaf, but it must go downwards (i.e., traveling only from parent nodes to child nodes).

Example 1:



Input: root = [10,5,-3,3,2,null,11,3,-2,null,1], targetSum = 8

Output: 3

Explanation: The paths that sum to 8 are shown.

Example 2:

Input: root = [5,4,8,11,null,13,4,7,2,null,null,5,1], targetSum = 22

Output: 3

Constraints:

- The number of nodes in the tree is in the range [0, 1000].
- -10^9 <= Node.val <= 10^9
- -1000 <= targetSum <= 1000

```
# Definition for a binary tree node.
# class TreeNode(object):
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution(object):
    cnt = 0
    def pathSum(self, root, targetSum):
        """
        :type root: TreeNode
        :type targetSum: int
        :rtype: int
        """
        self.cnt = 0

        def dfs(node, currentSum):
            if not node:
                return

            currentSum += node.val
            if currentSum == targetSum:
                self.cnt += 1

            dfs(node.left, currentSum)
            dfs(node.right, currentSum)

        def findPaths(node):
            if not node:
                return

            dfs(node, 0)

            findPaths(node.left)
            findPaths(node.right)

        findPaths(root)
        return self.cnt
```