1. Binary Tree Right Side View

Given a binary tree, imagine yourself standing on the *right* side of it, return the values of the nodes you can see ordered from top to bottom.

**Example:**

Input: [1,2,3,null,5,null,4]  
Output: [1, 3, 4]  
Explanation:  
  
 1 <---  
 / \  
2 3 <---  
 \ \  
 5 4 <---

**解1** 层序遍历，每一层最后一个节点肯定是最右边的节点。在层序遍历时，用cur表示当前层的节点数，next表示下一层节点数，每次从队列里面出来一个，cur减1，当cur==0时，表明当前出队的节点已经是最右边的，然后令cur=next, next = 0；每进队一个节点，next增加1

/\*\*  
 \* Definition for a binary tree node.  
 \* struct TreeNode {  
 \* int val;  
 \* TreeNode \*left;  
 \* TreeNode \*right;  
 \* TreeNode() : val(0), left(nullptr), right(nullptr) {}  
 \* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}  
 \* TreeNode(int x, TreeNode \*left, TreeNode \*right) : val(x), left(left), right(right) {}  
 \* };  
 \*/  
class Solution {  
public:  
 vector<int> rightSideView(TreeNode\* root) {  
 vector<int>ans;  
 if(root == NULL)return ans;  
   
 queue<TreeNode\*>q;  
 q.push(root);  
 int cur = 1, next = 0;  
 while(!q.empty()){  
 TreeNode\* tmp = q.front();  
 q.pop();  
 cur--;  
 if(tmp->left){  
 q.push(tmp->left);  
 next++;  
 }  
 if(tmp->right){  
 q.push(tmp->right);  
 next++;  
 }  
 if(cur == 0){  
 ans.push\_back(tmp->val);  
 cur = next;  
 next = 0;  
 }  
 }  
 return ans;  
 }  
};

**解2** dfs

class Solution {  
public:  
 vector<int> rightSideView(TreeNode\* root) {  
 vector<int>ans;  
 if(root == NULL)return ans;  
 dfs(root, 0, ans);  
 return ans;  
 }  
 void dfs(TreeNode\* node, int level, vector<int>& ans){  
 if(level == ans.size())ans.push\_back(node->val);  
 if(node->right)dfs(node->right, level+1, ans);  
 if(node->left)dfs(node->left, level+1, ans);  
 }  
};