PROBLEM

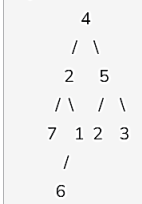
Sum of nodes on the longest path from root to leaf node

Medium Accuracy: 52.39% Submissions: 83K+ Points: 4

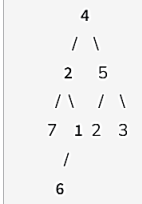
Given a binary tree having **n** nodes. Find the sum of all nodes on the longest path from root to any leaf node. If two or more paths compete for the longest path, then the path having maximum sum of nodes will be considered.

Example 1:

Input:



Output:
13
Explanation:



The highlighted nodes (4, 2, 1, 6) above are part of the longest root to leaf path having sum = (4 + 2 + 1 + 6) = 13

Example 2:

Input:

Output:
11
Explanation:
Use path 1->3->7, with sum 11.

Your Task:
You don't need to read input or print anything. Your task is to complete the function **sumOfLongRootToLeafPath()** which takes root node of the tree as input parameter and returns an integer denoting the sum of the longest root to leaf path of the tree.

Expected Time Complexity: O(n)
Expected Auxiliary Space: O(n)

Constraints:

$1 <= n <= 10^5$

$0 <= \text{data of each node} <= 10^4$

CODE

#User function Template for python3

'''

class Node:
def __init__(self,val):
self.data=val
self.left=None
self.right=None

'''

```
class Solution:
   def sumOfLongRootToLeafPath(self,root):
      maxPath=0
      maxSum=0
      def traverse(root,curLen,curSum):
         nonlocal maxPath,maxSum
         if not root:
            return
         curLen+=1
         curSum+=root.data
         if root.left==None and root.right==None:
            if maxPath<curLen or (curLen==maxPath and curSum>maxSum):
               maxSum=curSum
               maxPath=curLen
         traverse(root.left,curLen,curSum)
         traverse(root.right,curLen,curSum)
      traverse(root,0,0)
      return maxSum
```

EXPLANATION

def sumOfLongRootToLeafPath(self,root):

This line defines a method **sumOfLongRootToLeafPath** within the Solution class. It takes root as a parameter, which presumably represents the root of a tree.

```
maxPath=0
maxSum=0
```

These lines initialize two variables maxPath and maxSum to 0. These variables will be used to store the length and sum of the longest root-to-leaf path found during traversal.

```
def traverse(root,curLen,curSum):
```

This line defines an inner function named traverse. This function takes three parameters: root (current node being visited), curLen (current length of the path from root to the current node), and curSum (current sum along the path from root to the current node).

```
    nonlocal maxPath,maxSum
```

This line declares that maxPath and maxSum are non-local variables. This allows the inner function traverse to modify the values of maxPath and maxSum declared in the outer function.

```
    if not root:
        return
```

This line checks if the current node (root) is None. If it is, it means we've reached the end of a path, so the function returns.

```
    curLen+=1
    curSum+=root.data
```

These lines update curLen by adding 1 (since we're moving down the tree) and update curSum by adding the value of the current node's data.

```
    if root.left==None and root.right==None:
```

This line checks if the current node is a leaf node (i.e., it has no left or right child).

```
        if maxPath<curLen or (curLen==maxPath and curSum>maxSum):
            maxSum=curSum
            maxPath=curLen
```

This block of code updates maxPath and maxSum if the current path is longer than the previously recorded longest path (maxPath) or if the current path has the same length as the previously recorded longest path but a greater sum (curSum > maxSum).

traverse(root.left,curLen,curSum)
traverse(root.right,curLen,curSum)

These lines recursively call the traverse function on the left and right children of the current node, updating curLen and curSum accordingly.

traverse(root,0,0)

This line starts the traversal from the root of the tree, initializing curLen and curSum as 0.

return maxSum

Finally, the function returns the maximum sum found along the longest root-to-leaf path.