

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

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

Maximize your earnings for your published articles in Dev Scripiter 2024! Gain recognition & extra compensation while elevating your tech profile.



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:**

```

      4
     / \
    2   5
   / \ / \
  7  1 2  3
   /
  6

```

Output:

13

Explanation:

```

      4
     / \
    2   5
   / \ / \
  7  1 2  3
   /
  6

```

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:**

```

      1
     / \
    2   3
   / \ / \
  4  5 6  7

```

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:** $1 \leq n \leq 10^5$ $0 \leq \text{data of each node} \leq 10^4$



CODE

```

class Solution{
static int ans = Integer.MIN_VALUE,mxCount=0;
static void help(Node root, int sum, int count){
    if(root==null){
        if(count>mxCount){
            ans=sum;
            mxCount=count;
        }
        else if(count==mxCount){
            ans=Math.max(ans,sum);
        }
        return;
    }
    sum+=root.data;
    count++;
    help(root.left,sum,count);
    help(root.right,sum,count);

}
public int sumOfLongRootToLeafPath(Node root)
{
    //code here
    int sum=0,count=0;
    ans=Integer.MIN_VALUE;
    mxCount=0;
    help(root,sum,count);
    return ans;
}
}

```

OUTPUT**Compilation Completed**For Input:  

4 2 5 7 1 2 3 N N 6 N

Your Output:

13

Expected Output:

13

EXPLANATION

Static Variables:

- **ans:** Keeps track of the sum of the longest root-to-leaf path found so far.
- **mxCount:** Keeps track of the length (number of nodes) of the longest root-to-leaf path found so far.

Recursive Helper Method (help):

- This method traverses the binary tree recursively.
- At each step, it updates ans and mxCount based on the current path being explored.
- When it reaches the end of a path (a leaf node), it compares the length of this path with the longest path found so far (mxCount). If it's longer, it updates both ans and mxCount. If it's equally long, it updates ans if the sum of the current path is greater.

Main Method (sumOfLongRootToLeafPath):

- Initializes variables and resets ans and mxCount.
- Calls the recursive helper method (help) to compute the sum of the longest root-to-leaf path.
- Returns the computed sum.

In simple terms, the code explores every path from the root to a leaf in the binary tree. It keeps track of the longest path it finds and computes the sum of that path. Finally, it returns the sum of the longest path found.