题目 反转链表

考点 代码的鲁棒性 热点指数 79927 通过率 28.80%

具体题目

输入一个链表，反转链表后，输出新链表的表头。

容易出现的问题：

- 输入的链表头指针为NULL 或整个链表只有一个结点时，程序立即崩溃
- 反转后的链表出现断裂
- 返回的反转之后的头结点不是原始链表的尾结点。

Java　循环操作　详细思路

```
public class Solution {
    public ListNode ReverseList(ListNode head) {

        if(head==null)
            return null;
        //head为当前节点，如果当前节点为空的话，那就什么也不做，直接返回null；
        ListNode pre = null;
        ListNode next = null;
        //当前节点是head，pre为当前节点的前一节点，next为当前节点的下一节点
        //需要pre和next的目的是让当前节点从pre->head->next1->next2变成pre<-head next1->next2
        //即pre让节点可以反转所指方向，但反转之后如果不用next节点保存next1节点的话，此单链表就此断开了
        //所以需要用到pre和next两个节点
        //1->2->3->4->5
        //1<-2<-3 4->5
        while(head!=null){
            //做循环，如果当前节点不为空的话，始终执行此循环，此循环的目的就是让当前节点从指向next到指向pre
            //如此就可以做到反转链表的效果
            //先用next保存head的下一个节点的信息，保证单链表不会因为失去head节点的原next节点而就此断裂
            next = head.next;
            //保存完next，就可以让head从指向next变成指向pre了，代码如下
            head.next = pre;
            //head指向pre后，就继续依次反转下一个节点
            //让pre，head，next依次向后移动一个节点，继续下一次的指针反转
            pre = head;
            head = next;
        }
        //如果head为null的时候，pre就为最后一个节点了，但是链表已经反转完毕，pre就是反转后链表的第一个节点
        //直接输出pre就是我们想要得到的反转后的链表
        return pre;
    }
}
```

递归的方法其实是非常巧的，它利用递归走到链表的末端，然后再更新每一个node的next 值 ，实现链表的反转。而newhead 的值没有发生改变，为该链表的最后一个结点，所以，反转后，我们可以得到新链表的head。 注意关于链表问题的常见注意点的思考：
1、如果输入的头结点是 NULL，或者整个链表只有一个结点的时候
2、链表断裂的考虑

```
    public ListNode ReverseList(ListNode head) {
        ListNode pre = null;
        ListNode next = null;
        while (head != null) {
            next = head.next;
            head.next = pre;
            pre = head;
            head = next;
        }
        return pre;
    }
```

```
// 这里采用一种递归的方式，从链表节点的尾部进行反转指针即可。仔细体会，递归的简练。代码如下：
public class Solution {
    public ListNode ReverseList(ListNode head) {
           if(head == null || head.next == null) {
             return head;
         }
         ListNode preNode = ReverseList(head.next);
         head.next.next = head;
         head.next = null;
         return preNode;
    }
}
```

```
/**
 * 反转链表
 * 题目描述
 * 输入一个链表，反转链表后，输出链表的所有元素。
 *
 * @author shijiacheng
 * @date 2018/2/23
 */
public class ReverseListSolution {
    /**
     * 依次遍历所有节点，将所有节点的next指向前一个节点
     */
    public ListNode ReverseList(ListNode head) {
        ListNode pre = null;
        ListNode next = null;
        while (head != null) {
            next = head.next;//持有下一个节点的引用
            head.next = pre;//将当前节点对下一个节点的引用指向前一个节点
            pre = head;//将前一个节点指向当前节点
            head = next;//将当前节点指向下一个节点
        }
        return pre;
    }
}
```