剑指offer(python版) 13-16

13调整数组顺序使奇数位于偶数前面

题目：

思路：再弄一个list或是两个list 来存储其中的奇数或是偶数，最后进行合并操作

代码：

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. **def** reOrderArray(self, array):
4. # write code here
5. oddn=[]
6. evenn=[]
7. **for** i **in** range(len(array)):
8. **if** array[i]%2==0:
9. evenn.append(array[i])
10. **else**:
11. oddn.append(array[i])
12. oddn.extend(evenn)
13. **return** oddn

思路2：这道题应该算是考察排序的变形，所以想一下我们学过的排序，冒泡排序很是合适，这是第一种思路，以前是前一个数比后一个数大或是小进行相应的交换，也就是所谓的冒泡，这里的交换条件改成 前一个为 偶数后一个奇数就交换就好了。

这里有个易错的地方就是这里交换只能前后的交换，虽说似乎是冒泡，但是其实形似冒泡，但也不是冒泡。

代码：

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. **def** reOrderArray(self, array):
4. # write code here
5. change\_data = 0
6. **for** i **in** range(len(array)):
7. **for** j **in** range(len(array)-1,i,-1):
8. **if** (array[j-1]%2 == 0) **and** (array[j]%2 == 1):
9. change\_data = array[j-1]
10. array[j-1] = array[j]
11. array[j] = change\_data
12. **return** array

涉及到的新函数

extend（）:

14链表中倒数第k个结点

题目：输入一个链表，输出该链表中倒数第k个结点。

思路：使用快慢指针，走在前面的指针比后面的指针快k-1步，则当走在前面的指针走到头的时候，走在后面的指针正好到倒数第k个节点

本题要注意到代码的鲁棒性：程序在非正常情况下（输入不符合规定等）不会出错或崩溃

本题要注意到有以下三点：

1. 头结点不存在
2. k为0
3. 链表长度小于k

代码：

1. # -\*- coding:utf-8 -\*-
2. # class ListNode:
3. #     def \_\_init\_\_(self, x):
4. #         self.val = x
5. #         self.next = None
7. **class** Solution:
8. **def** FindKthToTail(self, head, k):
9. # write code here
10. **if** **not** head **or** **not** k:
11. **return** None
12. left,right=head,head
13. **for** i **in** range(k-1):
14. **if** **not** right.next:
15. **return** None
16. right=right.next
17. **while** right.next:
18. left=left.next
19. right=right.next
20. **return** left

15反转链表

题目：输入一个链表，反转链表后，输出新链表的表头

思路：用temp保存pHead的下一个节点。用newHead保存新的节点的开头。然后把pHead.next只想newHead，再移动一下 三个指针即可

代码：

1. # -\*- coding:utf-8 -\*-
2. # class ListNode:
3. #     def \_\_init\_\_(self, x):
4. #         self.val = x
5. #         self.next = None
6. **class** Solution:
7. # 返回ListNode
8. **def** ReverseList(self, pHead):
9. # write code here
10. **if** **not** pHead **or** **not** pHead.next:
11. **return** pHead
12. newHead=None
13. **while** pHead:
14. temp=pHead.next
15. pHead.next=newHead
16. newHead=pHead
17. pHead=temp
18. **return** newHead

16合并两个排序的链表

题目：输入两个单调递增的链表，输出两个链表合成后的链表，当然我们需要合成后的链表满足单调不减规则。

思路1：比较两个指针上的值哪个小，小的就填上，然后小指针就向后移动

代码：（超时了）

1. # -\*- coding:utf-8 -\*-
2. # class ListNode:
3. #     def \_\_init\_\_(self, x):
4. #         self.val = x
5. #         self.next = None
6. **class** Solution:
7. # 返回合并后列表
8. **def** Merge(self, pHead1, pHead2):
9. # write code here
10. **if** pHead1 == None:
11. **return** pHead2
12. **if** pHead2 == None:
13. **return** pHead1
15. p = ListNode(100)
16. root = p
17. **while** pHead1!=None **and** pHead2!=None:
18. **if** pHead1.val>=pHead2.val:
19. p.next = pHead2
20. p = p.next
21. pHead2 = pHead2.next
22. **else**:
23. p.next = pHead1
24. p = p.next
26. **if** pHead1 != None:
27. p.next = pHead1
28. **if** pHead2 != None:
29. p.next = pHead2
30. **return** root.next

思路2：采用递归

代码：

1. # -\*- coding:utf-8 -\*-
2. # class ListNode:
3. #     def \_\_init\_\_(self, x):
4. #         self.val = x
5. #         self.next = None
6. **class** Solution:
7. # 返回合并后列表
8. **def** Merge(self, pHead1, pHead2):
9. # write code here
10. **if** pHead1 == None:
11. **return** pHead2
12. **if** pHead2 == None:
13. **return** pHead1
14. **if** pHead1.val >= pHead2.val:
15. pHead2.next = self.Merge(pHead1,pHead2.next)
16. **return** pHead2
17. **else**:
18. pHead1.next = self.Merge(pHead1.next,pHead2)
19. **return** pHead1