# 653 Two Sum IV - Input is a BST

2018年4月22日 13:07

Given a Binary Search Tree and a target number, return true if there exist two elements in the BST such that their sum is equal to the given target.

**Example 1:**

**Input:**

```
    5
   / \
  3   6
 / \   \
2   4   7
```

Target = 9

**Output:** True

**Example 2:**

**Input:**

```
    5
   / \
  3   6
 / \   \
2   4   7
```

Target = 28

**Output:** False

来自 <https://leetcode.com/problems/two-sum-iv-input-is-a-bst/description/>

给定一个二叉搜索树和一个目标结果，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。

# Solution for Python3:

```python
class Solution1:
    def findTarget(self, root, k):
        """
        :type root: TreeNode
        :type k: int
        :rtype: bool
        """
        s = set()
        return self.find(root, k, s)

    def find(self, root, k, s):
        if not root:
            return False
        if k - root.val in s:
            return True
        s.add(root.val)
        return self.find(root.left, k ,s) or self.find(root.right, k, s)
```

```python

class Solution2:
    def findTarget(self, root, k):
        """
        :type root: TreeNode
        :type k: int
        :rtype: bool
        """
        if not root:
           return False
        bfs, s = [root], set()
        for i in bfs:
           if k - i.val in s:
               return True
           s.add(i.val)
           if i.left:
               bfs.append(i.left)
           if i.right:
               bfs.append(i.right)
        return False
```

## Solution for C++:

```cpp
class Solution1 {
public:
    bool findTarget(TreeNode* root, int k) {
        unordered_set<int> set;
        return find(root, k, set);
    }
    bool find(TreeNode* root, int k, unordered_set<int>& set) {
        if (root == NULL)
            return false;
        if (set.count(k - root->val))
            return true;
        set.insert(root->val);
        return find(root->left, k, set) || find(root->right, k, set);
    }
};

class Solution2 {
public:
    bool findTarget(TreeNode* root, int k) {
        unordered_set<int> set;
        queue<TreeNode*> que;
        que.push(root);
        while (!que.empty()) {
            TreeNode* node = que.front();
```

```
            que.pop();
            if (set.count(k - node->val))
                return true;
            set.insert(node->val);
            if (node->left)
                que.push(node->left);
            if (node->right)
                que.push(node->right);
        }
        return false;
    }
};

class Solution3 {
public:
    bool findTarget(TreeNode* root, int k) {
        vector<int> v;
        inorder(root, v);
        int l = 0, r = v.size() - 1;
        while (l < r) {
            int sum = v[l] + v[r];
            if (sum == k)
                return true;
            if (sum < k)
                l++;
            else
                r--;
        }
        return false;
    }
    void inorder(TreeNode* root, vector<int>& v) {
        if (root == NULL)
            return;
        inorder(root->left, v);
        v.push_back(root->val);
        inorder(root->right, v);
    }
};
```