**Binary Search Tree (BST) validation**

class Node{

    constructor(data){

        this.data = data;

        this.right = null;

        this.left = null;

    }

}

class BST{

    constructor(){

        this.root = null;

    }

    insert(val) {

        const newNode = new Node(val);

        if (!this.root) {

            this.root = newNode;

            return;

        }

        let current = this.root;

        while (current) {

            if (val < current.data) {

                if (!current.left) {

                    current.left = newNode;

                    return;

                }

                current = current.left;

            } else if (val > current.data) {

                if (!current.right) {

                    current.right = newNode;

                    return;

                }

                current = current.right;

            } else {

                return;

            }

        }

    }

    printInOrder(node = this.root){

        if(!node) return []

       return[...this.printInOrder(node.left),node.data,...this.printInOrder(node.right)]

    }

    insertAbnormal(val){

        let current = this.root;

    if (!current) {

        this.root = new Node(val);

        return;

    }

        while(current.left !== null){

            current= current.left

        }

        current.left = new Node(val)

    }

    isBstCheckAscendingMethod(node = this.root){

        let arr = this.printInOrder(node)

        for(let i =0; i<arr.length-1; i++){

            if(arr[i]>= arr[i+1]) {

                return false

            }

        }

        return true

    }

    isBST(node = this.root, min = null, max = null){

        if (!node) {

            return true;

        }

        if((min !== null && node.data <= min) || (max !== null && node.data >= max)){

            return false

        }

        return this.isBST(node.left, min, node.data) && this.isBST(node.right, node.data, max)

    }

}

const tree = new BST();

tree.insert(5);

tree.insert(3);

tree.insert(7);

tree.insert(2);

tree.insert(4);

tree.insert(6);

tree.insert(8);

console.log(tree.printInOrder())

console.log(tree.isBstCheckAscendingMethod())

console.log(tree.isBST())

tree.insertAbnormal(50)

console.log(tree.printInOrder())

console.log(tree.isBstCheckAscendingMethod())

console.log(tree.isBST())