

LintCode领扣题解 (/problem) / 平衡二叉树 · Balanced Binary Tree

平衡二叉树 · **Balanced Binary Tree**

中文

Bloomberg (/problem/?tags=bloomberg)

分治法 (/problem/?tags=divide-and-conquer)

递归 (/problem/?tags=recursion)

描述

给定一个二叉树,确定它是高度平衡的。对于这个问题,一棵高度平衡的二叉树的定义是: 一棵二叉树中每个节点的两个子树的深度相差不会超过1。

样例

样例 1:

输入: tree = {1,2,3}

输出: true

样例解释:

如下, 是一个平衡的二叉树。

1

/ \

2 3

样例 2:

输入: tree = {3,9,20,#,#,15,7}

输出: true

样例解释:

如下, 是一个平衡的二叉树。

3

/ \

9 20

/ \

15 7

样例 2:

输入: tree = {1,#,2,3,4}

输出: false

样例解释:

如下, 是一个不平衡的二叉树。1的左右子树高度差2

1

\

2

/ \

3 4

邀请有礼

invitation/shi

在线评测地址: <https://www.lintcode.com/problem/balanced-binary-tree/> (https://www.lintcode.com/problem/balanced-binary-tree/)

收起题目描述 ^



令狐冲

更新于 10/30/2020, 5:14:15 AM

Given a binary tree, determine if it is height-balanced.

For this problem, a height-balanced binary tree is defined as a binary tree in which the depth of the two subtrees of every node never differ by more than 1.

在树上做一次DFS, 计算以每个点为根的子树高度。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
"""
Definition of TreeNode:
class TreeNode:
    def __init__(self, val):
        this.val = val
        this.left, this.right = None, None
"""

class Solution:
    """
    @param root: The root of binary tree.
    @return: True if this Binary tree is Balanced, or false.
    """
    def isBalanced(self, root):
        balanced, _ = self.validate(root)
        return balanced

    def validate(self, root):
        if root is None:
            return True, 0

        balanced, leftHeight = self.validate(root.left)
        if not balanced:
            return False, 0
        balanced, rightHeight = self.validate(root.right)
        if not balanced:
            return False, 0

        return abs(leftHeight - rightHeight) <= 1, max(leftHeight, rightHeight) + 1
```

👍 获赞 4

💬 3 条评论

# 你的口袋题库

2000+ 算法真题、国内外名企题库免费开放



九章算法APP



令狐冲

更新于 8/24/2020, 11:14:59 PM

Given a binary tree, determine if it is height-balanced.

For this problem, a height-balanced binary tree is defined as a binary tree in which the depth of the two subtrees of every node never differ by more than 1.

在树上做一次DFS, 记录以每个点为根的子树高度。

解法2中提供了更简洁的方法, 将子树高度作为返回值返回。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ /
Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
// Version 1: with ResultType
/**
 * Definition of TreeNode:
 * public class TreeNode {
 *     public int val;
 *     public TreeNode left, right;
 *     public TreeNode(int val) {
 *         this.val = val;
 *         this.left = this.right = null;
 *     }
 * }
 */
class ResultType {
    public boolean isBalanced;
    public int maxDepth;
    public ResultType(boolean isBalanced, int maxDepth) {
        this.isBalanced = isBalanced;
        this.maxDepth = maxDepth;
    }
}

public class Solution {
    /**
     * @param root: The root of binary tree.
     * @return: True if this Binary tree is Balanced, or false.
     */
    public boolean isBalanced(TreeNode root) {
        return helper(root).isBalanced;
    }

    private ResultType helper(TreeNode root) {
        if (root == null) {
            return new ResultType(true, 0);
        }

        ResultType left = helper(root.left);
        ResultType right = helper(root.right);



        // subtree not balance
        if (!left.isBalanced || !right.isBalanced) {
            return new ResultType(false, -1);
        }

        // root not balance
        if (Math.abs(left.maxDepth - right.maxDepth) > 1) {
            return new ResultType(false, -1);
        }

        return new ResultType(true, Math.max(left.maxDepth, right.maxDepth) + 1);
    }
}

// Version 2: without ResultType
public class Solution {
```

```
public boolean isBalanced(TreeNode root) {  
    return maxDepth(root) != -1;  
}  
  
private int maxDepth(TreeNode root) {  
    if (root == null) {  
        return 0;  
    }  
  
    int left = maxDepth(root.left);  
    int right = maxDepth(root.right);  
    if (left == -1 || right == -1 || Math.abs(left-right) > 1) {  
        return -1;  
    }  
    return Math.max(left, right) + 1;  
}  
}
```

 获赞 3 5 条评论

令狐冲

更新于 6/9/2020, 7:04:29 AM

Given a binary tree, determine if it is height-balanced.

