0110. 平衡二叉树

ITCharge 大约 1 分钟

- 标签：树、深度优先搜索、二叉树
- 难度：简单

题目链接

- 0110. 平衡二叉树 - 力扣

题目大意

描述：给定一个二叉树的根节点 `root` 。

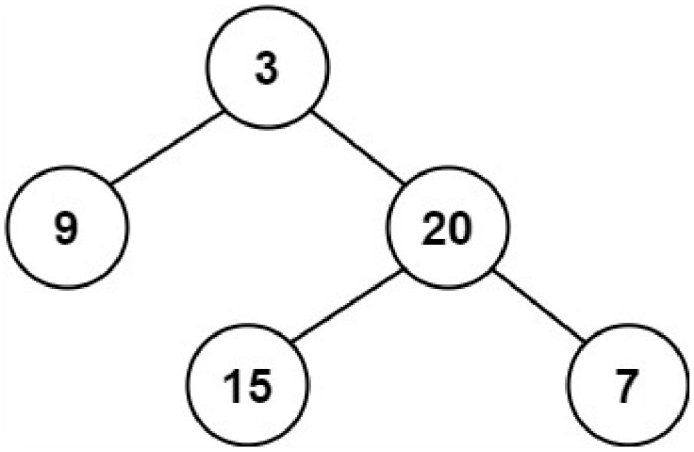
要求：判断该二叉树是否是高度平衡的二叉树。

说明：

- **高度平衡二叉树**：二叉树中每个节点左右两个子树的高度差的绝对值不超过 1。
- 树中的节点数在范围 $[0, 5000]$ 内。
- $-10^4 \leq Node.val \leq 10^4$。

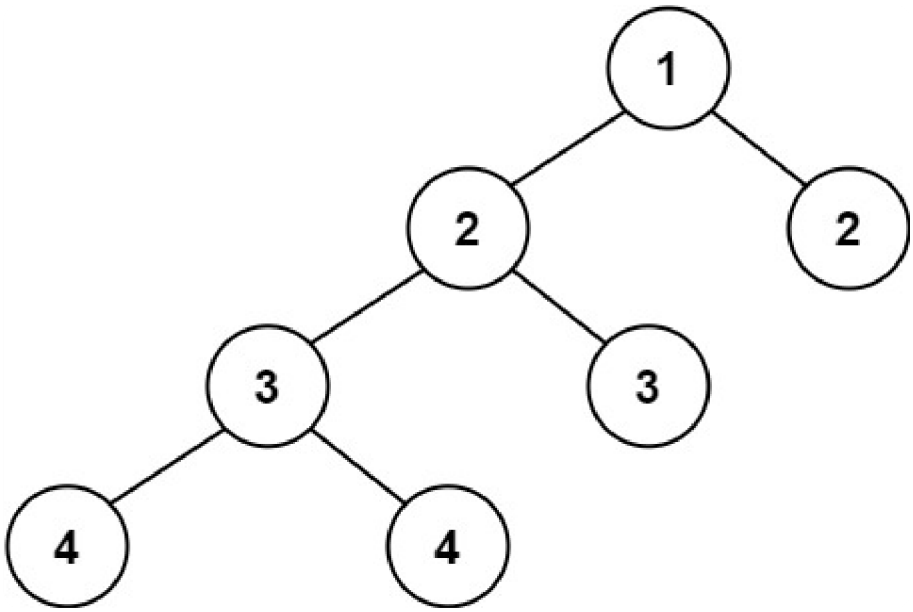
示例：

- 示例 1：



```py
输入：root = [3,9,20,null,null,15,7]
输出：True
```

- 示例 2：




```py
输入：root = [1,2,2,3,3,null,null,4,4]
输出：False
```

解题思路

思路 1：递归遍历

1. 先递归遍历左右子树，判断左右子树是否平衡，再判断以当前节点为根节点的左右子树是否平衡。
2. 如果遍历的子树是平衡的，则返回它的高度，否则返回 -1。
3. 只要出现不平衡的子树，则该二叉树一定不是平衡二叉树。

思路 1：代码

```py
class Solution:
    def isBalanced(self, root: TreeNode) -> bool:
        def height(root: TreeNode) -> int:
            if root == None:
                return False
            leftHeight = height(root.left)
            rightHeight = height(root.right)
            if leftHeight == -1 or rightHeight == -1 or abs(leftHeight-
rightHeight) > 1:
                return -1
            else:
                return max(leftHeight, rightHeight)+1
        return height(root) >= 0
```

思路 1：复杂度分析

- **时间复杂度**：$O(n)$，其中 n 是二叉树的节点数目。
- **空间复杂度**：$O(n)$。递归函数需要用到栈空间，栈空间取决于递归深度，最坏情况下递归深度为 n，所以空间复杂度为 $O(n)$。