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       <---
```

思路，从右往左看，那么一层只能看到一个最靠右的节点，那么从右子树开始遍历，通过当前层中，右边是否有节点已经被存储了来判读节点存储。因为是从上到下遍历，若是有节点存储后，那么层数必等于存储数组的长度，以这个为判断遍历即可。

代码

```
/**
 * 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:
    void dfs(TreeNode* root,vector<int> &result,int i){
        if(result.size()==i){
            result.push_back(root->val);
        }
        if(root->right!=NULL)
        {
            dfs(root->right,result,i+1);
        }
        if(root->left!=NULL){
            dfs(root->left,result,i+1);
        }
    }
    vector<int> rightSideView(TreeNode* root) {
        vector<int> result;
        if(root==NULL){
            return result;
```

```
        }
        dfs(root,result,0);
        return result;
    }
};
```