# 题目

给定一个二叉搜索树的 根节点 root 和一个整数 k , 请判断该二叉搜索树中是否存在两个节点它们的值之和等于 k 。假设二叉搜索树中节点的值均唯一。

示例 1：

输入: root = [8,6,10,5,7,9,11], k = 12

输出: true

解释: 节点 5 和节点 7 之和等于 12

示例 2：

输入: root = [8,6,10,5,7,9,11], k = 22

输出: false

解释: 不存在两个节点值之和为 22 的节点

提示：

二叉树的节点个数的范围是  [1, 104].

-104 <= Node.val <= 104

root 为二叉搜索树

-105 <= k <= 105

注意：本题与主站 653 题相同：

https://leetcode-cn.com/problems/two-sum-iv-input-is-a-bst/

# 分析

## 方法一：递归法

**思路：**

遍历的同时记录在hash表里

**代码：**

class Solution {

public:

unordered\_set<int> set;

bool findTarget(TreeNode\* root, int k) {

if (root == nullptr) return false;

if (set.find(k - root->val) != set.end()) return true;

set.insert(root->val);

return findTarget(root->left, k) || findTarget(root->right, k);

}

};

## 方法二：深度遍历+双指针

**思路：**

转化为有序数组是否存在两个节点它们的值之和等于k

**代码：**

class Solution {

private:

vector<int> nums;

// 中序遍历

void dfs(TreeNode\* root) {

if (root->left) dfs(root->left);

nums.push\_back(root->val);

if (root->right) dfs(root->right);

}

public:

bool findTarget(TreeNode\* root, int k) {

if (root == nullptr) return false;

dfs(root);

int left = 0, right = nums.size() - 1;

// 双指针，典中典

while (left < right) {

if (nums[left] + nums[right] < k) {

left++;

} else if (nums[left] + nums[right] > k) {

right--;

} else return true;

}

return false;

}

};