

剑指Offer（三十九）：平衡二叉树

🕒 2018年1月16日 17:40:32 🗨 15 🌡 4,244 °C 📄 编辑



一、前言

本系列文章为《剑指Offer》刷题笔记。

刷题平台：[牛客网](#)

书籍下载：[共享资源](#)

二、题目

输入一棵二叉树，判断该二叉树是否是平衡二叉树。

1、思路

平衡二叉树的定义是：所谓的平衡之意，就是树中任意一个结点下左右两个子树的高度差不超过 1。

解题思路有两种，只遍历一次的方法最优。

重复遍历多次：

在遍历树的每个结点的时候，调用函数TreeDepth得到它的左右子树的深度。如果每个结点的左右子树的深度相差都不超过1，则这是一颗平衡的二叉树。这种方法的缺点是，首先判断根结点是不是平衡的，需要使用TreeDepth获得左右子树的深度，然后还需要继续判断子树是不是平衡的，还是需要使用TreeDepth获得子树的左右子树的深度，这样就导致了大量的重复遍历。

只遍历一次：

重复遍历会影响算法的性能，所以很有必要掌握不需要重复遍历的方法。如果我们用后序遍历的方式遍历二叉树的每一个结点，在遍历到一个结点之前我们已经遍历了它的左右子树。只要在遍历每个结点的时候记录它的深度（某一结点的深度等于它到叶结点的路径的长度），我们就可以一边遍历一边判断每个结点是不是平衡的。

2、代码

C++：

重复遍历多次：

C+

```
1 class Solution {
2 public:
3     bool IsBalanced_Solution(TreeNode* pRoot) {
4         if(pRoot == NULL){
5             return true;
6         }
7         int left = TreeDepth(pRoot->left);
8         int right = TreeDepth(pRoot->right);
9         int diff = left - right;
10        if(diff > 1 || diff < -1){
11            return false;
12        }
13        return IsBalanced_Solution(pRoot->right) && IsBalanced_Solution(pRoot->left);
14    }
15 private:
16     int TreeDepth(TreeNode* pRoot)
17     {
```

```
18     if(pRoot == NULL){
19         return 0;
20     }
21     int left = TreeDepth(pRoot->left);
22     int right = TreeDepth(pRoot->right);
23     return (left > right) ? (left + 1) : (right + 1);
24 }
25 };
```

只遍历一次：


C+

```
1 class Solution {
2 public:
3     bool IsBalanced_Solution(TreeNode* pRoot) {
4         int depth = 0;
5         return IsBalanced(pRoot, &depth);
6     }
7 private:
8     int IsBalanced(TreeNode* pRoot, int* depth){
9         if(pRoot == NULL){
10             *depth = 0;
11             return true;
12         }
13         int left, right;
14         if(IsBalanced(pRoot->left, &left) && IsBalanced(pRoot->right, &right)){
15             int diff = left - right;
16             if(diff <= 1 && diff >= -1){
17                 *depth = 1 + (left > right ? left : right);
18                 return true;
19             }
20         }
21         return false;
22     }
23 };
```

感谢@小小毛提供的本地测试用例：

Pytho

```
1 class TreeNode:
2     def __init__(self, x):
3         self.val = x
4         self.left = None
5         self.right = None
6
7 class Solution:
8     def TreeDepth(self, root):
9         # write code here
10        if root is None:
11            return 0
12        left = self.TreeDepth(root.left)
13        right = self.TreeDepth(root.right)
14
15        if abs(left-right)>1:
16            return False
17        return max(left,right)+1
18
19 if __name__=='__main__':
20     A1 = TreeNode(1)
21     A2 = TreeNode(2)
22     A3 = TreeNode(3)
23     A4 = TreeNode(4)
24     A5 = TreeNode(5)
25     A6 = TreeNode(6)
26
27     A1.left=A2
28     A1.right=A3
29     A2.left=A4
30     A2.right=A5
31     A4.left=A6
32
33     solution=Solution()
34     ans=solution.TreeDepth(A1)
35     print('ans=',ans)
```



微信公众号

分享技术，乐享生活：微信公众号搜索「JackCui-AI」关注一个在互联网摸爬滚打的潜行者。

人因为有难忘的记忆而变得坚强，这就是所谓的成长吧。--- 纲手《火影忍者》