```
/*
反转一个单链表。

示例:

输入: 1->2->3->4->5->NULL
输出: 5->4->3->2->1->NULL

进阶:
你可以迭代或递归地反转链表。你能否用两种方法解决这道题?

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/reverse-linked-list
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析:

- 翻转数据结构，我首先想到的是借助栈这个数据结构，利用栈只压入val或者压入节点都可以，实现了方法一和方法二。
- 反转链表，迭代法，可以把分成两个部分：已经反转的部分和未反转的部分，其中$head_temp$指向已经反转部分的开始节点，$tail_temp$指向已经反转部分的结束节点,$temp$指向下一个要反转的节点，显然当$temp \neq NULL$时，$tail\_temp \rightarrow next = temp \rightarrow next$,$temp \rightarrow next = head\_temp$,$head\_temp = temp$，$temp = tail\_temp->next$,即可把指向$temp$的指向插入反转部分。（方法三）

方法一:C++,Stack_val

```
/**
 * 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)
    {
        stack<int>  si                  ;
        ListNode*   temp        = head  ;
        int         val_temp    = 0     ;
```

```
        while(temp!=NULL)
        {
            si.push(temp->val);
            temp = temp->next;
        }
        temp = head;
        while(temp!=NULL)
        {
            temp->val = si.top();
            si.pop();
            temp = temp->next;
        }
        return head;
    }
};


/*
执行结果:
通过
显示详情
执行用时 :16 ms, 在所有 C++ 提交中击败了59.98% 的用户
内存消耗 :9.2 MB, 在所有 C++ 提交中击败了20.28%的用户
*/
```

方法二:C++,Stack_node

```
/**
 * 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)
        {
            stack<struct ListNode*> sl          ;
            ListNode*               temp = head ;

            while(temp!=NULL)
            {
                sl.push(temp);
                temp = temp->next;
            }
            if(sl.empty()!=true)
            {
                head = sl.top()     ;
```

```
                temp = head         ;
                sl.pop()            ;
                while(sl.empty()!=true)
                {
                    temp->next = sl.top();
                    sl.pop();
                    temp = temp -> next;
                }
                temp->next = NULL;

            }
            return head;
        }
};


/*
执行结果:
通过
显示详情
执行用时 :12 ms, 在所有 C++ 提交中击败了87.55% 的用户
内存消耗 :9.2 MB, 在所有 C++ 提交中击败了14.09%的用户
*/
```

方法三:C,迭代法

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */

struct ListNode* reverseList(struct ListNode* head)
{
    struct ListNode* head_temp  = NULL;
    struct ListNode* tail_temp  = NULL;
    struct ListNode* temp       = NULL;

    if(  ( head       == NULL )
       ||( head->next == NULL )
    )
    {
        return head;
    }
    else
    {
        head_temp   = head              ;
        tail_temp   = head              ;
        temp        = tail_temp->next   ;
        while(1)
```

```
        {
            if(temp==NULL)
            {
                break;
            }
            else
            {
                tail_temp -> next   = tail_temp -> next -> next;
                temp->next          = head_temp;
                head_temp           = temp;
                temp                = tail_temp -> next;
            }
        }
    }
    return head_temp;
}

/*
执行结果:
通过
显示详情
执行用时 :4 ms, 在所有 C 提交中击败了93.16% 的用户
内存消耗 :7.6 MB, 在所有 C 提交中击败了23.61%的用户
*/
```

AlimyBreak
2019.07.28