https://course.acciojob.com/idle?question=3b992182-31dd-4aaa-a4fd-914a6a9669ff

- MEDIUM
- Max Score: 40 Points

Validate BST

Given a binary tree with `N` number of nodes, check if that input tree is BST (Binary Search Tree) or not. If yes, print `true`, print `false` otherwise. A binary search tree (BST) is a binary tree data structure which has the following properties.

• The left subtree of a node contains only nodes with data less than the node's data.

• The right subtree of a node contains only nodes with data greater than the node's data.

• Both the left and right subtrees must also be binary search trees.

Input Format

The first line contains an Integer 't', which denotes the number of test cases or queries to be run. Then the test cases follow.

The first line of input contains the elements of the tree in the level order form separated by a single space.

If any node does not have a left or right child, take -1 in its place.

Output Format

For each test case, print `true` if the binary tree is a BST, else print `false`.

Output for every test case will be denoted in a separate line.

Example 1

Input

```
1
3 1 5 -1 2 -1 -1 -1 -1
```

Output

```
true
```

Explanation

Level 1: For node 3 all the nodes in the left subtree (1,2) are less than 3 and all the nodes in the right subtree (5) are greater than 3.

Level 2: For node 1: The left subtree is empty and all the nodes in the right subtree (2) are greater than 1.

For node 5: Both right and left subtrees are empty.

Level 3: For node 2, both right and left subtrees are empty. Because all the nodes follow the property of a binary search tree, the function should return true.

Example 2

Input

```
1
3 2 5 1 4 -1 -1 -1 -1 -1 -1
```

Output

```
false
```

Explanation

For the root node, all the nodes in the right subtree (5) are greater than 3. But node with data 4 in the left subtree of node 3 is greater than 3, this does not satisfy the condition for the binary search tree. Hence, the function should return false.

Constraints

1 <= T <= 100

1 <= N <= 1000

-10^6 <= data <= 10^6

where N is the number of nodes in the tree, T represents the number of test cases, and data denotes data contained in the node of the binary tree.

Topic Tags

- **Trees**

My code

```
// n java
import java.util.LinkedList;
import java.util.Queue;
import java.io.*;
import java.util.*;

class Node{
    int data;
    Node left;
    Node right;
    Node(int data){
        this.data = data;
        left=null;
        right=null;
    }
}

class Main {
    static Node treeBuilder(String str){
        if(str.length()==0 || str.charAt(0)=='-'){
            return null;
        }
        String ip[] = str.split(" ");
        Node root = new Node(Integer.parseInt(ip[0]));
        Queue<Node> queue = new LinkedList<>();
```

```
    queue.add(root);
    int i = 1;
    while(queue.size()>0 && i < ip.length) {
        Node currNode = queue.peek();
        queue.remove();
        String currVal = ip[i];
        if(!currVal.equals("-1")) {
            currNode.left = new Node(Integer.parseInt(currVal));
            queue.add(currNode.left);
        }
        i++;
        if(i >= ip.length)
            break;
        currVal = ip[i];
        if(!currVal.equals("-1")) {
            currNode.right = new Node(Integer.parseInt(currVal));
            queue.add(currNode.right);
        }
        i++;
    }

    return root;
}
static void printInorder(Node root){
    if(root == null)
        return;
    printInorder(root.left);
    System.out.print(root.data+" ");
    printInorder(root.right);
}
```

```
  public static void main (String[] args) throws IOException{
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        int t=Integer.parseInt(br.readLine());
        while(t > 0){
           String s = br.readLine();
          Node root = treeBuilder(s);
           Solution g = new Solution();
           if(g.isBST(root))
           System.out.println("true");
        else
           System.out.println("false");
             t--;
      }
   }

}

class Solution{
       static int key=-100000;
      static  boolean flag=true;
      static void fun(Node root)
       {
             if(root!=null)
             {
                    fun(root.left);
                if(key>root.data)
                      flag=false;
                else  key=root.data;
```

```
                    fun(root.right);
                }

            }
        boolean isBST(Node node){
          // Your Code Here
              fun(node);
              return flag;

        }
}
```