# 题目

给定一个单链表 L 的头节点 head ，单链表 L 表示为：

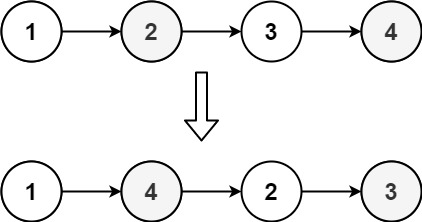
 L0 → L1 → … → Ln-1 → Ln

请将其重新排列后变为：

L0 → Ln → L1 → Ln-1 → L2 → Ln-2 → …

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。

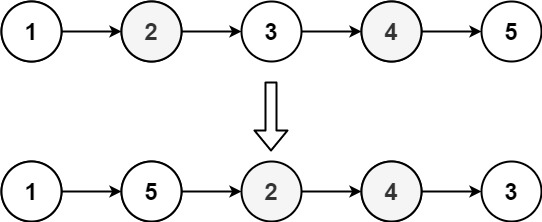
示例 1:



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]

提示：

链表的长度范围为 [1, 5 \* 104]

1 <= node.val <= 1000

# 分析

## 方法一：线性表

**思路：**

因为链表不支持下标访问，所以我们无法随机访问链表中任意位置的元素。

因此比较容易想到的一个方法是，我们利用线性表存储该链表，然后利用线性表可以下标访问的特点，直接按顺序访问指定元素，重建该链表即可。

**代码：**

/\*\*

 \* 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)

            return;

        vector<ListNode\*> vec;

        while(head)

        {

            vec.push\_back(head);

            head = head->next;

        }

        int i=0;

        int j=vec.size()-1;

        while(i<j)

        {

            vec[i]->next = vec[j];

            i++;

            if(i==j)

                break;

            vec[j]->next = vec[i];

            j--;

        }

        vec[i]->next = nullptr;

    }

};

复杂度分析

时间复杂度：O(N)，其中 N 是链表中的节点数。

空间复杂度：O(N)，其中 N 是链表中的节点数。主要为线性表的开销。