链表逆序\_LeetCode\_206\_从尾到头打印链表\_剑指Offer\_6

# 反转链表ReverseList

## 题目介绍

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*链表T1：反转链表\_LeetCode\_206. Reverse Linked List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* 链表T1：反转链表\_LeetCode\_206. Reverse Linked List

\* 题目描述 链表T1：反转链表

\* 输入一个链表，反转链表后，输出新链表的表头。

\* Reverse a singly linked list.

\* Example:

\* Input: 1->2->3->4->5->NULL

\* Output: 5->4->3->2->1->NULL

\* Follow up:

\* A linked list can be reversed either iteratively or recursively.

\* Could you implement both?



## 思路分析

\* 思路分析：两种实现：1、循环实现；2、递归实现。

\* 1. 循环实现。需要三个引用，前一个节点preNode，当前节点currentNode，

\* 后一个节点nextNode；断开链表之前，暂时保存下一节点nextNode；

\* 在头节点head之前引入null，作为head的pre节点。

## Java代码

### While循环实现

/\*\*

\* iteratively : while循环实现

\*/

public ListNode reverseList(ListNode head) {

// if(head == null||head.next == null) return head;//下面的已经处理这种特殊情况

ListNode currentNode = head,preNode = null;

while(currentNode != null){

ListNode nextNode = currentNode.next;

currentNode.next = preNode;

preNode = currentNode;

currentNode = nextNode;

}

return preNode;

}



### 递归实现

/\*\*

\* recursively : 递归实现

\*/

public ListNode reverseList\_recursive(ListNode head) {

if(head == null||head.next == null) return head;

return reverseOne(null,head);

}

/\*\*

\* 该方法递归实现逆序整个链表

\* @param preTail : 已经逆序好的尾节点

\* @param head：剩下链表的头节点

\* @return ： 返回逆序后链表的头节点

\*/

public ListNode reverseOne(ListNode preTail,ListNode head){

if(head == null) return preTail;

ListNode next = head.next;

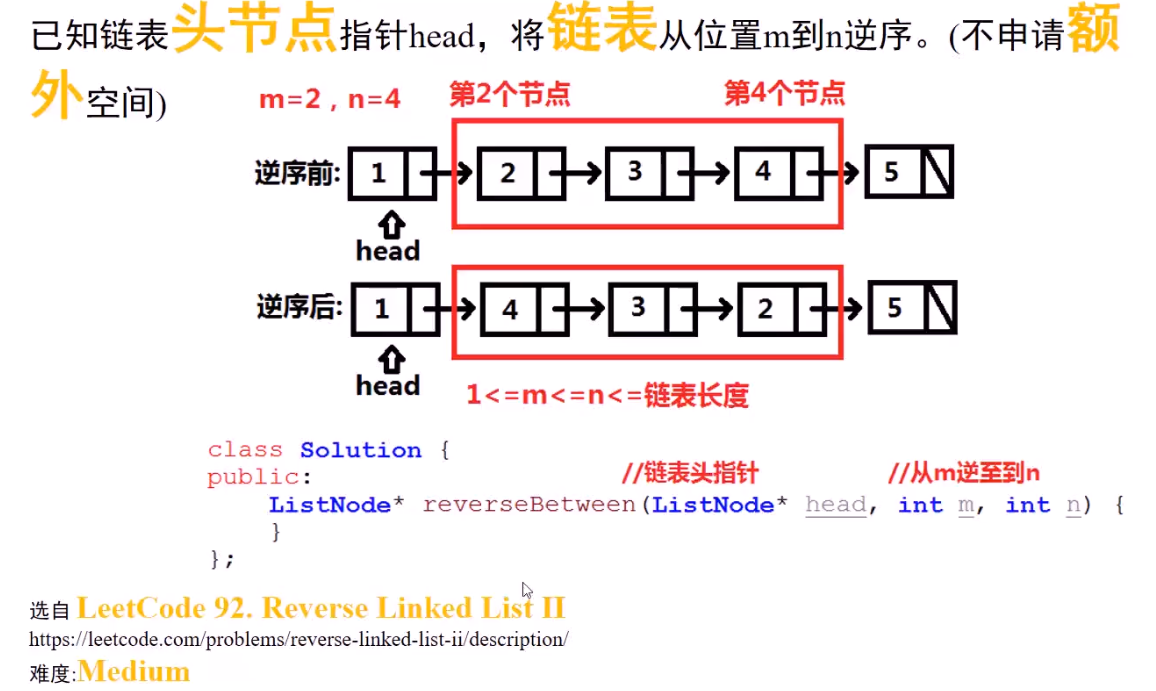
head.next = preTail;

preTail = head;

return reverseOne(preTail,next);

}

# 反转链表II-升级版



## 思路分析

