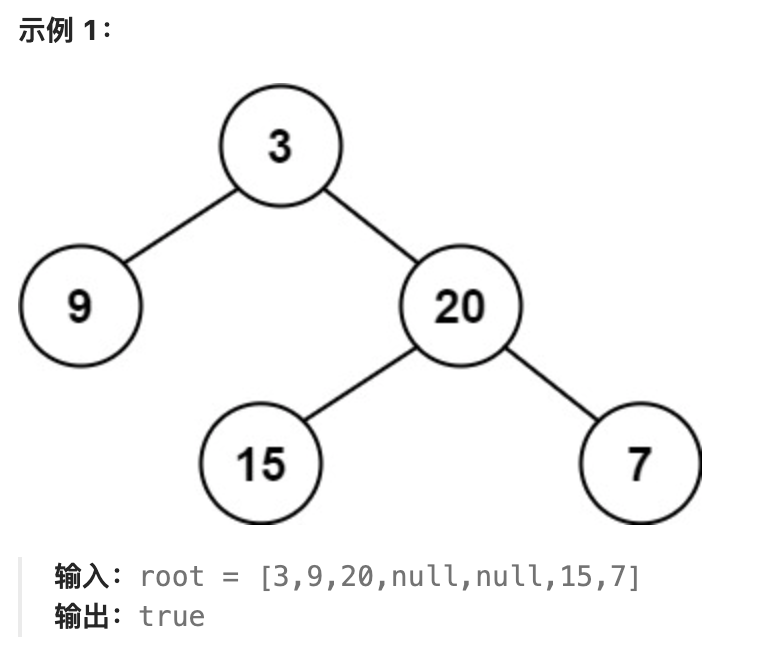
# 第五次算法设计作业

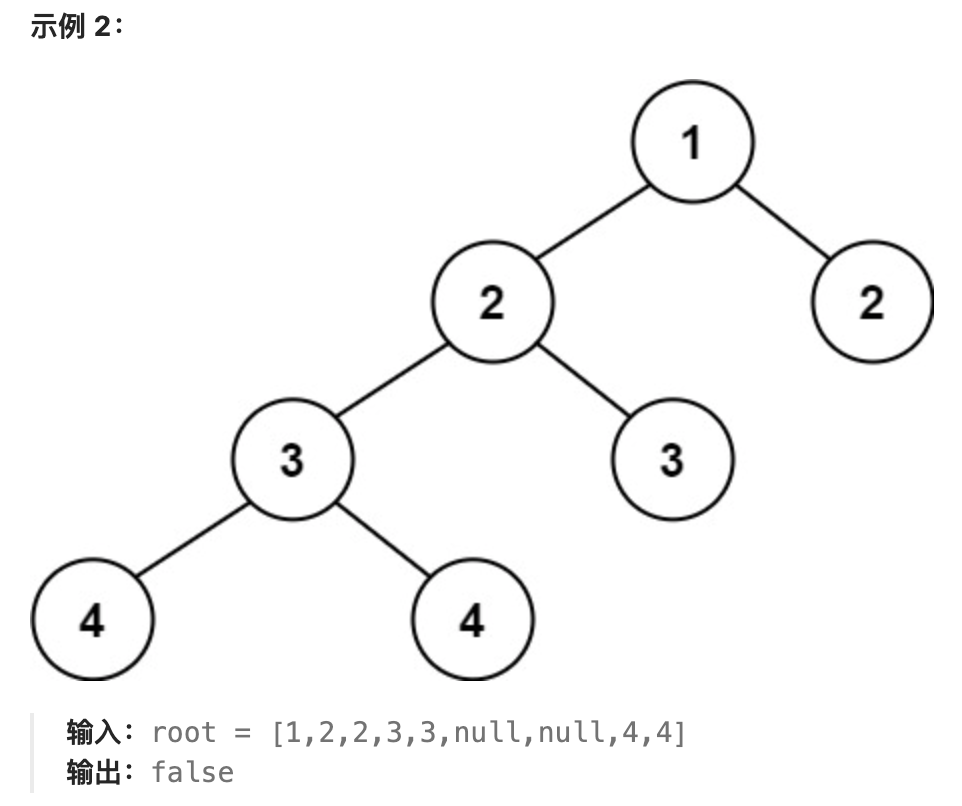
姓名： 学号：

【说明】共两道题，至少完成一题，全完成可加分。解答应包括解题思路（方法、时间复杂度分析）、代码以及按结果验证中的输入得到的输出截图。作业提交时请只保留选做的题，删除空白题。

【题一】给定一个二叉树，判断它是否是平衡二叉树

【示例】





【解题思路】

这段代码是一个用于检查二叉树是否平衡的Java类。一个平衡二叉树（Balanced Binary Tree）是指一个二叉树，其中任何两个子树的高度差不超过1。代码中使用了递归的方法来实现这一检查。

下面是代码的逻辑步骤：

1. isBalanced 方法：这是公共接口方法，它调用私有的 depth 方法来检查树是否平衡。如果 depth 方法返回 -1，则表示树不平衡，否则表示树是平衡的。
2. depth 方法：这是一个递归方法，用于计算二叉树的高度，并同时检查树是否平衡。
   * 如果传入的节点 node 为 null（即空），则返回0，因为空树的高度是0。
   * 递归地计算左子树 node.left 和右子树 node.right 的高度，分别存储在变量 l 和 r 中。
   * 如果 l 或 r 为 -1，表示子树不平衡，因此整个树也不平衡，返回 -1。
   * 如果 l 和 r 的差的绝对值大于1，表示当前节点的两个子树高度差超过了1，树不平衡，返回 -1。
   * 如果上述条件都不满足，说明当前节点的两个子树都是平衡的，返回两个子树中较高的那个的高度加1。

通过这种递归方法，depth 方法在计算树的高度的同时，也检查了树的平衡性。如果树是平衡的，depth 方法将返回树的高度；如果树不平衡，将返回 -1。isBalanced 方法根据 depth 方法的返回值来判断树是否平衡，并返回相应的布尔值。

【代码】

*import* my\_package.TreeNode;  
  
*public class* T110 {  
 *static class* Solution {  
 *public boolean* isBalanced(TreeNode root) {  
 *return* depth(root) != -1;  
 }  
  
 *private int* depth(TreeNode node) {  
 *if* (node == *null*) *return* 0;  
 *int* l = depth(node.left), r = depth(node.right);  
 *if* (l == -1 || r == -1 || Math.*abs*(l - r) > 1) *return* -1;  
 *return* Math.*max*(l, r) + 1;  
 }  
 }  
  
 *public static void* main(String[] args) {  
 Solution solution = *new* Solution();  
 TreeNode root = *new* TreeNode(1,  
 *new* TreeNode(2,  
 *new* TreeNode(3,  
 *new* TreeNode(4),  
 *new* TreeNode(4)),  
 *new* TreeNode(3)),  
 *new* TreeNode(2));  
 System.***out***.println(solution.isBalanced(root));  
 }  
}

【结果验证】1、输入：root = [1,2,2,3,3,null,null,4,4]

1. 输入：root = []

