# 题目

实现一个函数，检查二叉树是否平衡。在这个问题中，平衡树的定义如下：任意一个节点，其两棵子树的高度差不超过 1。

**示例 1:**

给定二叉树 [3,9,20,null,null,15,7]

3

/ \

9 20

/ \

15 7

返回 true 。

**示例 2:**

给定二叉树 [1,2,2,3,3,null,null,4,4]

1

/ \

2 2

/ \

3 3

/ \

4 4

返回 false 。

# 分析

## 方法一：递归法

**思路：**

前一题是求二叉树的深度，这一题其实是求二叉树的深度差不超过 1，本质上是一样的，建议先看一下上一题，这一题就是先看根节点的左右子树是否平衡，然后再看左右子树的左右子树是否平衡。

**代码：**

/\*\*

\* 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 {

public:

int depth(TreeNode\* root)//计算以root为根节点的子树的最大深度

{

if(root==NULL) return 0;

int left=depth(root->left);

int right=depth(root->right);

return max(left,right)+1;

}

bool isBalanced(TreeNode\* root) {

if(root==NULL) return true;

if(abs(depth(root->left) - depth(root->right)) < 2)//如果根节点的左右子树高度差小于2

return isBalanced(root->left) && isBalanced(root->right);//继续递归判断左孩子右孩子

return false;

}

};