

0116. 填充每个节点的下一个右侧节点指针

👤 [ITCharge](#) ⌚ 大约 2 分钟

- 标签：树、深度优先搜索、广度优先搜索、链表、二叉树
- 难度：中等

题目链接

- [0116. 填充每个节点的下一个右侧节点指针 - 力扣](#)

题目大意

描述： 给定一个完美二叉树，所有叶子节点都在同一层，每个父节点都有两个子节点。完美二叉树结构如下：

```
struct Node {  
    int val;  
    Node *left;  
    Node *right;  
    Node *next;  
}
```

py

要求： 填充每个 `next` 指针，使得这个指针指向下一个右侧节点。如果找不到下一个右侧节点，则将 `next` 置为 `None`。

说明：

- 初始状态下，所有 `next` 指针都被设置为 `None`。
- 树中节点的数量在 $[0, 2^{12} - 1]$ 范围内。
- $-1000 \leq node.val \leq 1000$ 。
- 进阶：
 - 只能使用常量级额外空间。
 - 使用递归解题也符合要求，本题中递归程序占用的栈空间不算做额外的空间复杂度。

示例：

- 示例 1:

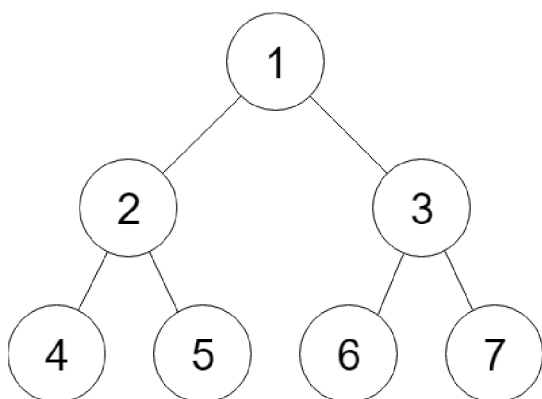


Figure A

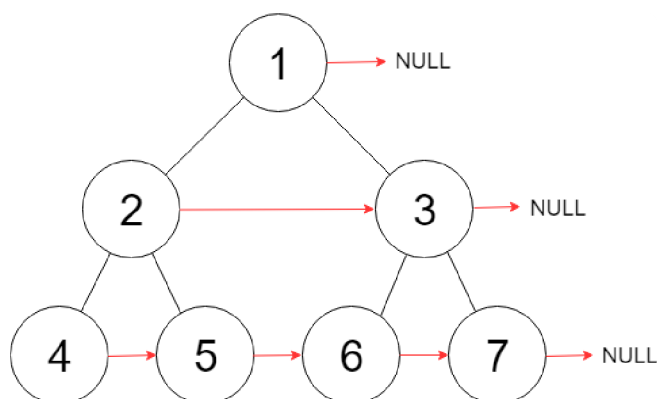


Figure B

输入: `root = [1,2,3,4,5,6,7]`

输出: `[1,#,2,3,#,4,5,6,7,#]`

解释: 给定二叉树如图 A 所示, 你的函数应该填充它的每个 `next` 指针, 以指向其下一个右侧节点, 如图 B 所示。序列化的输出按层序遍历排列, 同一层节点由 `next` 指针连接, `'#'` 标志着每一层的结束。

py

- 示例 2:

输入: `root = []`

输出: `[]`

py

解题思路

思路 1: 层次遍历

在层次遍历的过程中, 依次取出每一层的节点, 并进行连接。然后再扩展下一层节点。

思路 1: 代码

```
import collections

class Solution:
    def connect(self, root: 'Node') -> 'Node':
        if not root:
```

py

```
        return root
    queue = collections.deque()
    queue.append(root)
    while queue:
        size = len(queue)
        for i in range(size):
            node = queue.popleft()
            if i < size - 1:
                node.next = queue[0]

            if node.left:
                queue.append(node.left)
            if node.right:
                queue.append(node.right)
    return root
```

思路 1：复杂度分析

- **时间复杂度：** $O(n)$ ，其中 n 为树中的节点数量。
- **空间复杂度：** $O(1)$ 。