S RAMYA – AIML - 2DSA – 25/11/24

LEFT VIEW BST

import java.util.\*;

public class LeftViewBST {

    public static void dfs(TreeNode root,int level, List<Integer> ds){

        if (root == null) return;

        if (ds.size() == level) ds.add(root.val);

        dfs(root.left, level+1, ds);

        dfs(root.right, level+1, ds);

    }

    public static void main(String[] args){

        TreeNode root = ExampleBST.getTree();

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

        dfs(root, 0, leftView);

        System.out.println(leftView);

    }

}

class TreeNode {

    public int val;

    public TreeNode left, right;

    public TreeNode (int val){

        this.val = val;

        this.left = null;

        this.right = null;

    }

}

class ExampleBST {

    public static TreeNode getTree(){

        TreeNode one = new TreeNode(1);

        TreeNode two = new TreeNode(2);

        TreeNode three = new TreeNode(3);

        TreeNode four = new TreeNode(4);

        TreeNode seven = new TreeNode(7);

        two.left = one;

        two.right = three;

        four.left = two;

        four.right = seven; // note four is the root

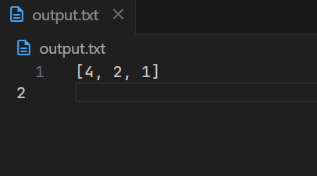
        return four;

    }

}

TC: O(N)

SC: O(H)



RIGHT VIEW BST

import java.util.\*;

public class RightViewBST {

    public static void dfs(TreeNode root,int level, List<Integer> ds){

        if (root == null) return;

        if (ds.size() == level) ds.add(root.val);

        dfs(root.right, level+1, ds);

        dfs(root.left, level+1, ds);

    }

    public static void main(String[] args){

        TreeNode root = ExampleBST.getTree();

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

        dfs(root, 0, rightView);

        System.out.println(rightView);

    }

}

class TreeNode {

    public int val;

    public TreeNode left, right;

    public TreeNode (int val){

        this.val = val;

        this.left = null;

        this.right = null;

    }

}

class ExampleBST {

    public static TreeNode getTree(){

        TreeNode one = new TreeNode(1);

        TreeNode two = new TreeNode(2);

        TreeNode three = new TreeNode(3);

        TreeNode four = new TreeNode(4);

        TreeNode seven = new TreeNode(7);

        two.left = one;

        two.right = three;

        four.left = two;

        four.right = seven; // note four is the root

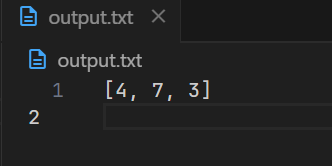
        return four;

    }

}

TC: O(N)

SC: O(H)



TOP VIEW BST

import java.util.\*;

public class TopViewBST {

    public static void topView(TreeNode root, List<Integer> tView){

        Queue<Tuple> q = new LinkedList<>();

        q.add(new Tuple(0, root));

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

        while (!q.isEmpty()){

            Tuple pol = q.poll();

            if (!mp.containsKey(pol.v)){

                mp.put(pol.v, pol.node.val);

            }

            if (pol.node.left != null) q.add(new Tuple(pol.v-1, pol.node.left));

            if (pol.node.right != null) q.add(new Tuple(pol.v+1, pol.node.right));

        }

        for (Map.Entry<Integer, Integer> item: mp.entrySet()){

            tView.add(item.getValue());

        }

    }

   public static void main(String[] args) {

    TreeNode root = ExampleBST.getTree();

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

    topView(root, tView);

    System.out.println(tView);

   }

}

class Tuple {

    int v;

    TreeNode node;

    Tuple(int v, TreeNode node){

        this.v = v;

        this.node = node;

    }

}

class TreeNode {

    public int val;

    public TreeNode left, right;

    public TreeNode (int val){

        this.val = val;

        this.left = null;

        this.right = null;

    }

}

class ExampleBST {

    public static TreeNode getTree(){

        TreeNode one = new TreeNode(1);

        TreeNode two = new TreeNode(2);

        TreeNode three = new TreeNode(3);

        TreeNode four = new TreeNode(4);

        TreeNode seven = new TreeNode(7);

        two.left = one;

        two.right = three;

        four.left = two;

        four.right = seven; // note four is the root

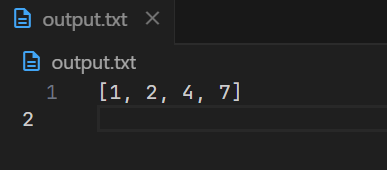
        return four;

    }

}

TC: O(N)

SC: O(2N)



