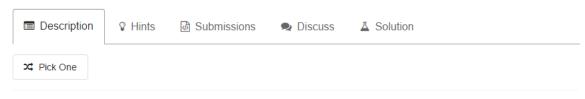
题意:给定一颗二叉树,返回从右边看这颗二叉树所看到的节点序列(从上到下) 思路:层次遍历法,遍历到每层最后一个节点时,把其放到结果集合中

199. 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:

```
public class L199 {
class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;
    public TreeNode(int x) {
        val = x;
public List<Integer> rightSideView(TreeNode root) {
    List<Integer> ans = new ArrayList<Integer>();
    if(root == null)
        return ans;
    LinkedList<TreeNode> queue = new LinkedList<L199.TreeNode>();
    queue.add(root);
    //这个是在每一层的节点加入到queue中以后,在后面加入一个null节点,这样,只要现在queue中的第一个节点是null,则说明当前这个节点是最右的那个节点
    queue.add(null);
    while (!queue.isEmpty()) {
        TreeNode node = queue.pollFirst();//pollFirst是检索并移除列表的第一个元素
        if(node == null){//如果当前的节点(queue的首及节点)时null,说明下一层的节点全部加入到queue中了
            if(queue.isEmpty()){//如果为空,代表下一层节点数为0,代表这时已经到了树的最后一层
            }else {
                queue.add(null);//如果不为空,需要在一层节点全部加入queue中后加入null节点
        }else {
            //peek方法是检索但不移除列表的第一个元素,当第一个节点是null时,则是最右的节点,从而加入到ans中
            if (queue.peek() == null) {
                ans.add(node.val);
            if(node.left != null) {
                queue.add(node.left);
            if(node.right != null) {
                queue.add(node.right);
        }
    return ans;
}
```