0111. 二叉树的最小深度

ITCharge 大约 1 分钟

- 标签：树、深度优先搜索、广度优先搜索
- 难度：简单

题目链接

- 0111. 二叉树的最小深度 - 力扣

题目大意

描述：给定一个二叉树的根节点 $root$。

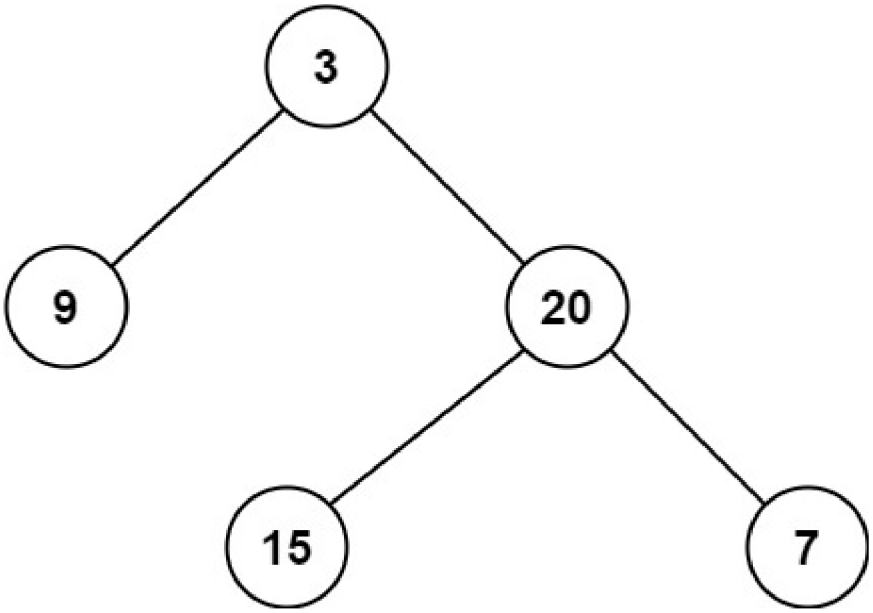
要求：找出该二叉树的最小深度。

说明：

- **最小深度**：从根节点到最近叶子节点最短路径上的节点数量。
- **叶子节点**：指没有子节点的节点。
- 树中节点数的范围在 $[0, 10^5]$ 内。
- $-1000 \leq Node.val \leq 1000$。

示例：

- 示例 1：




```py
输入：root = [3,9,20,null,null,15,7]
输出：2
```

- 示例 2：

```py
输入：root = [2,null,3,null,4,null,5,null,6]
输出：5
```

解题思路

思路 1：深度优先搜索

深度优先搜索递归遍历左右子树，记录最小深度。

对于每一个非叶子节点，计算其左右子树的最小叶子节点深度，将较小的深度+1 即为当前节点的最小叶子节点深度。

思路 1：代码

```py
class Solution:
    def minDepth(self, root: TreeNode) -> int:
        # 遍历到空节点，直接返回 0
        if root == None:
            return 0

        # 左右子树为空，说明为叶子节点 返回 1
        if root.left == None and root.right == None:
            return 1

        leftHeight = self.minDepth(root.left)
        rightHeight = self.minDepth(root.right)

        # 当前节点的左右子树的最小叶子节点深度
        min_depth = 0xffffff
        if root.left:
            min_depth = min(leftHeight, min_depth)
        if root.right:
            min_depth = min(rightHeight, min_depth)

        # 当前节点的最小叶子节点深度
        return min_depth + 1
```

思路 1：复杂度分析

- **时间复杂度**：$O(n)$，其中 n 是树中的节点数量。
- **空间复杂度**：$O(n)$。