```swift
import Foundation
// 剑指 Offer 55 - II. 平衡二叉树
//
 输入一棵二叉树的根节点，判断该树是不是平衡二叉树。如果某二叉树中任意节点的左右子树的深度相差不超
 过1，那么它就是一棵平衡二叉树。
//
// 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 。
public class TreeNode {
    public var val: Int
    public var left: TreeNode?
    public var right: TreeNode?
    public init(_ val: Int) {
        self.val = val
        self.left = nil
        self.right = nil
    }
}

class Solution {
    /// 执行用时：36 ms, 在所有 Swift 提交中击败了 56.10% 的用户
    /// 内存消耗：14.7 MB, 在所有 Swift 提交中击败了 29.27% 的用户
    /// 通过测试用例：39 / 39
    func isBalancedSolution1(_ root: TreeNode?) -> Bool {
        if root == nil { return true }
        return ((abs(maxDepth(root?.left) - maxDepth(root?.right)) <= 1) &&
                (isBalanced(root?.left)) && (isBalanced(root?.right)))
    }

    func maxDepth(_ root: TreeNode?) -> Int {
```

```
        if root == nil { return 0 }
        return max(maxDepth(root?.left), maxDepth(root?.right)) + 1
    }

    /// 执行用时：36 ms, 在所有 Swift 提交中击败了 56.10% 的用户
    /// 内存消耗：14.7 MB, 在所有 Swift 提交中击败了 29.27% 的用户
    /// 通过测试用例：39 / 39
    func isBalanced(_ root: TreeNode?) -> Bool {
        return recur(root) != -1
    }

    func recur(_ root: TreeNode?) -> Int {
        if root == nil { return 0 }
        let left = recur(root?.left)
        if left == -1 { return -1 }
        let right = recur(root?.right)
        if right == -1 { return -1 }
        return abs(left-right) <= 1 ? max(left, right)+1 : -1
    }
}
```