Documentation: Determine if a Binary Tree is Height-Balanced

Problem Statement

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

Definition

A binary tree is considered height-balanced if:

- 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

Explanation:

```
    3
   / \
  9  20
    /  \
   15   7
```

The tree is balanced because the height difference between the left and right subtrees of every node is no more than 1.

Example 2

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

Output: false

Explanation:

The tree is not balanced because the height difference between the left and right subtrees of node 3 is more than 1.

Example 3

Input: root = []

Output: true

Explanation: An empty tree is balanced by definition.

Constraints

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

Approach

To determine if the binary tree is height-balanced, we will use a recursive approach to check the height of each subtree. During the height calculation, we will also check if the subtree is balanced. If any subtree is found to be unbalanced, we return -1 to indicate that the tree is not balanced.

Detailed Steps

1. Define a helper function check_height that computes the height of a subtree.
2. For each node, recursively compute the height of the left and right subtrees.
3. If the height difference between the left and right subtrees is more than 1, return -1 to indicate the tree is not balanced.
4. If either the left or right subtree is unbalanced (i.e., check_height returns -1), propagate this value upwards.
5. If the subtree is balanced, return its height.
6. The tree is balanced if the root node's height calculation does not return -1.

Explanation of the Code

- The function isBalanced takes the root of the binary tree as input.
- *The helper function check_height:*
  - Returns 0 if the node is None, indicating the height of an empty subtree.
  - Recursively calculates the heights of the left and right subtrees.
  - Checks if the subtree is balanced by comparing the heights of the left and right subtrees.
  - Returns -1 if the subtree is unbalanced.
  - Otherwise, returns the height of the subtree.
- The isBalanced function returns True if the tree is balanced (check_height does not return -1), otherwise returns False.

This approach ensures that the tree is checked for balance in a single traversal, making the solution efficient with a time complexity of O(n), where n is the number of nodes in the tree.