# 题目

给定一个二叉树，返回其节点值自底向上的层次遍历。 （即按从叶子节点所在层到根节点所在的层，逐层从左向右遍历）

**例如：**

给定二叉树 [3,9,20,null,null,15,7],

3

/ \

9 20

/ \

15 7

返回其自底向上的层次遍历为：

[

[15,7],

[9,20],

[3]

]

# 分析

## 方法一：递归法

**思路：**

**代码：**

class Solution {

public:

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

int n = getDep(root);

vector<vector<int>> ans(n, vector<int>());

dfs(root, 0, ans, n - 1);

return ans;

}

void dfs(TreeNode \*root, int depth, vector<vector<int>>& ans, int n) {

if (root == NULL) return ;

ans[n - depth].push\_back(root->val); // 倒着装 n - depth

dfs(root->left, depth + 1, ans, n);

dfs(root->right, depth + 1, ans, n);

}

int getDep(TreeNode \*root) { // 求树的高度

if (root == NULL) return 0;

return max(getDep(root->left), getDep(root->right)) + 1;

}

};

或：

class Solution {

public:

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

vector<vector<int>> res;

helper(res,root,0);

reverse(res.begin(),res.end());

return res;

}

void helper(vector<vector<int>>& res,TreeNode\* node,int level){

if(node==NULL) return ;

if(level>=res.size()){

vector<int> level\_res;

res.emplace\_back(level\_res);

}

res[level].emplace\_back(node->val);

if(node->left!=NULL) helper(res,node->left,level+1);

if(node->right!=NULL) helper(res,node->right,level+1);

}

};

## 方法二：迭代法

**思路：**

**代码：**

class Solution

{

public:

vector<vector<int>> levelOrderBottom(TreeNode\* root)

{

vector<vector<int>> res;

if(root==NULL)return res;

queue<TreeNode\*> myQueue;

TreeNode\* p;

myQueue.push(root);

while(!myQueue.empty())

{

vector<int> temp;

int queueSize=myQueue.size();

for(int i=0;i<queueSize;++i)

{

p=myQueue.front();

myQueue.pop(); //弹出上一层的节点，存储当前层的节点

temp.push\_back(p->val);

if(p->left!=NULL)

myQueue.push(p->left);

if(p->right!=NULL)

myQueue.push(p->right);

}

res.push\_back(temp);

}

reverse(res.begin(),res.end());

return res;

}

};

或：

class Solution {

public:

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

std::queue<TreeNode\*> Q;

Q.push(root);

std::vector<vector<int>> ret;

while(!Q.empty())

{

uint curLevelSize = Q.size();

vector<int> vec;

while(curLevelSize--)

{

TreeNode \*node = Q.front();

Q.pop();

if(!node)

continue;

vec.push\_back(node->val);

Q.push(node->left);

Q.push(node->right);

}

if(vec.size()!=0)

ret.push\_back(vec);

}

reverse(ret.begin(),ret.end());

return ret;

}

};