549. Binary Tree Longest Consecutive Sequence II

Medium 843 64 Add to List Share

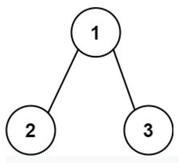
Given the `root` of a binary tree, return *the length of the longest consecutive path in the tree*.

A consecutive path is a path where the values of the consecutive nodes in the path differ by one. This path can be either increasing or decreasing.

- For example, `[1,2,3,4]` and `[4,3,2,1]` are both considered valid, but the path `[1,2,4,3]` is not valid.

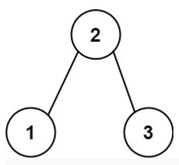
On the other hand, the path can be in the child-Parent-child order, where not necessarily be parent-child order.

Example 1:




```
Input: root = [1,2,3]
Output: 2
Explanation: The longest consecutive path is [1, 2] or [2, 1].
```

Example 2:




```
Input: root = [2,1,3]
Output: 3
Explanation: The longest consecutive path is [1, 2, 3] or [3, 2, 1].
```

Constraints:

- The number of nodes in the tree is in the range `[1, 3 * 10^4]`.
- `-3 * 10^4 <= Node.val <= 3 * 10^4`

Accepted 39,023 | Submissions 80,790

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Google 2

Related Topics

Tree Depth-First Search Binary Tree

Similar Questions

Binary Tree Longest Consecutive Sequence Medium

```java
public class Solution {
    int maxval = 0;

    public int longestConsecutive(TreeNode root) {
        longestPath(root);
        return maxval;
    }

    public int[] longestPath(TreeNode root) {
        if (root == null) {
            return new int[] {0,0};
        }

        int inr = 1, dcr = 1;
        if (root.left != null) {
            int[] left = longestPath(root.left);
            if (root.val == root.left.val + 1) {
                dcr = left[1] + 1;
            } else if (root.val == root.left.val - 1) {
                inr = left[0] + 1;
            }
        }

        if (root.right != null) {
            int[] right = longestPath(root.right);
            if (root.val == root.right.val + 1) {
                dcr = Math.max(dcr, right[1] + 1);
            } else if (root.val == root.right.val - 1) {
                inr = Math.max(inr, right[0] + 1);
            }
        }

        maxval = Math.max(maxval, dcr + inr - 1);
        return new int[] {inr, dcr};
    }
}
```