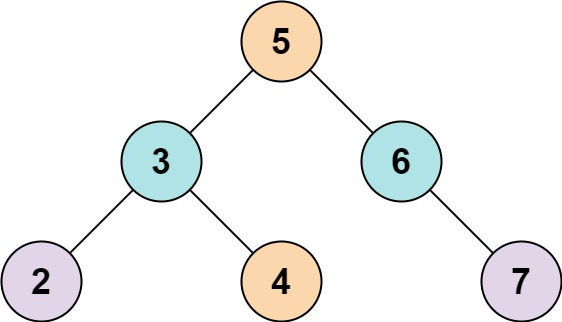
# 题目

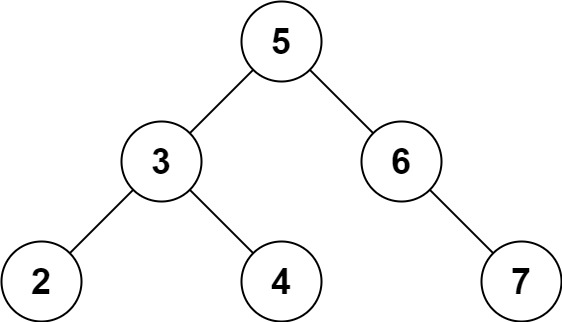
给定一个二叉搜索树root和一个目标结果k，如果BST中存在两个元素且它们的和等于给定的目标结果，则返回true。

示例 1：



输入: root = [5,3,6,2,4,null,7], k = 9输出: true

示例 2：



输入: root = [5,3,6,2,4,null,7], k = 28

输出: false

示例 3：

输入: root = [2,1,3], k = 4

输出: true

示例 4：

输入: root = [2,1,3], k = 1

输出: false

示例 5：

输入: root = [2,1,3], k = 3

输出: true

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

# 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;

}

};