For this problem, a height-balanced binary tree is defined as a binary tree in which the depth of the two subtrees of every node never differ by more than 1.

在树上做一次DFS, 计算以每个点为根的子树高度。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
/**
 * Definition of TreeNode:
 * class TreeNode {
 * public:
 *     int val;
 *     TreeNode *left, *right;
 *     TreeNode(int val) {
 *         this->val = val;
 *         this->left = this->right = NULL;
 *     }
 * }
 */
class Solution {
public:
    int depth(TreeNode *root) {
        if (root == NULL) {
            return 0;
        }
        int left = depth(root->left);
        int right = depth(root->right);
        if (left == -1 || right == -1 || abs(left - right) > 1) {
            return -1;
        }
        return max(left, right) + 1;
    }

    /**
     * @param root: The root of binary tree.
     * @return: True if this Binary tree is Balanced, or false.
     */
    bool isBalanced(TreeNode *root) {
        return depth(root) != -1;
    }
};
```

👍 获赞 0

💬 1 条评论



九章用户70C6D2

更新于 8/3/2020, 12:41:28 AM

计算左树高度lh, 右树高度rh, 比较高度差, 如果大于1, 返回false。如果小于1, 递归返回左子树是否平衡 且 右子树是否平衡

```
/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
public class Solution {
    /**
     * @param root: The root of binary tree.
     * @return: True if this Binary tree is Balanced, or false.
     */
    public boolean isBalanced(TreeNode root) {
        // write your code here
        if (root == null) return true;

        int lh = getHeight(root.left);
        int rh = getHeight(root.right);

        if (Math.abs(lh - rh) > 1) {
            return false;
        }

        return (isBalanced(root.left) && isBalanced(root.right));
    }

    int getHeight(TreeNode node) {
        if (node == null) return 0;

        int lh = getHeight(node.left);
        int rh = getHeight(node.right);

        return Math.max(lh, rh) + 1;
    }
}
```

👍 获赞 8

💬 4 条评论

**九章用户G1LOUJ**

更新于 10/30/2020, 5:14:20 AM

Python3 分治法 分别判断一棵二叉树的左子树，右子树是否平衡，再判断两个子树的高度差是否小于等于1 注意：本题解不是最优的代码，因为单独求二叉树的高度会有很多重复计算，仅供初学者先理解分治。更优的答案请参考精选中的Python答案

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
"""
Definition of TreeNode:
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left, self.right = None, None
"""

class Solution:
    """
    @param root: The root of binary tree.
    @return: True if this Binary tree is Balanced, or false.
    """
    def isBalanced(self, root):
        if not root:
            return True

        if not self.isBalanced(root.left):
            return False
        if not self.isBalanced(root.right):
            return False

        return abs(self.get_height(root.left) - self.get_height(root.right)) <= 1

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

👍 获赞 6

💬 2 条评论

**Tin**

更新于 12/23/2020, 5:19:41 AM

既然这题要求判定是否平衡二叉树, 输入就有可能不是平衡二叉树, 递归栈的深度就会是  $N$ , 造成栈溢出。

这题在面试中用递归是绝对通不过的。

给一个  $O(n)$  时间,  $O(n)$  空间的非递归解法, 适合Python用过至少半年以上的同学。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution:

    def isBalanced(self, root):

        stack, depth = [], collections.defaultdict(int)
        while root or stack:
            if root:
                stack.append(root)
                root = root.left if root.left else root.right
            else:
                root = stack.pop()
                if abs(depth[root.left] - depth[root.right]) > 1:
                    return False
                depth[root] = 1 + max(depth[root.left], depth[root.right])
                if stack and stack[-1].left is root:
                    root = stack[-1].right
                else:
                    root = None
        return True
```

