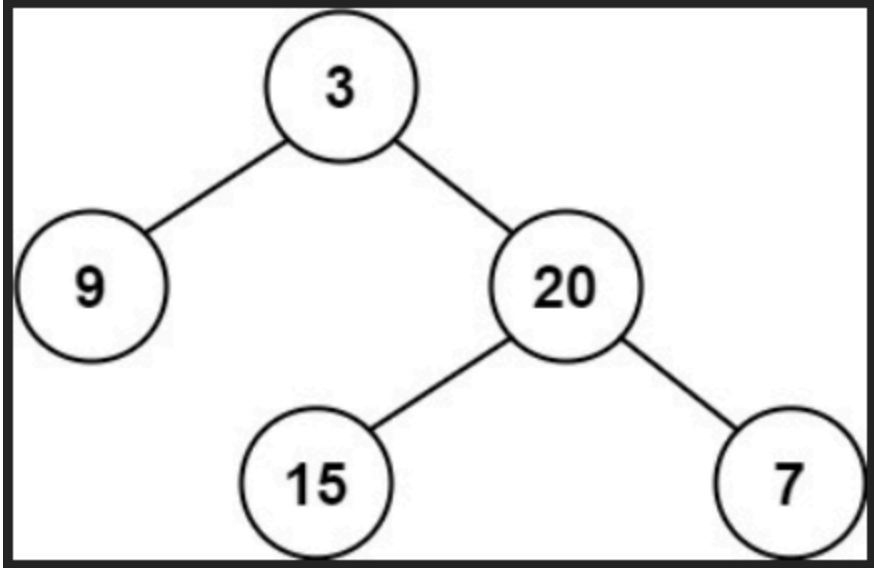
110. Balanced Binary Tree

Easy | Topics | Companies

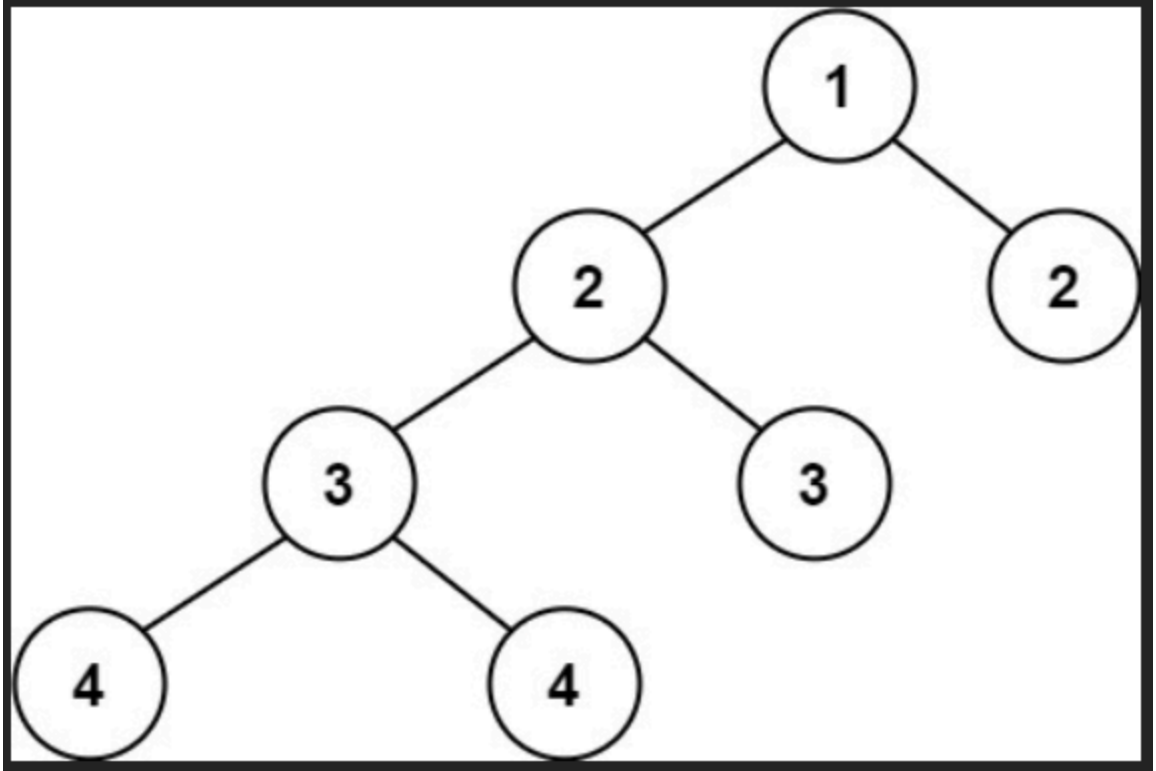
Given a binary tree, determine if it is **height-balanced**.

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
```

Constraints:

- The number of nodes in the tree is in the range `[0, 5000]`.
- `-10^4 <= Node.val <= 10^4`

Python:

```
# Definition for a binary tree node.
```

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class Solution:
    def isBalanced(self, root: TreeNode) -> bool:
        def check(node):
            if not node:
                return 0  # height of empty tree = 0

            left = check(node.left)
            if left == -1:  # left subtree not balanced
                return -1

            right = check(node.right)
            if right == -1:  # right subtree not balanced
                return -1

            if abs(left - right) > 1:  # not balanced
                return -1

            return max(left, right) + 1  # return height

        return check(root) != -1
```

JavaScript:

```
// Definition for a binary tree node.
function TreeNode(val, left = null, right = null) {
    this.val = val;
    this.left = left;
    this.right = right;
}

var isBalanced = function(root) {
    function checkHeight(node) {
        if (node === null) return 0; // Base case: height of empty tree is 0

        let leftHeight = checkHeight(node.left);
        if (leftHeight === -1) return -1; // Left subtree is unbalanced

        let rightHeight = checkHeight(node.right);
        if (rightHeight === -1) return -1; // Right subtree is unbalanced
```

```
        if (Math.abs(leftHeight - rightHeight) > 1) return -1; // Current node unbalanced

        return Math.max(leftHeight, rightHeight) + 1; // Return height of current node
    }

    return checkHeight(root) !== -1;
};
```

Java:

```
/**
 * 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 checkHeight(root) != -1;
    }

    private int checkHeight(TreeNode node) {
        if (node == null) return 0;
        int left = checkHeight(node.left);
        if (left == -1) return -1;
        int right = checkHeight(node.right);
        if (right == -1) return -1;
        if (Math.abs(left - right) > 1) return -1;
        return Math.max(left, right) + 1;
    }
}
```