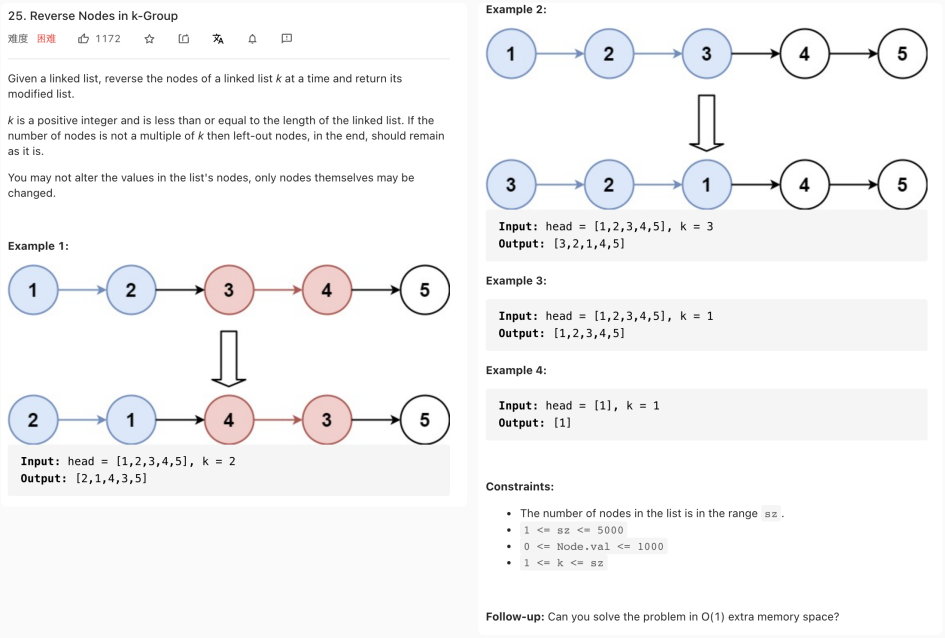
25. K 个一组翻转链表 ***

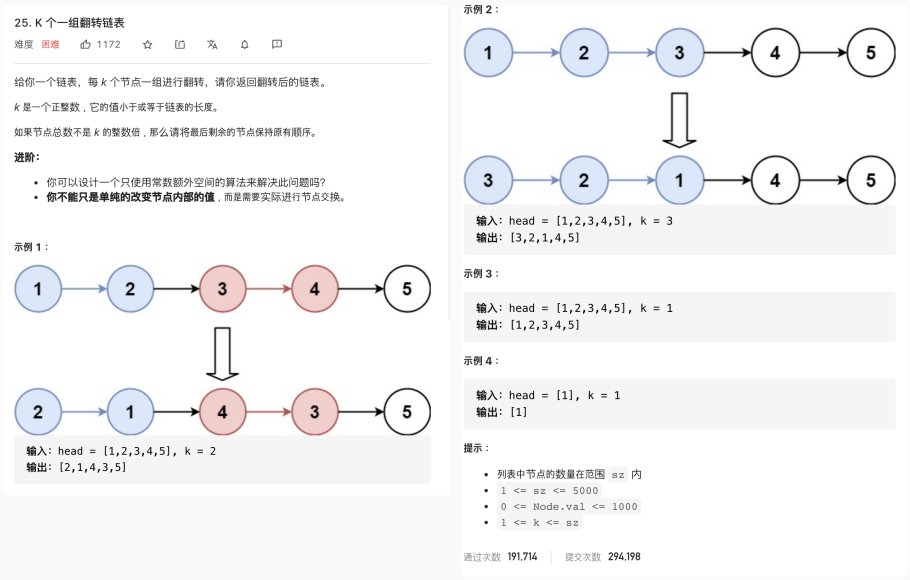
地址：🔗K 个一组翻转链表

题目：

- English：



- 中文：



思路 1：尾插法 （引自 房建斌学算法 - 图解 k 个一组翻转链表）

分析

⭐ 链表分区为已翻转部分 + 待反转部分 + 未反转部分；

⭐ 每次反转前，要确定翻转链表的范围，这个必须通过 k 次循环来确定；

