

# 110. Balanced Binary Tree

## 110 Balanced Binary Tree

- Depth-first Search + Tree

### Description

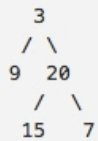
Given a binary tree, determine if it is height-balanced.

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

a binary tree in which the depth of the two subtrees of every node never differ by more than 1.

#### Example 1:

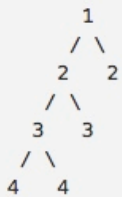
Given the following tree `[3,9,20,null,null,15,7]` :



Return true.

#### Example 2:

Given the following tree `[1,2,2,3,3,null,null,4,4]` :



Return false.

### 1. Thought line

- Height-balanced BST

### 2. Depth-first Search + Tree

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 * }
```

```

*   TreeNode *right;
*   TreeNode(int x) : val(x), left(NULL), right(NULL) {}
* };
*/
class Solution {
private:
    int findHeight(TreeNode* node){
        if (!node) return 0;
        return 1+max(findHeight(node->left), findHeight(node->right));
    }
public:
    bool isBalanced(TreeNode* root) {
        if (root==nullptr) return true;
        int leftHight = findHeight(root->left);
        int rightHight = findHeight(root->right);
        bool current = (abs(leftHight - rightHight)<=1)?true:false;
        return current && isBalanced(root->left) && isBalanced(root->right);
    }
};

```