Assignment 1

1. Balanced Tree

class Solution {

public boolean isBalanced(TreeNode root) {

if (root == null) return true;

int left = getHeight(root.left);

int right = getHeight(root.right);

if (Math.abs(left - right) > 1) return false;

return isBalanced(root.left) && isBalanced(root.right);

}

private int getHeight(TreeNode root) {

if (root == null) return 0;

return Math.max(getHeight(root.left), getHeight(root.right)) + 1;

}

}

1. Find Minimum depth of all the leaves in a tree.

class Solution {

public int minDepth(TreeNode root) {

if (root == null) return 0;

int left = getHeight(root.left) + 1;

int right = getHeight(root.right) + 1;

return Math.min(left, right) == 1 ? Math.max(left, right) : Math.min(left, right);

}

private int getHeight(TreeNode root) {

if (root == null) return 0;

int left = getHeight(root.left) + 1;

int right = getHeight(root.right) + 1;

if (left == 1) return right;

if (right == 1) return left;

return Math.min(left, right);

}

}

1. Print all the leaves of a tree

class Solution {

public List<List<Integer>> findLeaves(TreeNode root) {

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

if (root == null) return lists;

getHeight(root, lists);

return lists;

}

private int getHeight(TreeNode root, List<List<Integer>> lists) {

if (root == null) return -1;

int left = getHeight(root.left, lists);

int right = getHeight(root.right, lists);

int height = Math.max(left, right) + 1;

if (height == lists.size()) {

lists.add(new ArrayList<Integer>());

}

lists.get(height).add(root.val);

return height;

}

}

1. Print sum of all the left leaves

class Solution {

int sum = 0;

public int sumOfLeftLeaves(TreeNode root) {

if (root == null) return sum;

int left = sumOfLeftLeaves(root.left);

int right = sumOfLeftLeaves(root.right);

if (root.left != null && root.left.left == null && root.left.right == null) sum += root.left.val;

return sum;

}

}