⭐ 需要记录翻转链表前驱和后继，方便翻转完成后把已翻转部分和未翻转部分连接起来；

⭐ 初始需要两个变量 `pre` 和 `end`， `pre` 代表待翻转链表的前驱， `end` 代表待翻转链表的末尾；

⭐ 经过 k 次循环， `end` 到达末尾，记录待翻转链表的后继 `next = end.next` ；

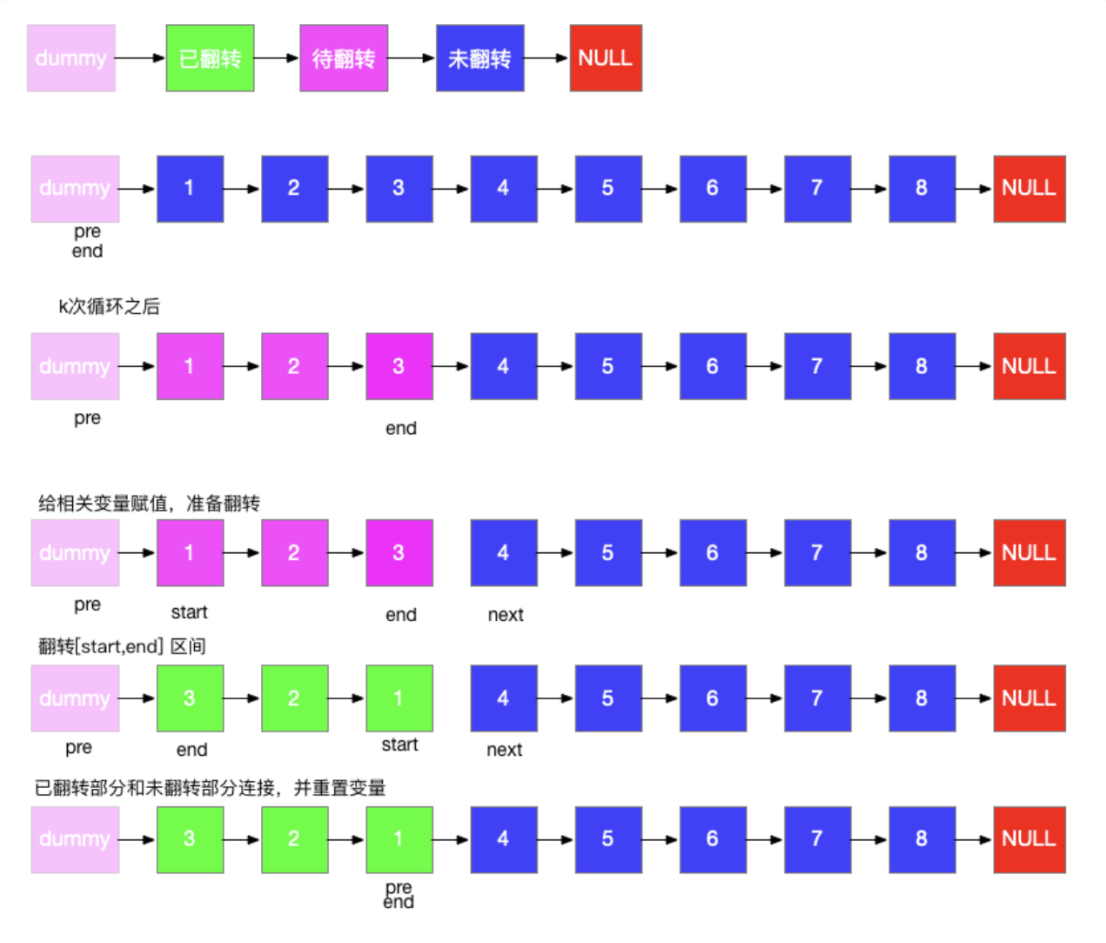
⭐ 翻转链表，然后将三部分链表连接起来，然后重置 `pre` 和 `end` 指针，然后进入下一次循环；

