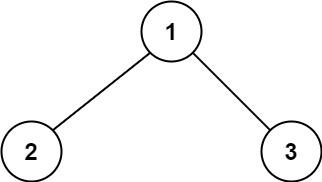
A **path** in a binary tree is a sequence of nodes where each pair of adjacent nodes in the sequence has an edge connecting them. A node can only appear in the sequence **at most once**. Note that the path does not need to pass through the root.

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:**

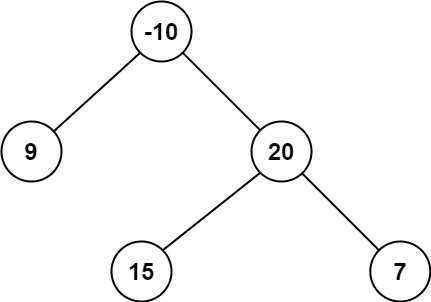


**Input:** root = [1,2,3]

**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.

**Solution:**

/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode() {}

\* TreeNode(int val) { this.val = val; }

\* TreeNode(int val, TreeNode left, TreeNode right) {

\* this.val = val;

\* this.left = left;

\* this.right = right;

\* }

\* }

\*/

class Solution {

public int maxPathSum(TreeNode root) {

int[] maxValue = new int[1];

maxValue[0] = Integer.MIN\_VALUE;

maxPathDown(root, maxValue);

return maxValue[0];

}

private int maxPathDown(TreeNode root, int maxValue[]){

if(root == null)

return 0;

int left = Math.max(0,maxPathDown(root.left, maxValue));

int right = Math.max(0, maxPathDown(root.right, maxValue));

maxValue[0] = Math.max(maxValue[0], left+right+root.val);

return root.val+Math.max(left, right);

}

}