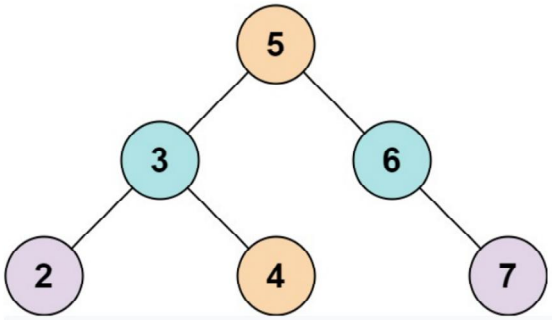
653. Two Sum IV - Input is a BST

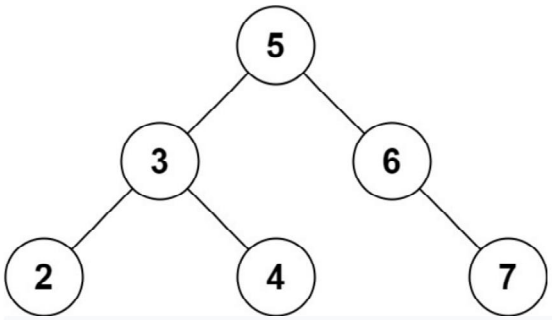
03 April 2022 05:16 PM

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

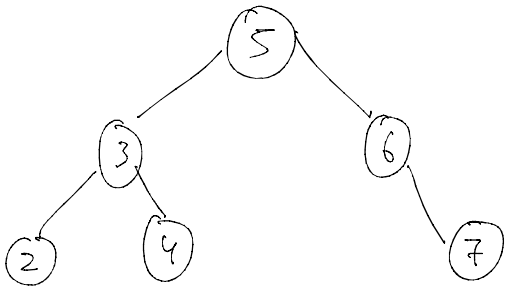
Example 1:




```
Input: root = [5,3,6,2,4,null,7], k = 9
Output: true
```




```
Input: root = [5,3,6,2,4,null,7], k = 28
Output: false
```



k = 9

O/P ⇒ True

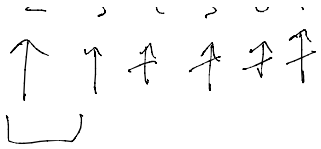
1) Brute force

Inorder Traversal ⟶ Sorted in nature ⟶ then use two pointer

k = 5

2 3 4 5 6 7

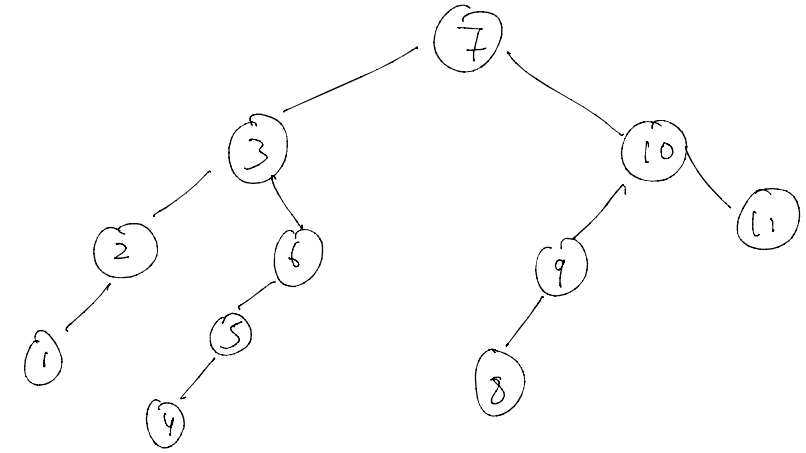
↑ ↑ ↑ ↑ ↑ ↑



if it greater we will reduce greater nos.

T.C ⇒ O(n) + O(n) S.C ⇒ O(n)

k = 16



Using the BST iterator

next() → st.top()

next ⇒ 1

" ⇒ 2

" ⇒ 3

" ⇒ 4

for before you push all right

right node left → desc sorted

before → push everything on right

↳ st.top() → left → all rights

k = 16

i = next()

j = before()

11

11+5=16

1

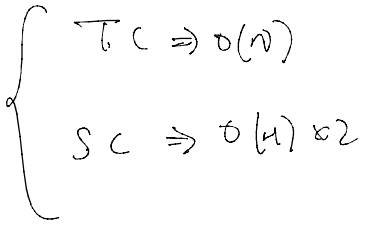
2

3

4

5

ans




```
public class BSTIterator {
    private Stack<TreeNode> stack = new Stack<TreeNode>();
    // reverse -> true -> before
    // reverse -> false -> next
    boolean reverse = true;       // before

    public BSTIterator(TreeNode root, boolean isReverse) {
        reverse = isReverse;
        pushAll(root);
    }

    /** @return whether we have a next smallest number */
    public boolean hasNext() {
        return !stack.isEmpty();
    }


    /** @return the next smallest number */
    public int next() {
        TreeNode tmpNode = stack.pop();
        if(reverse == false) pushAll(tmpNode.right);
        else pushAll(tmpNode.left);
        return tmpNode.val;
    }


    private void pushAll(TreeNode node) {
        while(node != null) {
             stack.push(node);
             if(reverse == true) {
                 node = node.right;
             } else {
                 node = node.left;
             }
        }
    }
}

class Solution {
    public boolean findTarget(TreeNode root, int k) {
        if(root == null) return false;
        // next
        BSTIterator l = new BSTIterator(root, false);
        // for before
        BSTIterator r = new BSTIterator(root, true);

        int i = l.next();
        int j = r.next();
        while(i<j) {
            if(i + j == k) return true;
            else if(i + j < k) i = l.next();
            else j = r.next();
        }
        return false;
    }
```