剑指 Offer II 026. 重排链表

给定一个单链表 L 的头节点 head ，单链表 L 表示为：

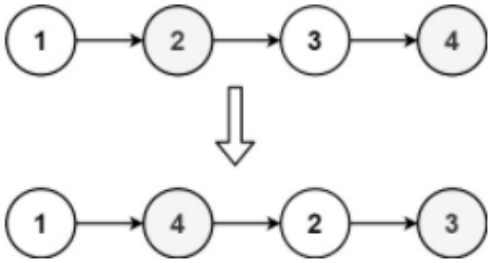
$L_0 \rightarrow L_1 \rightarrow \ldots \rightarrow L_{n-1} \rightarrow L_n$

请将其重新排列后变为：

$L_0 \rightarrow L_n \rightarrow L_1 \rightarrow L_{n-1} \rightarrow L_2 \rightarrow L_{n-2} \rightarrow \ldots$

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。

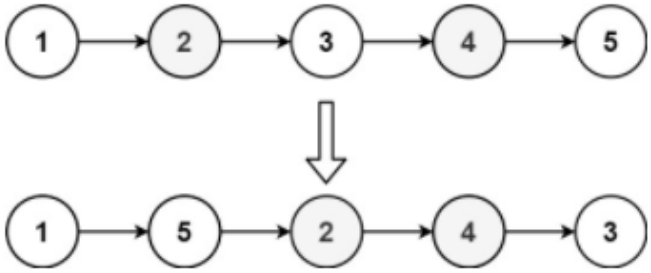
示例 1:



输入： head = [1,2,3,4]

输出： [1,4,2,3]

示例 2:



输入： head = [1,2,3,4,5]

输出： [1,5,2,4,3]

```
/**
 * 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:
    void reorderList(ListNode* head) {
        if(head==nullptr)
            return;
        //（1）快慢指针找到中间节点
        ListNode* fast=head;
        ListNode* slow=head;
        while(fast->next!=nullptr&&fast->next->next!=nullptr)
        {
            slow=slow->next;
            fast=fast->next->next;
        }
        //（2）后半部分翻转
        ListNode* secondlist=slow->next;
        slow->next=nullptr;
        ListNode* prev=nullptr;
        while(secondlist)
        {
            ListNode* nextnode=secondlist->next;
            secondlist->next=prev;
            prev=secondlist;
            secondlist=nextnode;
        }
        //（3）将前后两部分的两个链表合并
        secondlist=prev;
        ListNode* firstlist=head;
        while(firstlist!=nullptr&&secondlist!=nullptr)
        {
            ListNode* tmp=firstlist->next;
            firstlist->next=secondlist;
            firstlist=tmp;
            tmp=secondlist->next;
            secondlist->next=firstlist;
            secondlist=tmp;
        }
    }
};
```