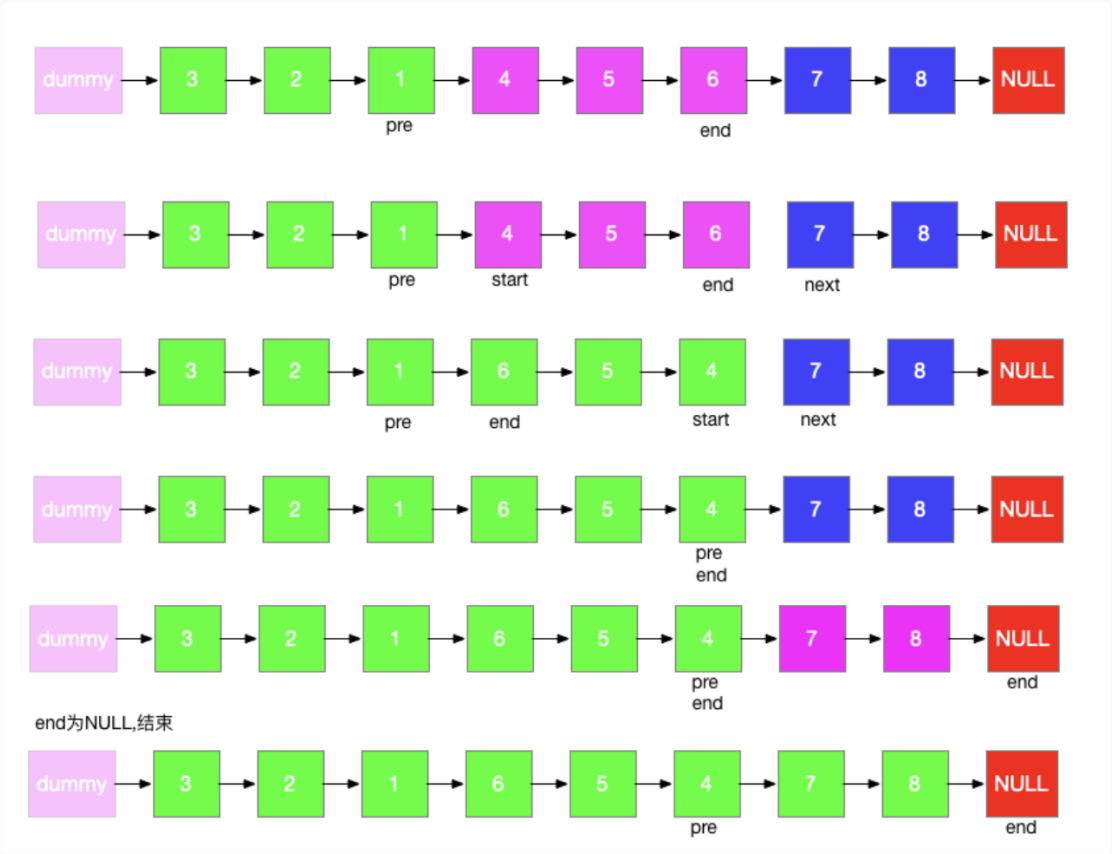
⭐ 特殊情况，当翻转部分长度不足 `k` 时，在定位 `end` 完成后， `end == null` ，已经到达末尾，说明题目已经完成，直接返回即可；

⭐ 时间复杂度：平均情况 `O( n * K )` ，最好情况为 `O(n)` , 最差情况 `O(n^2)` ；

⭐ 空间复杂度： `O（1）` ，除了几个必须的节点指针外，并没有占用其他空间。

图示






```java
// Java
// Time : 2021 - 07 - 14


public ListNode reverseKGroup(ListNode head, int k) {

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


  // 定义一个假的节点。
  ListNode dummy = new ListNode(0);


  // 假节点的next指向head。
  // dummy->1->2->3->4->5
  dummy.next = head;


  // 初始化pre和end都指向dummy
  // pre指每次要翻转的链表的头结点的上一个节点
  // end指每次要翻转的链表的尾节点
  ListNode pre = dummy;
  ListNode end = dummy;


  while (end.next != null) {


    // 循环k次，找到需要翻转的链表的结尾
    // 这里每次循环要判断end是否等于空,因为如果为空，end.next会报空指针异常。
```

```
    // dummy->1->2->3->4->5 若k为2，循环2次，end指向2
    for (int i = 0; i < k && end != null; i++) {
      end = end.next;
    }


    //如果end==null，即需要翻转的链表的节点数小于k，不执行翻转。
    if (end == null) {
      break;
    }


    //先记录下end.next,方便后面链接链表
    ListNode next = end.next;


    //然后断开链表
    end.next = null;


    //记录下要翻转链表的头节点
    ListNode start = pre.next;


    //翻转链表,pre.next指向翻转后的链表：1->2 变成2->1，  dummy->2->1
    pre.next = reverse(start);


    //翻转后头节点变到最后，通过.next把断开的链表重新链接。
    start.next = next;


    //将pre换成下次要翻转的链表的头结点的上一个节点。即start
    pre = start;


    //翻转结束，将end置为下次要翻转的链表的头结点的上一个节点。即start
    end = start;
  }
  return dummy.next;
}


//链表翻转
// 例子：    head：  1->2->3->4
public ListNode reverse(ListNode head) {


  //单链表为空或只有一个节点，直接返回原单链表
  if (head == null || head.next == null) {
    return head;
  }


  //前一个节点指针
  ListNode preNode = null;
```

```
89
90
91      //当前节点指针
92      ListNode curNode = head;
93
94
95      //下一个节点指针
96      ListNode nextNode = null;
97
98
99      while (curNode != null) {
100       nextNode = curNode.next; //nextNode 指向下一个节点,保存当前节点后面的链表。
101       curNode.next = preNode; //将当前节点next域指向前一个节点    null<-1<-2<-3<-4
102       preNode = curNode; //preNode 指针向后移动。preNode指向当前节点。
103       curNode = nextNode; //curNode指针向后移动。下一个节点变成当前节点
104     }
105     return preNode;
106 }
```