543 Diameter of Binary Tree

2018年4月14日 16:03

Given a binary tree, you need to compute the length of the diameter of the tree. The diameter of a binary tree is the length of the **longest** path between any two nodes in a tree. This path may or may not pass through the root.

Example:

Given a binary tree

```
      1
     / \
    2   3
   / \
  4   5
```

Return **3**, which is the length of the path [4,2,1,3] or [5,2,1,3].

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

来自 <https://leetcode.com/problems/diameter-of-binary-tree/description/>

给定一棵二叉树，你需要计算它的直径长度。一棵二叉树的直径长度是任意两个结点路径长度中的最大值。这条路径可能穿过根结点。

示例：

给定二叉树

```
      1
     / \
    2   3
   / \
  4   5
```

返回 **3**, 它的长度是路径 [4,2,1,3] 或者 [5,2,1,3]。

注意：两结点之间的路径长度是以它们之间边的数目表示。

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 Solution1:
    def diameterOfBinaryTree(self, root):
        """
        :type root: TreeNode
        :rtype: int
        """
        self.ans = 0
        def depth(r):
            if not r:
                return 0
```

```
            left, right = depth(r.left), depth(r.right)
            self.ans = max(self.ans, left + right)
            return 1 + max(left, right)
        depth(root)
        return self.ans

class Solution2:
    def diameterOfBinaryTree(self, root):
        """
        :type root: TreeNode
        :rtype: int
        """
        self.ans = 0
        self.depth(root)
        return self.ans

    def depth(self, root):
        if not root:
            return 0
        left, right = self.depth(root.left), self.depth(root.right)
        self.ans = max(self.ans, left + right)
        return 1 + max(left, right)
```

Solution for C++:

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
    int maxD = 0;
public:
    int diameterOfBinaryTree(TreeNode* root) {
        maxDepth(root);
        return maxD;
    }

    int maxDepth(TreeNode* root) {
        if (!root)
            return 0;
        int left = maxDepth(root->left);
        int right = maxDepth(root->right);
        maxD = max(maxD, left + right);
```

```
24              return max(left, right) + 1;
25          }
26      };
```