Leetcode python版 链表（困难）

23合并K个排序链表

题目：合并 *k*个排序链表，返回合并后的排序链表。请分析和描述算法的复杂度。

**示例:**

**输入:**

[

  1->4->5,

  1->3->4,

  2->6

]

**输出:** 1->1->2->3->4->4->5->6

思路：把链表都加入一个数组中，然后排序，再加到最后的链表中，算法复度为O(nlog(n))

代码：

1. # Definition for singly-linked list.
2. # class ListNode(object):
3. #     def \_\_init\_\_(self, x):
4. #         self.val = x
5. #         self.next = None
7. **class** Solution(object):
8. **def** mergeKLists(self, lists):
9. """
10. :type lists: List[ListNode]
11. :rtype: ListNode
12. """
13. res=[]
14. **for** i **in** lists:
15. **while** i:
16. res.append(i.val)
17. i=i.next
18. **if** res==[]:
19. **return** []
20. res.sort()
21. l=ListNode(0)
22. first=l
23. **while** res:
24. l.next=ListNode(res.pop(0))
25. l=l.next
26. **return** first.next

25 K个一组翻转链表

题目：给出一个链表，每 *k*个节点一组进行翻转，并返回翻转后的链表。

*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，那么将最后剩余节点保持原有顺序。

**示例 :**

给定这个链表：1->2->3->4->5

当 *k*= 2 时，应当返回: 2->1->4->3->5

当 *k*= 3 时，应当返回: 3->2->1->4->5

**说明 :**

* 你的算法只能使用常数的额外空间。
* **你不能只是单纯的改变节点内部的值**，而是需要实际的进行节点交换。

思路：将list的每k个节点作为一组，依次对每组进行翻转，当某组的节点数小于k时，直接返回head。

代码：

1. # Definition for singly-linked list.
2. # class ListNode(object):
3. #     def \_\_init\_\_(self, x):
4. #         self.val = x
5. #         self.next = None
7. **class** Solution(object):
8. **def** reverseKGroup(self, head, k):
9. """
10. :type head: ListNode
11. :type k: int
12. :rtype: ListNode
13. """
14. start = preNode =ListNode(0)
15. start.next = l = r = head
16. **while** True:
17. **for** i **in** range(k):
18. **if** r:
19. r = r.next
20. **else**:
21. **return** start.next
22. cur, nex = l, r
23. **for** i **in** range(k):
24. cur.next, cur, nex = nex, cur.next, cur
25. l, preNode.next, preNode = r, nex, l

思路2：其实以上代码最后两行看不懂意思，所以使用接下来这种方法

关于这个问题，首先我们很容易想到的思路就是通过两个指针指向一个group的头和尾，然后对这个group做reverse操作，如果这两个指针间的距离小于k，我们不进行操作。另外，我们要确定我们reverse的边界，我这里假设边界为（pre, lat）

pre lat

h -> 1 -> 2 -> 3 -> 4 -> 5

k = 3

接着，我们来看怎么进行reverse，首先定义两个指针lpre=pre.next和cur=lpre.next

pre lpre cur lat

h -> 1 -> 2 -> 3 -> 4 -> 5

k = 3

当cur!=lat的时候我们要做的操作就是lpre.next=cur.next、cur.next=pre.next、pre.next=cur、cur=lpre.next

pre lpre cur lat

-----------

| |

h 1 <- 2 3 -> 4 -> 5

| |

-----------

代码2：

1. # Definition for singly-linked list.
2. # class ListNode(object):
3. #     def \_\_init\_\_(self, x):
4. #         self.val = x
5. #         self.next = None
7. **class** Solution(object):
8. **def** \_reverseGroup(self, pre, lat):
9. lpre = pre.next
10. cur = lpre.next
11. **while** cur != lat:
12. lpre.next = cur.next
13. cur.next = pre.next
14. pre.next = cur
15. cur = lpre.next
17. **return** lpre
19. **def** reverseKGroup(self, head, k):
20. """
21. :type head: ListNode
22. :type k: int
23. :rtype: ListNode
24. """
25. h = ListNode(-1)
26. h.next = head
27. pre = h
28. cur = head
30. t = 1
31. **while** cur != None:
32. **if** t % k == 0:
33. pre = self.\_reverseGroup(pre, cur.next)
34. cur = pre.next
35. **else**:
36. cur = cur.next
37. t += 1
39. **return** h.next