BOTTOM VIEW BST

import java.util.\*;

public class BottomViewBST {

    public static void bottomView(TreeNode root, List<Integer> bView) {

        Queue<Tuple> q = new LinkedList<>();

        q.add(new Tuple(0, root));

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

        while (!q.isEmpty()) {

            Tuple pol = q.poll();

            mp.put(pol.v, pol.node.val);

            if (pol.node.left != null)

                q.add(new Tuple(pol.v - 1, pol.node.left));

            if (pol.node.right != null)

                q.add(new Tuple(pol.v + 1, pol.node.right));

        }

        for (Map.Entry<Integer, Integer> item : mp.entrySet()) {

            bView.add(item.getValue());

        }

    }

    public static void main(String[] args) {

        TreeNode root = ExampleBST.getTree();

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

        bottomView(root, bView);

        System.out.println(bView);

    }

}

class Tuple {

    int v;

    TreeNode node;

    Tuple(int v, TreeNode node) {

        this.v = v;

        this.node = node;

    }

}

class TreeNode {

    public int val;

    public TreeNode left, right;

    public TreeNode(int val) {

        this.val = val;

        this.left = null;

        this.right = null;

    }

}

class ExampleBST {

    public static TreeNode getTree() {

        TreeNode one = new TreeNode(1);

        TreeNode two = new TreeNode(2);

        TreeNode three = new TreeNode(3);

        TreeNode four = new TreeNode(4);

        TreeNode seven = new TreeNode(7);

        two.left = one;

        two.right = three;

        four.left = two;

        four.right = seven; // note four is the root

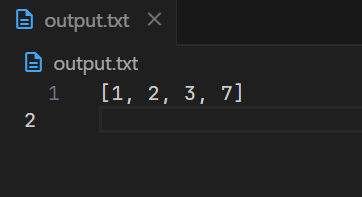
        return four;

    }

}

TC: O(N)

SC: O(2N)



VALIDATE BST

public class ValidateBST {

    public static boolean checkBST(TreeNode root, int minn, int maxx){

        if (root == null) return true;

        if (root.val <=minn || root.val >= maxx) return false;

        if (checkBST(root.left, minn, root.val) && checkBST(root.right, root.val, maxx)) return true;

        return false;

    }

    public static void main(String[] args) {

        TreeNode root = ExampleBST.getTree();

        if (checkBST(root, Integer.MIN\_VALUE, Integer.MAX\_VALUE)) System.out.println("valid");

        else System.out.println("invalid");

    }

}

class TreeNode {

    public int val;

    public TreeNode left, right;

    public TreeNode(int val) {

        this.val = val;

        this.left = null;

        this.right = null;

    }

}

class ExampleBST {

    public static TreeNode getTree() {

        TreeNode one = new TreeNode(1);

        TreeNode two = new TreeNode(2);

        TreeNode three = new TreeNode(3);

        TreeNode four = new TreeNode(4);

        TreeNode seven = new TreeNode(7);

        two.left = one;

        two.right = three;

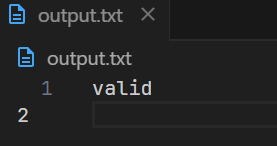
        four.left = two;

        four.right = seven; // note four is the root

        return four;

    }

}



TC: O(N)

SC: O(H)

CONVERT BT TO BST

import java.util.\*;

public class ConvertBTBST {

    public static void inorder(TreeNode root, List<Integer> ds){

        if (root == null) return;

        inorder(root.left, ds);

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

        ds.add(root.val);

        inorder(root.right, ds);

    }

    public static void convert(TreeNode root, List<Integer> ds, int[] i){

        if (root == null) return;

        convert(root.left, ds, i);

        root.val = ds.get(i[0]);

        i[0]++;

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

        convert(root.right, ds, i);

    }

    public static void main(String[] args) {

        TreeNode root = ExampleBST.getTree();

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

        System.out.println("Inorder original: ");

        inorder(root, ds);

        System.out.println(" ");

        Collections.sort(ds);

        System.out.println("Inorder after converted: ");

        int[] i = {0};

        convert(root, ds, i);

    }

}

class TreeNode {

    public int val;

    public TreeNode left, right;

    public TreeNode(int val) {

        this.val = val;

        this.left = null;

        this.right = null;

    }

}

class ExampleBST {

    public static TreeNode getTree() {

        TreeNode one = new TreeNode(1);

        TreeNode two = new TreeNode(2);

        TreeNode three = new TreeNode(3);

        TreeNode four = new TreeNode(4);

        TreeNode seven = new TreeNode(7);

        two.left = one;

        two.right = three;

        four.left = two;

        four.right = seven; // note four is the root

        return Cfour;

    }

}

TC: O(N) + O(NLOGN) + O(N)

SC: O(N) + O(H) + O(H)

