# 题目

给定一棵二叉树，设计一个算法，创建含有某一深度上所有节点的链表（比如，若一棵树的深度为 D，则会创建出 D 个链表）。返回一个包含所有深度的链表的数组。

示例：

输入：[1,2,3,4,5,null,7,8]

1

/ \

2 3

/ \ \

4 5 7

/

8

输出：[[1],[2,3],[4,5,7],[8]]

# 分析

## 方法一：广度优先搜索

思路：

可以使用广度优先搜索（BFS）来遍历二叉树，并在遍历过程中将每层的节点连接成链表。以下是实现此功能的代码示例：

struct TreeNode {

int val;

TreeNode\* left;

TreeNode\* right;

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

};

struct ListNode {

int val;

ListNode\* next;

ListNode(int x) : val(x), next(NULL) {}

};

class Solution {

public:

vector<ListNode\*> listOfDepth(TreeNode\* tree) {

vector<ListNode\*> result;

if (!tree) return result;

queue<TreeNode\*> q;

q.push(tree);

while (!q.empty()) {

int size = q.size();

ListNode\* head = nullptr;

ListNode\* prev = nullptr;

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

TreeNode\* node = q.front();

q.pop();

ListNode\* current = new ListNode(node->val);

if (!head) {

head = current;

} else {

prev->next = current;

}

prev = current;

if (node->left) q.push(node->left);

if (node->right) q.push(node->right);

}

result.push\_back(head);

}

return result;

}

};

这段代码首先创建一个队列q，并将根节点tree推入队列。然后，开始一个循环，每次迭代都表示处理当前层的节点。在循环中，首先计算当前层的节点数size，然后依次处理这些节点：

1、弹出队列中的节点node。

2、创建一个新的链表节点current，将node的值赋给current。

3、如果当前层的第一个节点，将current赋给head，否则将current接到prev节点的后面。

4、更新prev为current。

5、将node的左右子节点（如果存在）推入队列。

完成当前层的节点处理后，将当前层的链表头节点 `head` 推入结果数组 `result` 中。最终返回结果数组即可。