

25. K 个一组翻转链表 ***

地址：[🔗 K 个一组翻转链表](#)

题目：

- English :

25. Reverse Nodes in k-Group

难度 困难 1172

Given a linked list, reverse the nodes of a linked list k at a time and return its modified list.

k is a positive integer and is less than or equal to the length of the linked list. If the number of nodes is not a multiple of k then left-out nodes, in the end, should remain as it is.

You may not alter the values in the list's nodes, only nodes themselves may be changed.

Example 1:

Input: head = [1,2,3,4,5], k = 2
Output: [2,1,4,3,5]

Example 2:

Input: head = [1,2,3,4,5], k = 3
Output: [3,2,1,4,5]

Example 3:

Input: head = [1,2,3,4,5], k = 1
Output: [1,2,3,4,5]

Example 4:

Input: head = [1], k = 1
Output: [1]

Constraints:

- The number of nodes in the list is in the range `sz`.
- $1 \leq sz \leq 5000$
- $0 \leq \text{Node.val} \leq 1000$
- $1 \leq k \leq sz$

Follow-up: Can you solve the problem in $O(1)$ extra memory space?

- 中文：

25. K 个一组翻转链表

难度 困难 1172

给你一个链表，每 k 个节点一组进行翻转，请你返回翻转后的链表。

k 是一个正整数，它的值小于或等于链表的长度。

如果节点总数不是 k 的整数倍，那么请将最后剩余的节点保持原有顺序。

进阶：

- 你可以设计一个只使用常数额外空间的算法来解决此问题吗？
- 你不能只是单纯的改变节点内部的值，而是需要实际进行节点交换。

示例 1：

输入： head = [1,2,3,4,5], k = 2
输出： [2,1,4,3,5]

示例 2：

输入： head = [1,2,3,4,5], k = 3
输出： [3,2,1,4,5]

示例 3：

输入： head = [1,2,3,4,5], k = 1
输出： [1,2,3,4,5]

示例 4：

输入： head = [1], k = 1
输出： [1]

提示：

- 列表中节点的数量在范围 `sz` 内
- $1 \leq sz \leq 5000$
- $0 \leq \text{Node.val} \leq 1000$
- $1 \leq k \leq sz$

