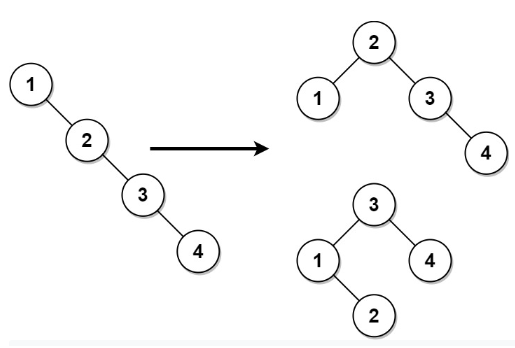
<https://leetcode.com/problems/balance-a-binary-search-tree/>

Given the root of a binary search tree, return *a* ***balanced*** *binary search tree with the same node values*. If there is more than one answer, return **any of them**.

A binary search tree is **balanced** if the depth of the two subtrees of every node never differs by more than 1.

**Example 1:**

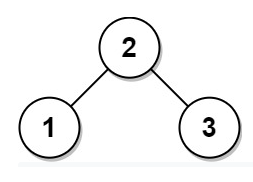


Input: root = [1,null,2,null,3,null,4,null,null]

Output: [2,1,3,null,null,null,4]

Explanation: This is not the only correct answer, [3,1,4,null,2] is also correct.

**Example 2:**



Input: root = [2,1,3]

Output: [2,1,3]

**Constraints:**

* The number of nodes in the tree is in the range [1, 104].
* 1 <= Node.val <= 105

**Attempt 1: 2022-11-09**

**Solution 1:  Recursive traversal and build BST based on sorted array (10min)**

/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode() {}

\* TreeNode(int val) { this.val = val; }

\* TreeNode(int val, TreeNode left, TreeNode right) {

\* this.val = val;

\* this.left = left;

\* this.right = right;

\* }

\* }

\*/

class Solution {

List<TreeNode> list = new ArrayList<TreeNode>();

public TreeNode balanceBST(TreeNode root) {

// Convert the tree to a sorted array using an in-order traversal

toInorderArray(root);

// Construct a new balanced tree from the sorted array recursively

return arrayToBST(0, list.size() - 1);

}

private void toInorderArray(TreeNode root) {

if(root == null) {

return;

}

toInorderArray(root.left);

list.add(root);

toInorderArray(root.right);

}

private TreeNode arrayToBST(int start, int end) {

if(start > end) {

return null;

}

int mid = start + (end - start) / 2;

TreeNode root = list.get(mid);

root.left = arrayToBST(start, mid - 1);

root.right = arrayToBST(mid + 1, end);

return root;

}

}

**Refer to**

<https://leetcode.com/problems/balance-a-binary-search-tree/discuss/539686/JavaC%2B%2B-Sorted-Array-to-BST-O(N)-Clean-code>

**Intuitive**

* Traverse binary tree in-order to get sorted array
* The problem become [108. Convert Sorted Array to Binary Search Tree](https://leetcode.com/problems/convert-sorted-array-to-binary-search-tree/)

class Solution {

List<TreeNode> sortedArr = new ArrayList<>();

public TreeNode balanceBST(TreeNode root) {

inorderTraverse(root);

return sortedArrayToBST(0, sortedArr.size() - 1);

}

void inorderTraverse(TreeNode root) {

if (root == null) return;

inorderTraverse(root.left);

sortedArr.add(root);

inorderTraverse(root.right);

}

TreeNode sortedArrayToBST(int start, int end) {

if (start > end) return null;

int mid = (start + end) / 2;

TreeNode root = sortedArr.get(mid);

root.left = sortedArrayToBST(start, mid - 1);

root.right = sortedArrayToBST(mid + 1, end);

return root;

}

}