

## 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。

```
1 // 我的没有剪枝
2 class Solution {
3 public:
4     bool isB;
5     int search(TreeNode* root) {
6         if(root == NULL) {
7             return 0;
8         }
9         int l = search(root->left);
10        int r = search(root->right);
11        if(abs(l - r) > 1) isB = false;
12        return max(l, r) + 1;
13    }
14
15    bool isBalanced(TreeNode* root) {
16        isB = true;
17        search(root);
18        return isB;
19    }
20 };
21
22 // 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。
23 class Solution {
24 public:
25     int search(TreeNode* root) {
26         if(root == NULL) {
27             return 0;
28         }
29         int l = search(root->left);
30         if(l == -1) return -1;
31         int r = search(root->right);
32         if(r == -1) return -1;
33         if(abs(l - r) > 1) return -1;
34         return max(l, r) + 1;
35     }
36
37     bool isBalanced(TreeNode* root) {
38         return search(root) == -1 ? false : true;
39     }
40 };
```

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

```
5         if (root == null) return true;
6         return Math.abs(depth(root.left) - depth(root.right)) <= 1 && isBalanced(root.left)
&& isBalanced(root.right);
7     }
8
9     private int depth(TreeNode root) {
10         if (root == null) return 0;
11         return Math.max(depth(root.left), depth(root.right)) + 1;
12     }
13 }
14
```