leetcode25.k个一组翻转链表————困难

笔记本:	leetcode刷题		
创建时间:	2021/8/8 19:55	**更新时间:**	2021/8/8 23:32
作者:	Zard		

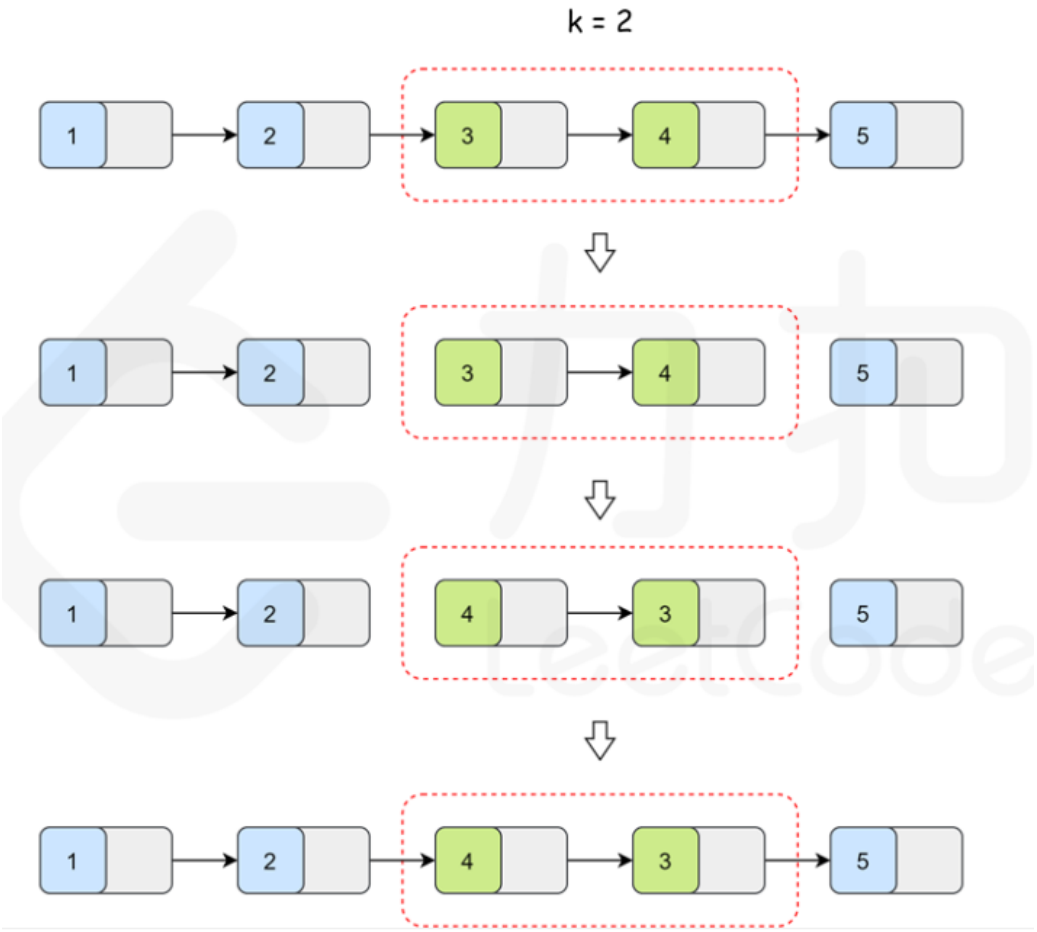
leetcode25.k个一组翻转链表————困难
题目描述:
给你一个链表，每k个节点一组进行翻转，请你返回翻转后的链表。
k是一个正整数，它的值小于或等于链表的长度。
如果节点总数不是k的整数倍，那么请将最后剩余的节点保持原有顺序。
进阶:
(1).设计一个只使用常数额外空间的算法。
(2).不能只是单纯地改变节点内部的值，而是需要实际进行节点交换。
示例 1:
输入：head = [1,2,3,4,5], k = 2
输出：[2,1,4,3,5]

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 <= sz <= 5000
0 <= Node.val <= 1000
1 <= k <= sz
方法一：模拟
思路与算法:
我们需要把链表节点按照k个一组分组，所以可以使用一个指针head依次指向每组的头节点。这个指针每次向前移动k步，直至链表结尾。对于每个分组，我们先判断它的长度是否等于k。若是，我们就翻转这部分链表，否则不需要翻转。
接下来的问题就是如何翻转一个分组内的子链表。但是对于一个子链表，除了翻转其本身之外，还需要将子链表的头部与上一个子链表连接，以及子链表的尾部与下一个子链表连接。



因此，在翻转链表的时候，我们不仅需要子链表头节点head，还需要有head的上一个节点pre，以便翻转完后把子链表再接回pre。
又一件麻烦事是：链表翻转之后，链表的头节点发生了变化，那么应该返回哪个节点呢？照理来说，前k个节点翻转之后，链表的头节点应该是第k个节点。那么要在遍历过程中记录第k个节点吗？但是如果链表里面没有k个节点，答案又还是原来的头节点。

but，我们之前创建了节点pre，这个节点一开始被连接到了头节点的前面，而无论之后链表有没有翻转，它的next指针都会指向正确的头节点。那么我们只要返回它的下一个节点就好了。

```
/**
* Definition for singly-linked list.
* struct ListNode {
*     int val;
*     ListNode *next;
*     ListNode() : val(0), next(nullptr) {}
*     ListNode(int x) : val(x), next(nullptr) {}
*     ListNode(int x, ListNode *next) : val(x), next(next) {}
* };
*/
class Solution {
public:
    pair<ListNode*,ListNode*> myReverse(ListNode* head,ListNode* tail){
        ListNode* pre=new ListNode(-1);
        ListNode* cur=head;
        while(pre!=tail){
            ListNode* next=cur->next;
            cur->next=pre;
            pre=cur;
            cur=next;
        }
            return {tail,head};
    }
    ListNode* reverseKGroup(ListNode* head, int k) {
        ListNode* dummy=new ListNode(-1);
        dummy->next=head;
```

```
        ListNode* pre=dummy;


        while(head!=NULL){
            ListNode* tail=pre;
            //查看剩余部分长度是否大于等于k
            for(int i=0;i<k;i++){
                tail=tail->next;
                if(tail==NULL){
                    return dummy->next;
                }
            }
            ListNode* next=tail->next;
            pair<ListNode*,ListNode*> result=myReverse(head,tail);
            head=result.first;
            tail=result.second;
            //tie(head,tail)=myReverse(head,tail);
            //把子链表重新接回原链表。
            pre->next=head;
            tail->next=next;
            pre=tail;
            head=tail->next;
        }
            return dummy->next;
    }
};
```