

110. Balanced Binary Tree

110 Balanced Binary Tree

- Depth-first Search + Tree

Description

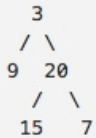
Given a binary tree, determine if it is height-balanced.

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:

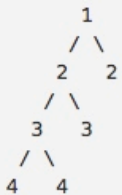
Given the following tree `[3,9,20,null,null,15,7]` :



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

```
1 /**
2  * Definition for a binary tree node.
3  * struct TreeNode {
4  *     int val;
5  *     TreeNode *left;
6  *     TreeNode *right;
7  *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
8  * };
```

```

9  */
10 class Solution {
11 private:
12     int findHeight(TreeNode* node){
13         if (!node) return 0;
14         return 1+max(findHeight(node->left), findHeight(node->right));
15     }
16 public:
17     bool isBalanced(TreeNode* root) {
18         if (root==nullptr) return true;
19         int leftHight = findHeight(root->left);
20         int rightHight = findHeight(root->right);
21         bool current = (abs(leftHight - rightHight)<=1)?true:false;
22         return current && isBalanced(root->left) && isBalanced(root->right);
23     }
24 };

```