LeetCode Explore Problems Mock New Contest Articles Interviewing? Discuss here! Discuss Store Premium

Description Solution Submissions Discuss (197)

C#

Quick Navigation View in Article

Intuition

We can think of any path (of nodes with the same values) as up to two arrows extending from it's root.

Specifically, the *root* of a path will be the unique node such that the parent of that node does not appear in the path, and an *arrow* will be a path where the root only has one child node in the path.

Then, for each node, we want to know what is the longest possible arrow extending left, and the longest possible arrow extending right? We can solve this using recursion.

Algorithm

Let `arrow_length(node)` be the length of the longest arrow that extends from the `node`. That will be `1 + arrow_length(node.left)` if `node.left` exists and has the same value as `node`. Similarly for the `node.right` case.

While we are computing arrow lengths, each candidate answer will be the sum of the arrows in both directions from that node. We record these candidate answers and return the best one.

```
/**
 * Definition
node.
 * public clas
 *     public
 *     public
 *     public
 *     public
val = x; }
 * }
 */
public class S
    int ans;
    public int
LongestUnivalu
root) {
        ans =
        GetArr
        returr
    }

        public
GetArrowLenght
        {
            if
return 0;

            ir
GetArrowLenght
            ir
GetArrowLenght

            ir
arrowRight = 0
```

Testcase Run Code Result

Finished Runtime:

Your input	[5,4,5,1,
Output	2
Expected	2

Console How

Problems Pick One Prev 687/1157 Next Run Code