 获赞 3 添加评论

马同学

更新于 6/9/2020, 7:03:50 AM

用了令狐老师在预习教程中引入ResultType类的方法, 改写为Python版本。要注意在Python中不需要声明返回值。



```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
"""
Definition of TreeNode:
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left, self.right = None, None
"""

class ResultType:
    def __init__(self, depth, IsBalanced):
        self.depth = depth
        self.IsBalanced = IsBalanced

class Solution:
    """
    @param root: The root of binary tree.
    @return: True if this Binary tree is Balanced, or false.
    """
    def isBalanced(self, root):
        # write your code here
        # return 1. BalancedBinary or not? 2. height
        return self.maxDepth(root).IsBalanced

# 1. balanced: return height, not balanced, return false
def maxDepth(self, root):
    if root is None:
        return ResultType(0, True)

    left = self.maxDepth(root.left)
    right = self.maxDepth(root.right)

    if not left.IsBalanced or not right.IsBalanced:
        return ResultType(-1, False)

    if abs(left.depth - right.depth) > 1:
        return ResultType(-1, False)

    return ResultType(max(left.depth, right.depth) + 1, True)

```

👍 获赞 2

💬 3 条评论



九章令狐冲

更新于 11/8/2020, 2:00:13 AM

Given a binary tree, determine if it is height-balanced.

For this problem, a height-balanced binary tree is defined as a binary tree in which the depth of the two subtrees of every node never differ by more than 1.

在树上做一次DFS, 计算以每个点为根的子树高度。

python

java

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    """
    @param root: The root of binary tree.
    @return: True if this Binary tree is Balanced, or false.
    """
    def isBalanced(self, root):
        is_balanced, _ = self.helper(root)
        return is_balanced

    def helper(self, root):
        if not root:
            return True, 0

        is_left_balanced, left_height = self.helper(root.left)
        is_right_balanced, right_height = self.helper(root.right)

        root_height = max(left_height, right_height) + 1

        if not is_left_balanced or not is_right_balanced:
            return False, root_height

        if abs(left_height - right_height) > 1:
            return False, root_height

        return True, root_height
```

👍 获赞 1    💬 添加评论



**Keven**

更新于 8/1/2020, 9:46:07 PM

和老师的答案类似, 使用了 global variable 用两个DFS分别得到左子树和右子树的深度, 然后进行比较。

```
/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
public class Solution {
    /**
     * @param root: The root of binary tree.
     * @return: True if this Binary tree is Balanced, or false.
     */

    boolean isBalanced = true;
    public boolean isBalanced(TreeNode root) {
        // write your code here

        dfs(root);
        return isBalanced;
    }

    private int dfs(TreeNode root) {
        if (root == null) {
            return 0;
        }

        int leftDepth = dfs(root.left);
        int rightDepth = dfs(root.right);

        if (Math.abs(leftDepth - rightDepth) > 1) {
            isBalanced = false;
        }

        return Math.max(leftDepth, rightDepth) + 1;
    }
}
```

👍 获赞 1

💬 添加评论



社会我喵哥

更新于 6/9/2020, 7:04:23 AM

还是利用分治法，构思的时候需要返回一个returnType，包括子树的高度和平衡性，我这里利用了python的元组tuple，省去了重新定义一个新的类。C++的话可以用pair实现类似效果

```

/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
"""
Definition of TreeNode:
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left, self.right = None, None
"""

class Solution:
    """
    @param root: The root of binary tree.
    @return: True if this Binary tree is Balanced, or false.
    """
    def isBalanced(self, root):
        # write your code here
        result = self.helper(root)
        return result[1]
    def helper(self, root):
        if root is None:
            return (0, True)
        leftResult = self.helper(root.left)
        rightResult = self.helper(root.right)
        height = max(leftResult[0], rightResult[0]) + 1
        balance1 = leftResult[1] and rightResult[1]
        balance2 = True if abs(leftResult[0] - rightResult[0]) < 2 else False
        return (height, balance1 and balance2 )

```

👍 获赞 0    💬 添加评论



九章用户51HVT8

更新于 6/9/2020, 7:04:22 AM

我个人有时候对递归不太敏感，因为我有时候后套进去就出不来了，我猜好多人大概和我一个情况，所以令狐冲老师的代码当时有点没看明白，所以自己想了一个比较笨的递归写法，大概思路是一样的吗？

有一个findheight helper 目的就是找到当前节点往后的最大高度，重点就是主方程先调用看root两边平衡不，然后单独拆分每一个左右subtree,再开始新一轮的对比。话说这个helper 我也没用递归，用的BFS。。。题外话，佩服Python题解的两个神人，早上6点就起来刷题的，那我绝对是夜猫子(ω \ )

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution {
public:
    queue<TreeNode*> Q;
    bool isBalanced(TreeNode* root)
    {
        if(root == NULL)
            return true;
        if(abs(findheight(root->left) - findheight(root->right)) > 1)
            return false;
        if(isBalanced(root->left) && isBalanced(root->right) != false)
            return true;
    }

    int findheight(TreeNode* root)
    {
        if(root == NULL)
        {
            return 0;
        }
        Q.push(root);
        int deep = 0;
        while(!Q.empty())
        {
            int size = Q.size();
            for(int i = 0; i < size; i++)
            {
                TreeNode* temp = Q.front();
                Q.pop();
                if(temp->left)
                    Q.push(temp->left);
                if(temp->right)
                    Q.push(temp->right);
            }
            deep += 1;
        }
        return deep;
    }
};
```

