# 题目

给定一个二叉树：

struct Node {

int val;

Node \*left;

Node \*right;

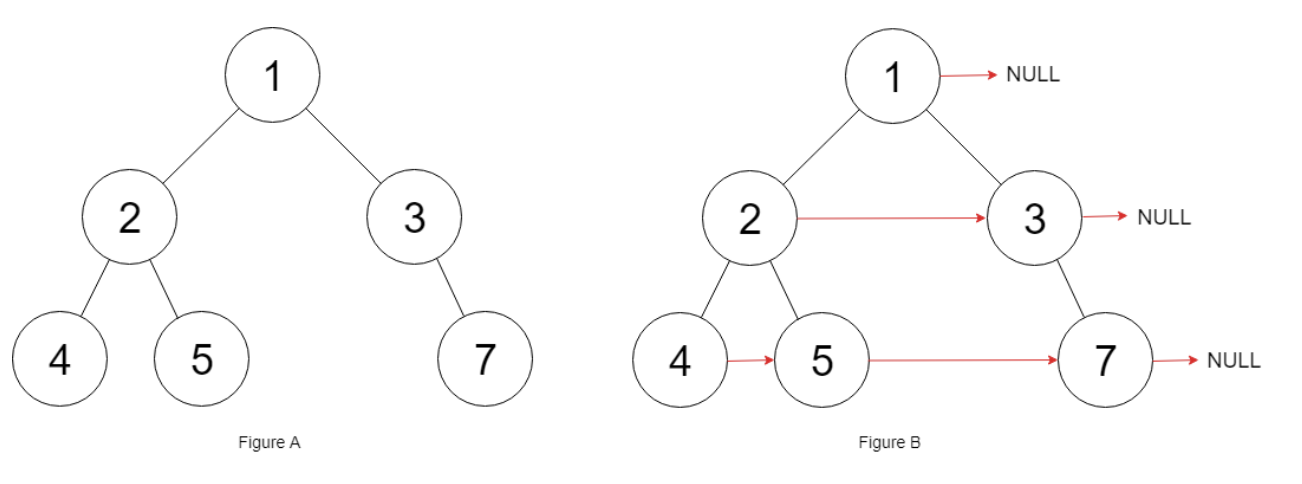
Node \*next;

}

填充它的每个 next 指针，让这个指针指向其下一个右侧节点。如果找不到下一个右侧节点，则将 next 指针设置为 NULL 。

初始状态下，所有 next 指针都被设置为 NULL 。

示例 1：



输入：root = [1,2,3,4,5,null,7]

输出：[1,#,2,3,#,4,5,7,#]

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。序列化输出按层序遍历顺序（由 next 指针连接），'#' 表示每层的末尾。

示例 2：

输入：root = []

输出：[]

提示：

树中的节点数在范围 [0, 6000] 内

-100 <= Node.val <= 100

进阶：

你只能使用常量级额外空间。

使用递归解题也符合要求，本题中递归程序的隐式栈空间不计入额外空间复杂度。

# 分析

/\*

// Definition for a Node.

class Node {

public:

int val;

Node\* left;

Node\* right;

Node\* next;

Node() : val(0), left(NULL), right(NULL), next(NULL) {}

Node(int \_val) : val(\_val), left(NULL), right(NULL), next(NULL) {}

Node(int \_val, Node\* \_left, Node\* \_right, Node\* \_next)

: val(\_val), left(\_left), right(\_right), next(\_next) {}

};

\*/

class Solution {

public:

Node\* connect(Node\* root) {

if (!root) {

return nullptr;

}

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

int size = q.size();

Node\* prev = nullptr;

for (int i = 0; i < size; i++) {

Node\* current = q.front();

q.pop();

if (prev) {

prev->next = current;

}

if (current->left) {

q.push(current->left);

}

if (current->right) {

q.push(current->right);

}

prev = current;

}

}

return root;

}

};