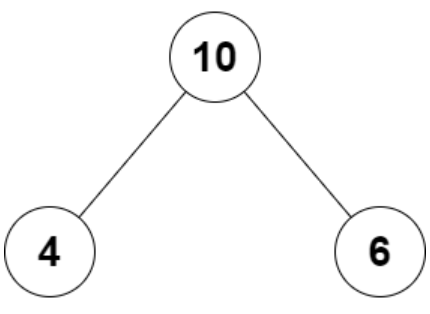
# 题目

给你一个二叉树的根结点root，该二叉树由恰好3个结点组成：根结点、左子结点和右子结点。

如果根结点值等于两个子结点值之和，返回true，否则返回false。

示例 1：



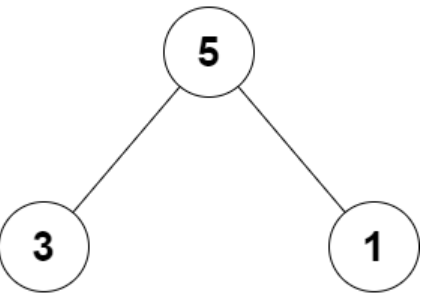
输入：root = [10,4,6]

输出：true

解释：根结点、左子结点和右子结点的值分别是10、4和6。

由于10等于4 + 6，因此返回true。

示例 2：



输入：root = [5,3,1]

输出：false

解释：根结点、左子结点和右子结点的值分别是5、3和1。

由于5不等于3 + 1，因此返回false。

提示：

树只包含根结点、左子结点和右子结点

-100 <= Node.val <= 100

# 分析

## 方法一：直接判断

思路：

计算两个子结点值之和，判断是否等于根结点值即可。

代码：

class Solution {

public:

bool checkTree(TreeNode\* root) {

if (root->val == root->left->val + root->right->val) {

return true;

}

return false;

}

};

复杂度分析：

时间复杂度：O(1)。

空间复杂度：O(1)。

如果不是3节点的二叉树，则算法如下：

class Solution {

public:

bool checkTree(TreeNode\* root) {

if (!root) return true;

return isSumTreeHelper(root) != -1;

}

private:

int isSumTreeHelper(TreeNode\* node) {

if (!node) return 0;

int leftSum = isSumTreeHelper(node->left);

int rightSum = isSumTreeHelper(node->right);

if (leftSum == -1 || rightSum == -1) return -1;

if ((node->left && node->val != node->left->val + leftSum) ||

(node->right && node->val != node->right->val + rightSum)) {

return -1;

}

return node->val + leftSum + rightSum;

}

};