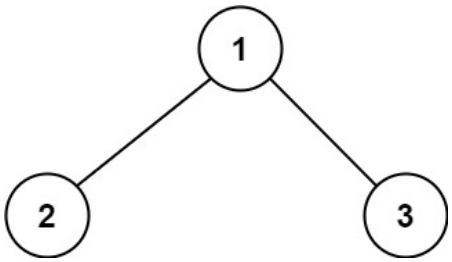
124. Binary Tree Maximum Path Sum

A **path** in a binary tree is a sequence of nodes where each pair of adjacent nodes in the sequence has an edge connecting them. A node can only appear in the sequence **at most once**. Note that the path does not need to pass through the root.

The **path sum** of a path is the sum of the node's values in the path.

Given the root of a binary tree, return *the maximum **path sum** of any **non-empty** path*.

Example 1:

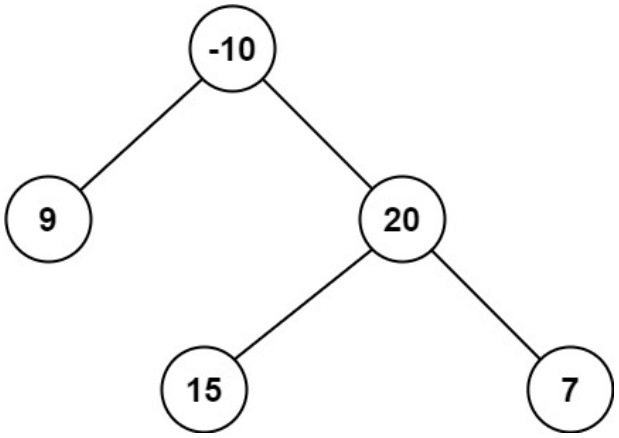


Input: root = [1,2,3]

Output: 6

Explanation: The optimal path is 2 -> 1 -> 3 with a path sum of 2 + 1 + 3 = 6.

Example 2:



Input: root = [-10,9,20,null,null,15,7]

Output: 42

Explanation: The optimal path is 15 -> 20 -> 7 with a path sum of 15 + 20 + 7 = 42.

Constraints:

- The number of nodes in the tree is in the range [1, 3 * 10^4].
- -1000 <= Node.val <= 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):
    def maxPathSum(self, root):
        """
        :type root: TreeNode
        :rtype: int
        """
        if not root:
            return 0

        ans = [None]

        def check(node):

            if not node:
                return 0

            l = max(0,check(node.left))
            r = max(0, check(node.right))

            ans[0] = max(ans[0], l + node.val + r)
            return node.val + max(l,r)

        check(root)
        return ans[0]
```