

0104. 二叉树的最大深度

👤 [ITCharge](#) ⌚ 大约 2 分钟

- 标签：树、深度优先搜索、广度优先搜索、二叉树
- 难度：简单

题目链接

- [0104. 二叉树的最大深度 - 力扣](#)

题目大意

描述： 给定一个二叉树的根节点 `root` 。

要求： 找出该二叉树的最大深度。

说明：

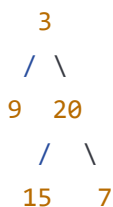
- **二叉树的深度：** 根节点到最远叶子节点的最长路径上的节点数。
- **叶子节点：** 没有子节点的节点。

示例：

- 示例 1:

输入: [3,9,20,null,null,15,7]

对应二叉树



输出: 3

解释: 该二叉树的最大深度为 3

py

解题思路

思路 1：递归算法

根据递归三步走策略，写出对应的递归代码。

1. 写出递推公式：当前二叉树的最大深度 = $\max(\text{当前二叉树左子树的最大深度}, \text{当前二叉树右子树的最大深度}) + 1$ 。
 - 即：先得到左右子树的高度，在计算当前节点的高度。
2. 明确终止条件：当前二叉树为空。
3. 翻译为递归代码：
 1. 定义递归函数：`maxDepth(self, root)` 表示输入参数为二叉树的根节点 `root`，返回结果为该二叉树的最大深度。
 2. 书写递归主体：`return max(self.maxDepth(root.left) + self.maxDepth(root.right))`。
 3. 明确递归终止条件：`if not root: return 0`

思路 1：代码

```
class Solution:
    def maxDepth(self, root: Optional[TreeNode]) -> int:
        if not root:
            return 0

        return max(self.maxDepth(root.left), self.maxDepth(root.right)) + 1
```

py

思路 1：复杂度分析

- **时间复杂度：** $O(n)$ ，其中 n 是二叉树的节点数目。
- **空间复杂度：** $O(n)$ 。递归函数需要用到栈空间，栈空间取决于递归深度，最坏情况下递归深度为 n ，所以空间复杂度为 $O(n)$ 。