Problem 110: Balanced Binary Tree

Problem Information

Difficulty: Easy
Acceptance Rate: 56.53%
Paid Only: No
Tags: Tree, Depth-First Search, Binary Tree

Problem Description

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

Example 1:

![](https://assets.leetcode.com/uploads/2020/10/06/balance_1.jpg)

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

Example 2:

![](https://assets.leetcode.com/uploads/2020/10/06/balance_2.jpg)

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]`. * `-104 <= Node.val <= 104`

Code Snippets

C++:

```
/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode() : val(0), left(nullptr), right(nullptr) {}
* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
* TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),
right(right) {}
* };
*/
class Solution {
public:
bool isBalanced(TreeNode* root) {

}
};
```

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) {

}
}
```

Python3:

```
# 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: Optional[TreeNode]) -> bool:
```