# 题目

从上到下按层打印二叉树，同一层的节点按从左到右的顺序打印，每一层打印到一行。

**例如:**

给定二叉树: [3,9,20,null,null,15,7],

3

/ \

9 20

/ \

15 7

返回其层次遍历结果：

[

[3],

[9,20],

[15,7]

]

**提示：**

节点总数 <= 1000

**注意：**本题与主站 102 题相同：

https://leetcode-cn.com/problems/binary-tree-level-order-traversal/

# 分析

## 方法一：递归法

**思路：**深度优先遍历

**代码：**

## 方法二：迭代法

**思路：**宽度优先

**代码：**

class Solution {

public:

vector<vector<int>> levelOrder(TreeNode\* root) {

vector <vector <int>> ret;

if (!root) return ret;

queue <TreeNode\*> q;

q.push(root);

while (!q.empty()) {

int currentLevelSize = q.size();

ret.push\_back(vector <int> ());

for (int i = 1; i <= currentLevelSize; ++i) {

auto node = q.front(); q.pop();

ret.back().push\_back(node->val);

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

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

}

}

return ret;

}

};

或：

class Solution {

public:

vector<vector<int>> levelOrder(TreeNode\* root) {

queue<TreeNode\*> que;

que.push(root);

vector<vector<int>> res;

while (que.size() != 0) { //或者使用empty()

int size = que.size(); //当前层节点个数

vector<int> level;

while (size --) {

TreeNode\* cur = que.front();

que.pop();

if (!cur) { //必不可少，跳过null节点

continue;

}

level.push\_back(cur->val);

que.push(cur->left);

que.push(cur->right);

}

if (level.size() != 0) { //防止最后输出null

res.push\_back(level);

}

}

return res;

}

};

注：上述输出不要求输出null，则需要过滤这种情况（标红处）。