# 题目

输入一棵二叉树的根节点，求该树的深度。从根节点到叶节点依次经过的节点（含根、叶节点）形成树的一条路径，最长路径的长度为树的深度。

**示例：**

给定二叉树 [3,9,20,null,null,15,7]，

3

/ \

9 20

/ \

15 7

返回它的最大深度 3 。

**类似题目：Leetcode 104**

# 分析

## 方法一：递归

**思路：**

直观的方法是通过递归来解决问题。利用DFS（深度优先遍历）。

**代码：**

/\*\*

\* Definition for a binary tree node.

\* struct TreeNode {

\* int val;

\* TreeNode \*left;

\* TreeNode \*right;

\* TreeNode(int x) : val(x), left(NULL), right(NULL) {}

\* };

\*/

class Solution {

public:

int dfs(TreeNode\* root, int max\_dept) {

if(!root) return max\_dept;

//到叶子节点的时候返回，此时max\_dept是递增后的

int left\_dept = dfs(root->left, max\_dept+1);

int right\_dept = dfs(root->right, max\_dept+1);

//不能用max\_dept++/++max\_depth

return max(left\_dept, right\_dept);

}

int maxDepth(TreeNode\* root) {

int max\_dept = 0;

return dfs(root, max\_dept);

}

};

另外一种写法：

**复杂度：**

时间复杂度：我们每个结点只访问一次，因此时间复杂度为O(N)，其中N是结点的数量。

空间复杂度：在最糟糕的情况下，树是完全不平衡的，例如每个结点只剩下左子结点，递归将会被调用N次（树的高度），因此保持调用栈的存储将是O(N)。但在最好的情况下（树是完全平衡的），树的高度将是log(N)。因此，在这种情况下的空间复杂度将是 O(log(N))。

## 方法二：迭代

**思路：**

**代码：**

**复杂度：**