👍 获赞 0    💬 添加评论



n同学

更新于 6/9/2020, 7:04:20 AM

虽然题目简单, 可是在写递归函数时需要返回FALSE或左右两树最深高度, 递归需多练习

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ /
Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
"""
Definition of TreeNode:
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left, self.right = None, None
"""

class Solution:
    """
    @param root: The root of binary tree.
    @return: True if this Binary tree is Balanced, or false.
    """
    def isBalanced(self, root):
        # write your code here
        if not root:
            return True

        return not not self.dfs(root, 0)

    def dfs(self, root, depth):
        if not root:
            return depth

        depth += 1
        left = self.dfs(root.left, depth)
        right = self.dfs(root.right, depth)

        if abs(left - right) > 1:
            return False

        return False if left == False or right == False else max(left, right)
```

👍 获赞 0    💬 添加评论



**jerron**

更新于 6/9/2020, 7:04:11 AM

shortcut: if any unbalanced subtree is found, no need to continue but return false immediately

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
/**
 * Definition of TreeNode:
 * class TreeNode {
 * public:
 *     int val;
 *     TreeNode *left, *right;
 *     TreeNode(int val) {
 *         this->val = val;
 *         this->left = this->right = NULL;
 *     }
 * }
 */

class Solution {
public:
    /**
     * @param root: The root of binary tree.
     * @return: True if this Binary tree is Balanced, or false.
     */
    bool isBalanced(TreeNode * root) {
        // write your code here
        function<int(TreeNode*)>dfs;
        dfs=[&dfs](TreeNode* root)->int{
            if(root){
                int depthl=dfs(root->left),depth2;
                if(depthl<0||(depth2=dfs(root->right))<0||
                    depth2>depthl+1||depthl>depth2+1)
                    return -1;
                return max(depthl,depth2)+1;
            }
            return 0;
        };
        return dfs(root)>=0;
    }
};
```

👍 获赞 0

💬 添加评论

**Wade**

更新于 6/9/2020, 7:04:10 AM

题解没有那么复杂, 同样是返回值需要两元, 这里用pair类型解决, 代码简洁高效

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution {
public:
    bool isBalanced(TreeNode* root) {
        return dfs(root).first;
    }

    pair<bool, int> dfs(TreeNode* root) {
        if (!root) return {true, 0};
        auto l = dfs(root->left);
        auto r = dfs(root->right);
        pair<bool, int> ans = {true, max(l.second, r.second) + 1};
        ans.first &= l.first && r.first && (abs(l.second - r.second) <= 1);
        return ans;
    }
};
```

👍 获赞 0

💬 添加评论



c同学

更新于 6/9/2020, 7:04:07 AM

分别计算左右的高度, 如果不是balanced 就直接返回-1; 如过是balanced, 就返回左右子数里面最大的那个再加 1



```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
public class Solution {
    /**
     * @param root: The root of binary tree.
     * @return: True if this Binary tree is Balanced, or false.
     */
    public boolean isBalanced(TreeNode root) {
        if (root == null) {
            return true;
        }
        return getHeight(root) != -1;
    }
    private int getHeight(TreeNode root){
        if (root == null){
            return 0;
        }
        int leftHeight = getHeight(root.left);
        if (leftHeight == -1){
            return -1;
        }
        int rightHeight = getHeight(root.right);
        if (rightHeight == -1){
            return -1;
        }
        if (Math.abs(leftHeight - rightHeight) > 1){
            return -1;
        }
        return Math.max(leftHeight, rightHeight) + 1;
    }
}

```

👍 获赞 0

💬 添加评论

**Jet**

更新于 6/9/2020, 7:04:04 AM

/\*\*

- Definition of TreeNode:
- class TreeNode {
- public:

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
/**
 * Definition of TreeNode:
 * class TreeNode {
 * public:

