# 题目

给定一棵二叉树，想象自己站在它的右侧，按照从顶部到底部的顺序，返回从右侧所能看到的节点值。

**示例:**

输入: [1,2,3,null,5,null,4]

输出: [1, 3, 4]

解释:

1 <---

/ \

2 3 <---

\ \

5 4 <---

# 分析

/\*\*

\* Definition for a binary tree node.

\* struct TreeNode {

\* int val;

\* TreeNode \*left;

\* TreeNode \*right;

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

\* };

\*/

class Solution {

public:

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

unordered\_map<int, int> rightmostValueAtDepth;

int max\_depth = -1;

queue<TreeNode\*> nodeQueue;

queue<int> depthQueue;

nodeQueue.push(root);

depthQueue.push(0);

while (!nodeQueue.empty()) {

TreeNode\* node = nodeQueue.front();

nodeQueue.pop();

int depth = depthQueue.front();

depthQueue.pop();

if (node != NULL) {

// 维护二叉树的最大深度

max\_depth = max(max\_depth, depth);

// 由于每一层最后一个访问到的节点才是我们要的答案，因此不断更新对应深度的信息即可

rightmostValueAtDepth[depth] = node -> val;

nodeQueue.push(node -> left);

nodeQueue.push(node -> right);

depthQueue.push(depth + 1);

depthQueue.push(depth + 1);

}

}

vector<int> rightView;

for (int depth = 0; depth <= max\_depth; ++depth) {

rightView.push\_back(rightmostValueAtDepth[depth]);

}

return rightView;

}

};