剑指55-II 平衡二叉树

简单题。直接递归,二叉树的后序遍历，谁都能想到。关键是怎么剪枝提升效率。

```cpp
// 我的没有剪枝
class Solution {
public:
    bool isB;
    int search(TreeNode* root) {
        if(root == NULL) {
            return 0;
        }
        int l = search(root->left);
        int r = search(root->right);
        if(abs(l - r) > 1) isB = false;
        return max(l, r) + 1;
    }

    bool isBalanced(TreeNode* root) {
        isB = true;
        search(root);
        return isB;
    }
};

// 剪枝了，不需要额外的bool变量，深度直接返回-1表示二叉树不平衡了。最佳方案。
class Solution {
public:
    int search(TreeNode* root) {
        if(root == NULL) {
            return 0;
        }
        int l = search(root->left);
        if(l == -1) return -1;
        int r = search(root->right);
        if(r == -1) return -1;
        if(abs(l - r) > 1) return -1;
        return max(l, r) + 1;
    }

    bool isBalanced(TreeNode* root) {
        return search(root) == -1 ? false : true;
    }
};
```

```java
// 还有一种先序遍历的思想，思想值得学习讨论。
// 但是做了重复计算，在这到题目里很不适合。
class Solution {
    public boolean isBalanced(TreeNode root) {
```

```
        if (root == null) return true;
        return Math.abs(depth(root.left) - depth(root.right)) <= 1 && isBalanced(root.left)
&& isBalanced(root.right);
    }

    private int depth(TreeNode root) {
        if (root == null) return 0;
        return Math.max(depth(root.left), depth(root.right)) + 1;
    }
}

```