```

```
*   int val;
*   TreeNode *left, *right;
*   TreeNode(int val) {
*       this->val = val;
*       this->left = this->right = NULL;
*   }
* }
*/

class Solution {
public:
    /**
     * @param root: The root of binary tree.
     * @return: True if this Binary tree is Balanced, or false.
     */
    bool isBalanced(TreeNode * root) {
        return maxDepth(root)!=-1;
    }
    int maxDepth(TreeNode* root){
        if(root==nullptr){
            return 0;
        }
        int left=maxDepth(root->left);
        int right=maxDepth(root->right);
        if(left==-1){
            return -1;
        }
        if(right==-1){
            return -1;
        }
        if(abs(left-right)>1){
            return -1;
        }
        return max(left,right)+1;
    }

    bool isBalanced_first(TreeNode * root) {
        if(root==nullptr){return true;}
        // write your code here

        if(!(isBalanced(root->left) ) ){
            return false;
        }
        if(!(isBalanced(root->right))){
            return false;
        }
        int dif=max_height(root->left)-max_height(root->right);
        if(!(dif>=-1&&dif<=1)){return false;}
        return true;
    }

    int max_height(TreeNode* root){
        if(root==nullptr){
            return 0;
        }
        int l=max_height(root->left);
        int r=max_height(root->right);
        return max(l,r)+1;
    }
};
```

👍 获赞 0    💬 添加评论

**Bin.VII**

更新于 6/9/2020, 7:04:01 AM

1. 比较好理解的代码, 一个是判断是不是balanced, 一个获得其深度, 而为了不增加重复计算, 引入记忆化搜索
2. 使用Result类, 然后divide conquer方法比较左右两棵子树

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
// 记忆化搜索 + DFS (balance + depth)
public class Solution {
    Map<TreeNode, Integer> memo = new HashMap<>();
    public boolean isBalanced(TreeNode root) {
        if (root == null) {
            return true;
        }
        int left = getDepth(root.left, memo);
        int right = getDepth(root.right, memo);
        return isBalanced(root.left) && isBalanced(root.right) && Math.abs(left - right) <= 1;
    }

    private int getDepth(TreeNode root, Map<TreeNode, Integer> memo) {
        if (root == null) {
            return 0;
        }
        if (memo.containsKey(root)) {
            return memo.get(root);
        }
        int left = getDepth(root.left, memo);
        int right = getDepth(root.right, memo);
        int depth = Math.max(left, right) + 1;
        memo.put(root, depth);
        return depth;
    }
}

// Divide Conquer
public class Solution {
    class Result {
        int depth;
        boolean isBalanced;
        Result(int depth, boolean isBalanced) {
            this.depth = depth;
            this.isBalanced = isBalanced;
        }
    }

    public boolean isBalanced(TreeNode root) {
        // write your code here
        return divideConquer(root).isBalanced;
    }

    private Result divideConquer(TreeNode root) {
        if (root == null) {
            return new Result(0, true);
        }

        Result left = divideConquer(root.left);
```

```
Result right = divideConquer(root.right);

if (!left.isBalanced || !right.isBalanced || Math.abs(left.depth - right.depth) > 1) {
    return new Result(-1, false);
}

return new Result(Math.max(left.depth, right.depth) + 1, true);
}
```

👍 获赞 0

💬 添加评论

## 进阶课程

视频+互动

直播+互动

直播+互动

互动课

### 九章算法班 2021 版

8周时间精通 57 个核心高频考点, 9  
招击破 FLAG、BATJ 算法面试。22....

### 系统架构设计 System Design 2021 版

成为百万架构师必上。30 课时带你快  
速掌握 18 大系统架构设计知识点与面...

### 九章算法面试高频题冲刺班

每期更新 15% 题目, 考前押题, 一举  
拿下 FLAG & BATJ Offer



### 面向对象设计 OOD

应届生及亚马逊面试必考, IT 求职必备  
基础

[首页 \(/?skip\\_redirect=true\)](#) | [联系我们 \(mailto:info@jiuzhang.com\)](mailto:info@jiuzhang.com) | [加入我们 \(/joinus\)](#)

Copyright © 2013-2020 九章算法 浙ICP备19045946号-1  
(<http://www.miibeian.gov.cn/>)

商务合作: [fukesu@jiuzhang.com](mailto:fukesu@jiuzhang.com) (<mailto:fukesu@jiuzhang.com>)

 (<http://weibo.com/ninechapter>)  知  
(<https://www.zhihu.com/people/crackinterview/>)

(0)