通过次数 191,714 | 提交次数 294,198

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。

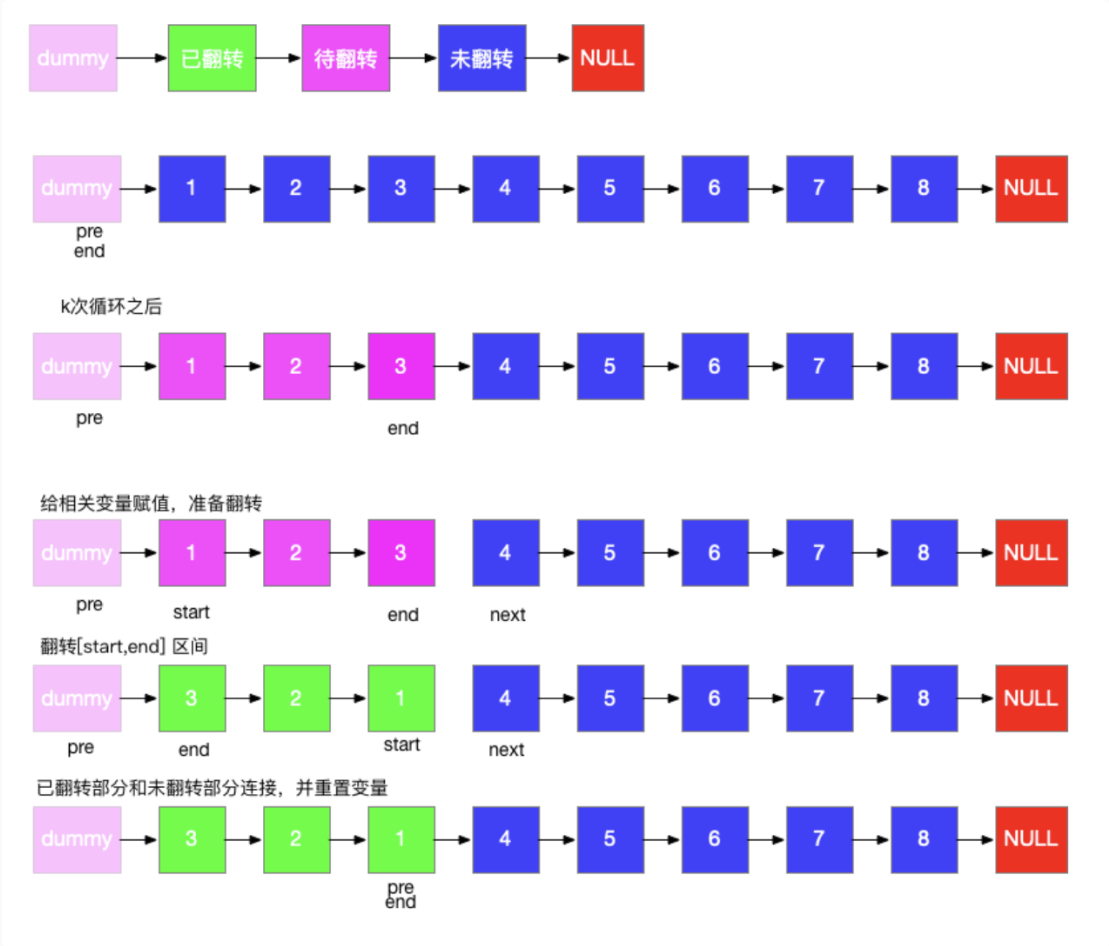
作者：reals

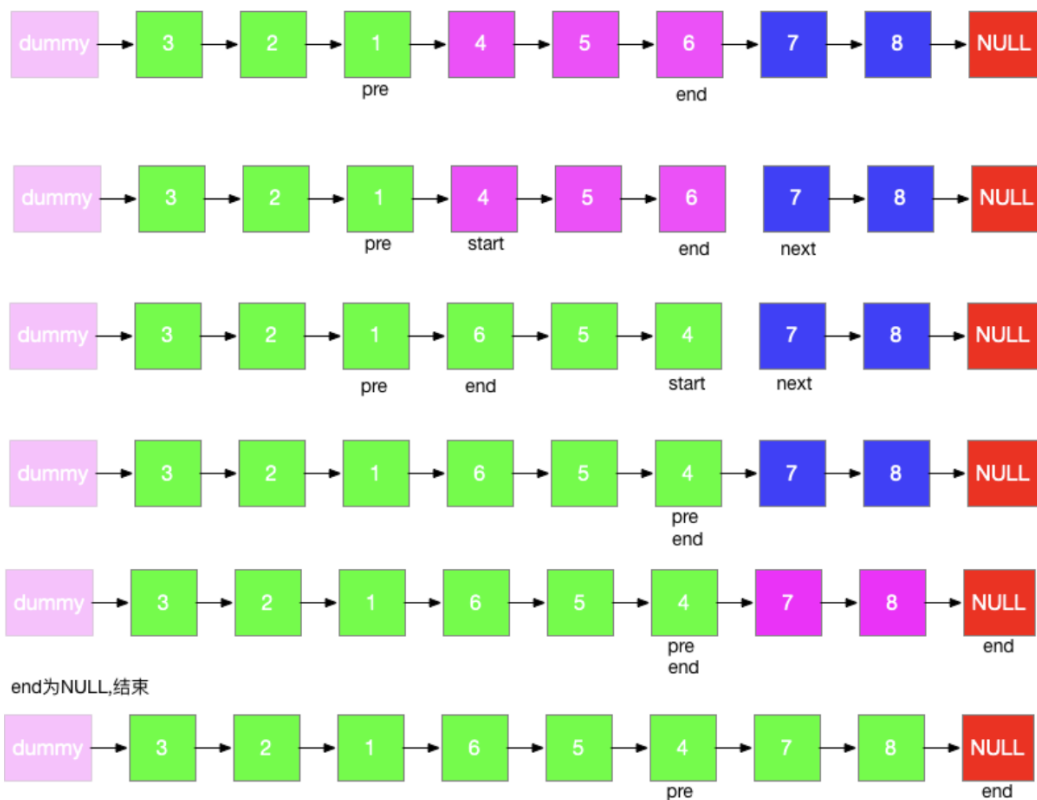
链接：<https://leetcode-cn.com/problems/reverse-nodes-in-k-group/solution/tu-jie-kge-yi-zu-fan-zhuan-lian-biao-by-user7208t/>

来源：力扣（LeetCode）

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

图示





```

1 // Java
2 // Time : 2021 - 07 - 14
3
4
5 public ListNode reverseKGroup(ListNode head, int k) {
6
7     if (head == null || head.next == null) {
8         return head;
9     }
10
11
12     // 定义一个假的节点。
13     ListNode dummy = new ListNode(0);
14
15
16     // 假节点的next指向head。
17     // dummy->1->2->3->4->5
18     dummy.next = head;
19
20
21     // 初始化pre和end都指向dummy
22     // pre指每次要翻转的链表的头结点的上一个节点
23     // end指每次要翻转的链表的尾节点
24     ListNode pre = dummy;
25     ListNode end = dummy;
26
27
28     while (end.next != null) {
29
30
31         // 循环k次，找到需要翻转的链表的结尾
32         // 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。

```

```

33 // dummy->1->2->3->4->5 若k为2, 循环2次, end指向2
34 for (int i = 0; i < k && end != null; i++) {
35     end = end.next;
36 }
37
38
39 //如果end==null, 即需要翻转的链表的节点数小于k, 不执行翻转。
40 if (end == null) {
41     break;
42 }
43
44
45 //先记录下end.next,方便后面链接链表
46 ListNode next = end.next;
47
48
49 //然后断开链表
50 end.next = null;
51
52
53 //记录下要翻转链表的头节点
54 ListNode start = pre.next;
55
56
57 //翻转链表,pre.next指向翻转后的链表: 1->2 变成2->1, dummy->2->1
58 pre.next = reverse(start);
59
60
61 //翻转后头节点变到最后, 通过.next把断开的链表重新链接。
62 start.next = next;
63
64
65 //将pre换成下次要翻转的链表的头结点的上一个节点。即start
66 pre = start;
67
68
69 //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
70 end = start;
71 }
72 return dummy.next;
73 }
74
75
76 //链表翻转
77 // 例子: head: 1->2->3->4
78 public ListNode reverse(ListNode head) {
79
80
81 //单链表为空或只有一个节点, 直接返回原单链表
82 if (head == null || head.next == null) {
83     return head;
84 }
85
86
87 //前一个节点指针
88 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 }
```