687 Longest Univalue Path

2018年5月2日 19:43

Given a binary tree, find the length of the longest path where each node in the path has the same value. This path may or may not pass through the root.

Note: The length of path between two nodes is represented by the number of edges between them.

Example 1:

Input:

```
      5
     / \
    4   5
   / \   \
  1   1   5
```

Output:

2

Example 2:

Input:

```
      1
     / \
    4   5
   / \   \
  4   4   5
```

Output:

2

Note: The given binary tree has not more than 10000 nodes. The height of the tree is not more than 1000.

来自 <https://leetcode.com/problems/longest-univalue-path/description/>

给定一个二叉树，找到最长的路径，这个路径中的每个节点具有相同值。 这条路径可以经过也可以不经过根节点。

注意：两个节点之间的路径长度由它们之间的边数表示。

注意: 给定的二叉树不超过10000个结点。 树的高度不超过1000。

Solution for Python3:

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None

class Solution:
    def longestUnivaluePath(self, root):
```

```
        """
        :type root: TreeNode
        :rtype: int
        """
        self.ans = 0

        def child_length(root):
            if not root:
                return 0
            left_len = child_length(root.left)
            right_len = child_length(root.right)
            left_edge = right_edge = 0
            if root.left and root.left.val == root.val:
                left_edge = left_len + 1
            if root.right and root.right.val == root.val:
                right_edge = right_len + 1
            self.ans = max(self.ans, left_edge + right_edge)
            return max(left_edge, right_edge)
        child_length(root)
        return self.ans
```

Solution for C++:

```
class Solution {
    int ans;
public:
    int longestUnivaluePath(TreeNode* root) {
        ans = 0;
        child_length(root);
        return ans;
    }

    int child_length(TreeNode* root) {
        if (root == NULL)
            return 0;
        int left = child_length(root->left);
        int right = child_length(root->right);
        int L_edge = 0, R_edge = 0;
        if (root->left && root->left->val == root->val)
            L_edge += left + 1;
        if (root->right && root->right->val == root->val)
            R_edge += right + 1;
        ans = max(ans, L_edge + R_edge);
        return max(L_edge, R_edge);
```

```
22        }
23    };
```