剑指offer 22 链表中倒数第K个节点

llabel：链表、双指针
输入一个链表，输出该链表中倒数第k个节点。为了符合大多数人的习惯，本题从1开始计数，即链表的尾节点是倒数第1个节点。例如，一个链表有6个节点，从头节点开始，它们的值依次是1、2、3、4、5、6。这个链表的倒数第3个节点是值为4的节点。

> 给定一个链表: 1->2->3->4->5, 和 k = 2.
> 返回链表 4->5

- 双指针(一前一后间隔移动，当后面的指针移动到末端时，前面的指针就可以用来获取倒数第k个结点)

```
class Solution {
    public ListNode getKthFromEnd(ListNode head, int k) {
        ListNode node2 =  head; //在后面的
        ListNode node1 = null; // 在前面的
        ListNode currNode = head;
        for(int i =0 ;i< k; i++){
            currNode = currNode.next;
            node1 = currNode;
        }

        while (node1 != null){
            node1 = node1.next;
            node2 = node2.next;
        }
        return node2;
    }
}
```

- 用List把结点存起来

```
class Solution {
    public ListNode getKthFromEnd(ListNode head, int k) {

        if (head == null) return null;

        List<ListNode> list = new ArrayList<ListNode>();
        while (head != null) {
            list.add(head);
            head = head.next;
        }
        int totalNum = stack.size();

        // 出
        return list.get(totalNum - k);
    }
}
```

- 栈

```
class Solution {
    public ListNode getKthFromEnd(ListNode head, int k) {

        Stack<ListNode> stack = new Stack<>();
        //链表节点压栈
        while (head != null) {
            stack.push(head);
            head = head.next;
        }
        //在出栈串成新的链表
        ListNode currNode = stack.pop();
        while (--k > 0) {
            currNode = stack.pop();
        }
            return currNode;
    }
}
```

- 递归

```
class Solution {
   //全局变量，记录递归往回走的时候访问的结点数量
    int size;

    public ListNode getKthFromEnd(ListNode head, int k) {
        if (head == null) return head;
        ListNode node = getKthFromEnd(head.next, k);
        ++size;
        //从后面数结点数小于k，返回空
        if (size < k) {
            return null;
        } else if (size == k) {
            //从后面数访问结点等于k，直接返回传递的结点k即可
            return head;
        } else {
            //从后面数访问的结点大于k，说明我们已经找到了，
            //直接返回node即可
            return node;
        }
    }
}
```