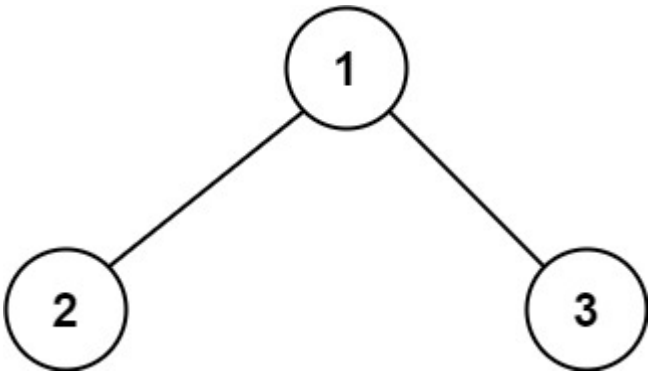
124. Binary Tree Maximum Path Sum

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.

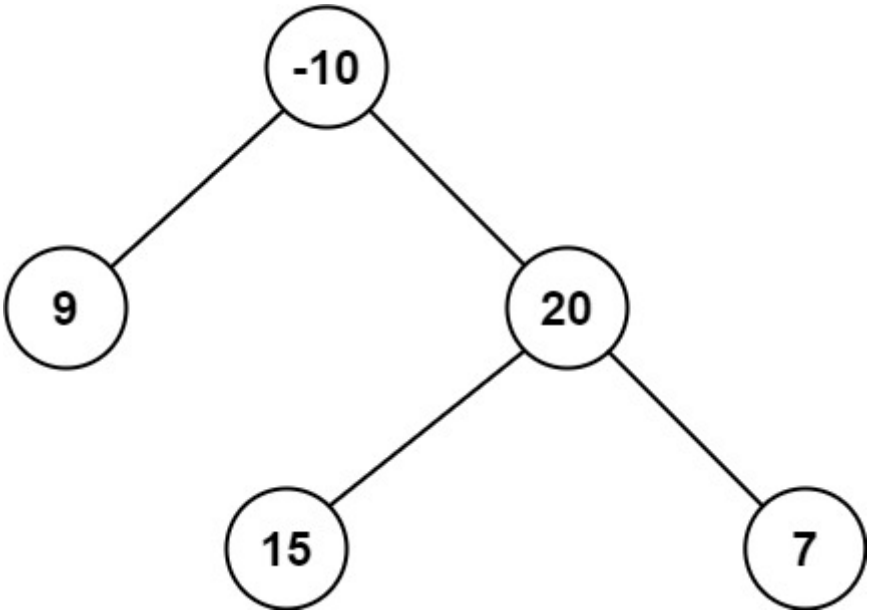
Given the `root` of a binary tree, return *the maximum **path sum** of any **non-empty** path*.

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<sup>4</sup>]`.
- `-1000 <= Node.val <= 1000`

```
import sys

class Solution:
    def maxPathSum(self, root: Optional[TreeNode]) -> int:
        if root.left is root.right:
            return root.val
        self.res = [-1001]
        self.maxPathSumHelper(root)
        return self.res[0]


    def maxPathSumHelper(self,root):
        if root is None:
            return 0

        left = max(0,self.maxPathSumHelper(root.left))
        right = max(0,self.maxPathSumHelper(root.right))

        self.res[0] = max(self.res[0],left+right+root.val)
        return max(left,right)+root.val
```

```
#         self.right = right
import sys
class Solution:
    def maxPathSum(self, root: TreeNode) -> int:
        if root.left is None and root.right is None:
            return root.val
        res = [-sys.maxsize]
        self.maxPathSumHelper(root,res)
        return res[0]
    def maxPathSumHelper(self,root,res):
        if root is None:
            return 0
        lt = self.maxPathSumHelper(root.left,res)
```

```
        rt = self.maxPathSumHelper(root.right,res)
        temp = max(max(lt,rt)+root.val, root.val)
        ans = max(temp,root.val+lt+rt)
        res[0] = max(res[0],ans)
        return temp
```