199. Binary Tree Right Side View

link

正常想的是bfs加最后一个, 但是dfs应该也可以

```
public List<Integer> rightSideView(TreeNode root) {
    List<Integer> res = new ArrayList();
    if(root == null) return res;
    Queue<TreeNode> queue = new LinkedList();
    queue.offer(root);
    while(!queue.isEmpty()){
        int size = queue.size();
        for(int i = 0; i < size; ++i){
            TreeNode node = queue.poll();
            if(i == size - 1)
                res.add(node.val);
            if(node.left != null)
                queue.offer(node.left);
            if(node.right != null)
                queue.offer(node.right);
        }
    }
    return res;
}
```

想了很久都没想出来怎么用level来判断, 一个用参数传递当前level, 一个用return传递深度, 这里可以用list.size判断是否需要加入list.

```
int maxLevel;
public List<Integer> rightSideView(TreeNode root) {
    List<Integer> res = new ArrayList();
    if(root != null){
        res.add(root.val);
        dfs(root, 1, res);
    }
    return res;
}
public void dfs(TreeNode node, int level, List res){
    if(node == null) return;
    if(level > maxLevel.size()){
        maxLevel = level;
        res.add(node.val);
    }
    maxLevel = Math.max(maxLevel, level);
    dfs(node.right, level + 1, res);
    dfs(node.left, level + 1, res);
    }
```

改了以后

```
    public List<Integer> rightSideView(TreeNode root) {
        List<Integer> res = new ArrayList();
        if(root != null){
            dfs(root, 1, res);
        }
        return res;
    }
    public void dfs(TreeNode node, int level, List res){
        if(node == null) return;
        if(level > res.size()){
            res.add(node.val);
        }
        dfs(node.right, level + 1, res);
        dfs(node.left, level + 1, res);

    }
```