第8题——反转链表

定义一个函数，输入一个链表的头节点，反转该链表并输出反转后链表的头节点。

示例:

```
输入: 1->2->3->4->5->NULL
```

```
输出: 5->4->3->2->1->NULL
```

限制：0 <= 节点个数 <= 5000

头插法（1）

使用头插法，将旧链表的数据转移到新链表即可

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
struct ListNode* reverseList(struct ListNode* head){
    struct ListNode *temp=NULL;//定义临时节点
    struct ListNode *newlist=NULL;//定义新的头节点
    while(head!=NULL)
    {
        temp=head;
        head=head->next;//将原本链表的第一个结点摘下给temp
        temp->next=newlist;
        newlist=temp;//头插法
    }
    return newlist;
}
```

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
```

```
        ListNode* new_head=NULL;
        ListNode* temp=NULL;
        while(head!=NULL)
        {
            //从链表摘下一个元素(temp指向该元素)
            temp=head;
            head=head->next;//head后移一位

            //头插法
            temp->next=new_head;
            new_head=temp;
        }
        return new_head;
    }
};
```

三指针迭代（2）

想了想，需要把链表逆置，那么可以把每一个结点的指向反转即可； 我先定义一个指向空的结点second； 再定义first指向head; 再定义一个first的后继结点temp; 然后就比较简单了： 我们有三个指针。后面两个改变指向（反转），然后三个指针同时前移即可。

如图

第一次替换前：

second=NULL;

data	x1	x2	x3	x4	...
指针域	x2	x3	x4	...	...
	first	temp			

第一次替换后：

data	x1	x2	x3	x4	...
指针域	NULL	x3	x4	...	...
	second	first	temp		

第二次替换后：

data	x1	x2	x3	x4	...
指针域	NULL	x1	x3->x4	x4->...	...
		second	first	temp	

当first->next==NULL跳出循环，此时first后的second结点指向的就是头节点，返回second即可。

```
struct ListNode* reverseList(struct ListNode* head){
    struct ListNode *first=head;
    struct ListNode *second=NULL;
    struct ListNode *temp=head;
    while(first!=NULL)
    {
        temp=temp->next;
        first->next=second;
        second=first;
        first=temp;
    }
    return second;
}
```

递归法（3）

1. 创建临时节点，将原本链表的值赋给临时节点

2. head->next-next=head。实际上就是head的下一个节点指向head,完成反转

比如：1->2->3->4->5

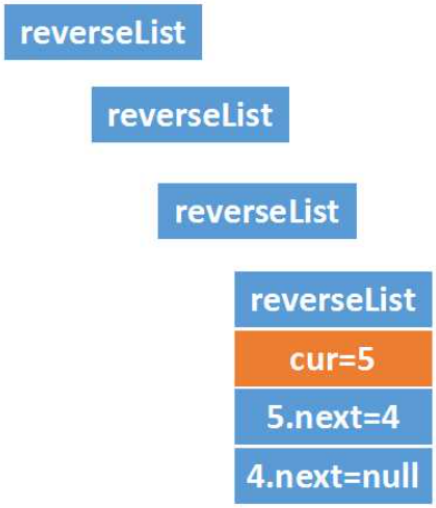
（1）到第五层。**head=5**。不满足条件了,开始回溯。

（2）到第四层。**head=4**。执行newHead=reverseList(head->next)后，newHead=5;

此时，head->next->next=head相当于4->next->next=4。也就是**5->next=4**。

紧接着再执行head->next=NULL。4指向NULL。因此，**此时5指向4**。返回**newHead也就是5**。


```
public ListNode reverseList(ListNode head) {
        if(head==null || head.next==null) {
                return head;
        }
        ListNode cur = reverseList(head.next);
        head.next.next = head;
        head.next = null;
        return cur;
}
```



（3）到第三层。**head=3**。执行newhead=reverseList(head->next)后。newHead=4;

此时，head->next->next=head相当于3->next->next=3。也就是**4->next=3**。

紧接着再执行head->next=NULL。3指向NULL。因此，此时4指向3。返回**newHead也就是4**。


```
public ListNode reverseList(ListNode head) {
        if(head==null || head.next==null) {
                return head;
        }
        ListNode cur = reverseList(head.next);
        head.next.next = head;
        head.next = null;
        return cur;
}
```

（4）如此回溯到第一层就完成了链表的反转。

```
struct ListNode* reverseList(struct ListNode* head) {
    if (head == NULL || head->next == NULL) {
        return head;
    }
    struct ListNode* newHead;
    newHead = reverseList(head->next);//创建临时节点
    head->next->next = head;
    head->next = NULL;
    return newHead;
}
```