Name : Sudheep S(22CS171) Binary Search Tree Date:25-11-2024

1.Binary Search Tree

package Test;

import java.util.TreeSet;

public class Test {

public static void main(String[] args) {

TreeSet<Integer> b=new TreeSet();

b.add(10);

b.add(30);

b.add(20);

b.add(40);

b.add(50);

b.add(60);

System.***out***.println(b);

int s=10;

if (b.contains(s)) {

System.***out***.println("found");

}

b.remove(50);

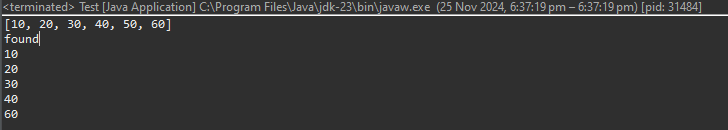
for(Integer v:b) {

System.***out***.println(v);

}

}

}



2.validate whether the given binary tree is bst

class TreeNode {

int val;

TreeNode left;

TreeNode right;

TreeNode(int val) {

this.val = val;

left = null;

right = null;

}

}

public class Main {

private boolean validate(TreeNode node, Integer lower, Integer upper) {

if (node == null) return true;

int val = node.val;

if (lower != null && val <= lower) return false;

if (upper != null && val >= upper) return false;

return validate(node.left, lower, val) && validate(node.right, val, upper);

}

public boolean isValidBST(TreeNode root) {

return validate(root, null, null);

}

public static void main(String[] args) {

TreeNode root = new TreeNode(2);

root.left = new TreeNode(1);

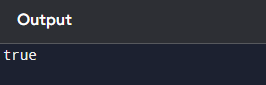
root.right = new TreeNode(3);

Main validator = new Main();

System.out.println(validator.isValidBST(root)); // Output: true

}

}



3.Convert the given tree into bst

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

class TreeNode {

int val;

TreeNode left;

TreeNode right;

TreeNode(int val) {

this.val = val;

left = null;

right = null;

}

}

public class Main {

private void collectValues(TreeNode node, List<Integer> values) {

if (node == null) return;

collectValues(node.left, values);

values.add(node.val);

collectValues(node.right, values);

}

private void convertToBST(TreeNode node, List<Integer> values, int[] index) {

if (node == null) return;

convertToBST(node.left, values, index);

node.val = values.get(index[0]++);

convertToBST(node.right, values, index);

}

public void convertToBST(TreeNode root) {

List<Integer> values = new ArrayList<>();

collectValues(root, values);

Collections.sort(values);

convertToBST(root, values, new int[]{0});

}

private void inOrder(TreeNode node) {

if (node == null) return;

inOrder(node.left);

System.out.print(node.val + " ");

inOrder(node.right);

}

public static void main(String[] args) {

TreeNode root = new TreeNode(10);

root.left = new TreeNode(2);

root.right = new TreeNode(7);

root.left.left = new TreeNode(8);

root.left.right = new TreeNode(4);

Main obj = new Main();

System.out.print("Original Tree (In-order): ");

obj.inOrder(root);

System.out.println();

obj.convertToBST(root);

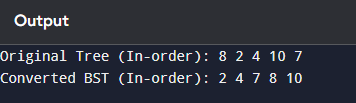
System.out.print("Converted BST (In-order): ");

obj.inOrder(root);

System.out.println();

}

}



4.Top,Bottom,left,right

import java.util.\*;

class TreeNode {

int val;

TreeNode left, right;

TreeNode(int val) {

this.val = val;

left = right = null;

}

}

public class Main {

private void topView(TreeNode root) {

if (root == null) return;

Map<Integer, Integer> map = new TreeMap<>();

Queue<Pair> queue = new LinkedList<>();

queue.add(new Pair(root, 0));

while (!queue.isEmpty()) {

Pair p = queue.poll();

if (!map.containsKey(p.hd)) {

map.put(p.hd, p.node.val);

}

if (p.node.left != null) queue.add(new Pair(p.node.left, p.hd - 1));

if (p.node.right != null) queue.add(new Pair(p.node.right, p.hd + 1));

}

System.out.println("Top View: " + map.values());

}

private void bottomView(TreeNode root) {

if (root == null) return;

Map<Integer, Integer> map = new TreeMap<>();

Queue<Pair> queue = new LinkedList<>();

queue.add(new Pair(root, 0));

while (!queue.isEmpty()) {

Pair p = queue.poll();

map.put(p.hd, p.node.val);

if (p.node.left != null) queue.add(new Pair(p.node.left, p.hd - 1));

if (p.node.right != null) queue.add(new Pair(p.node.right, p.hd + 1));

}

System.out.println("Bottom View: " + map.values());

}

private void leftView(TreeNode root) {

if (root == null) return;

Queue<TreeNode> queue = new LinkedList<>();

queue.add(root);

System.out.print("Left View: ");

while (!queue.isEmpty()) {

int size = queue.size();

System.out.print(queue.peek().val + " ");

for (int i = 0; i < size; i++) {

TreeNode node = queue.poll();

if (node.left != null) queue.add(node.left);

if (node.right != null) queue.add(node.right);

}

}

System.out.println();

}

private void rightView(TreeNode root) {

if (root == null) return;

Queue<TreeNode> queue = new LinkedList<>();

queue.add(root);

System.out.print("Right View: ");

while (!queue.isEmpty()) {

int size = queue.size();

TreeNode last = null;

for (int i = 0; i < size; i++) {

TreeNode node = queue.poll();

if (node.left != null) queue.add(node.left);

if (node.right != null) queue.add(node.right);

last = node;

}

System.out.print(last.val + " ");

}

System.out.println();

}

public static void main(String[] args) {

TreeNode root = new TreeNode(20);

root.left = new TreeNode(10);

root.right = new TreeNode(30);

root.left.left = new TreeNode(5);

root.left.right = new TreeNode(15);

root.right.left = new TreeNode(25);

root.right.right = new TreeNode(35);

Main obj = new Main();

obj.topView(root);

obj.bottomView(root);

obj.leftView(root);

obj.rightView(root);

}

}

class Pair {

TreeNode node;

int hd;

Pair(TreeNode node, int hd) {

this.node = node;

this.hd = hd;

}

}

