https://course.acciojob.com/idle?question=8b9059a2-281c-4f68-b038-6e7cd889f558

- EASY
- Max Score: 30 Points

Two sum in BST

You are provided with `root` of a valid Binary Search Tree and a target number `k`.

Your task is to return `true` if there exist two elements in the BST such that the sum is equal to the given target, Otherwise return `false`.

A Binary Search Tree is a binary tree where for every node, any descendant of `Node.left` has a value strictly less than `Node.val`, and any descendant of `Node.right` has a value strictly greater than `Node.val`.

NOTE: You have to complete the function `checkTarget`. The input and output is already handled.

Input Format

First line contains two space separated integers `n` and `k` denoting the nodes in the BST and target sum respectively.

Second line contains `n` space-seprated integers denoting InOrder traversal of the tree

Output Format

Print `true` if elements exist Otherwise print `false`

Example 1

Input

```
6 9
2 3 4 5 6 7
```

Output

```
true
```

Explanation

```
The sum can be obtained by 3+6=9 or 4+5=9 or 7+2=9
```

Example 2

Input

```
6 28
2 3 4 5 6 7
```

Output

```
false
```

Explanation

```
The sum cannot be obtained by any combination
```

Constraints

1<=n<=10^4

-10^4<=node.val<=10^4

-10^5<=k<=10^5

Topic Tags

- BST
- BFS
- DFS

- Trees
- 2-Pointers
- Hashing

My code

```
// in java
import java.util.*;

class Node {
    int val;
    Node left, right;
    public Node(int item){
        val = item;
        left = right = null;
    }
}

class BinarySearchTree
{
    Node constructBST(int[]arr,int start,int end,Node root)
    {
        if(start>end)return null;
        int mid=(start+end)/2;
        if(root==null)root=new Node(arr[mid]);
        root.left=constructBST(arr,start,mid-1, root.left);
        root.right=constructBST(arr,mid+1,end, root.right);
        return root;
    }
}
```

```
class Solution{
        static HashMap<Integer,Integer>hm=new HashMap<>();

        static boolean check(Node root, int k)
        {
                if(root==null)
                        return false;
          if(hm.containsKey(root.val))
                  return true;
                else
          {
                  hm.put(k-root.val,1);
                  return check(root.left,k) ||check(root.right,k);

          }
        }
    public static boolean checkTarget(Node root, int k){
        //write code here
                return check(root,k);
    }
}

public class Main {
    public static void main(String[] args) throws Throwable {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int k = sc.nextInt();
        int arr[]=new int[n];
        for (int i = 0; i < n; i++){
            arr[i] = sc.nextInt();
        }
```

```
        Node root=null;
        BinarySearchTree bst=new BinarySearchTree();
        root=bst.constructBST(arr,0,n-1,root);
        Solution A = new Solution();
        boolean ans = A.checkTarget(root,k);
        if(ans==true){
            System.out.println("true");
        }else{
            System.out.println("false");
        }
    }
}
```