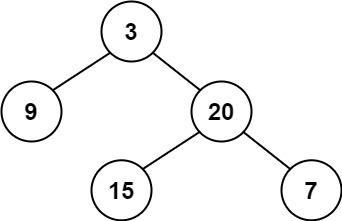
Given a binary tree, determine if it is height-balanced.

For this problem, a height-balanced binary tree is defined as:

a binary tree in which the left and right subtrees of *every* node differ in height by no more than 1.

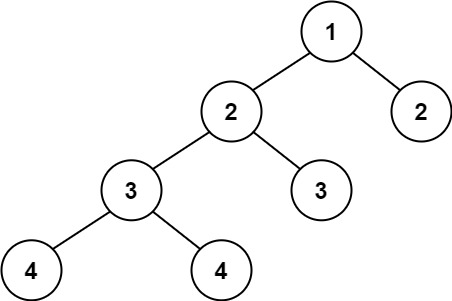
**Example 1:**



**Input:** root = [3,9,20,null,null,15,7]

**Output:** true

**Example 2:**



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

**Output:** false

**Example 3:**

**Input:** root = []

**Output:** true

**Solution:**

/\*\*

\* 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 {

public boolean isBalanced(TreeNode root) {

return dfsHeight (root) != -1;

}

int dfsHeight (TreeNode root) {

if (root == null) return 0;

int leftHeight = dfsHeight (root.left);

if (leftHeight == -1) return -1;

int rightHeight = dfsHeight (root.right);

if (rightHeight == -1) return -1;

if (Math.abs(leftHeight - rightHeight) > 1) return -1;

return Math.max(leftHeight, rightHeight) + 1;

}

}

T.C -> O(N) [For recursion]

S.C-> O(N) [For a skew tree]