143. Reorder List

You are given the head of a singly linked-list. The list can be represented as:

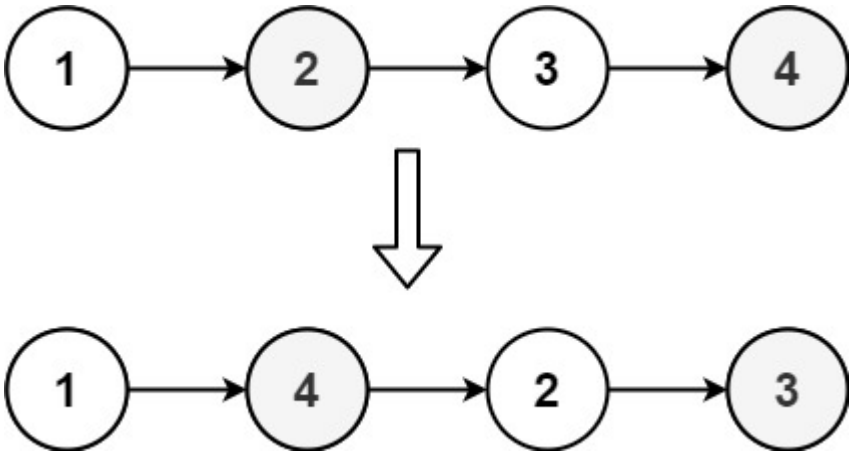
L0 → L1 → … → Ln - 1 → Ln

Reorder the list to be on the following form:

L0 → Ln → L1 → Ln - 1 → L2 → Ln - 2 → …

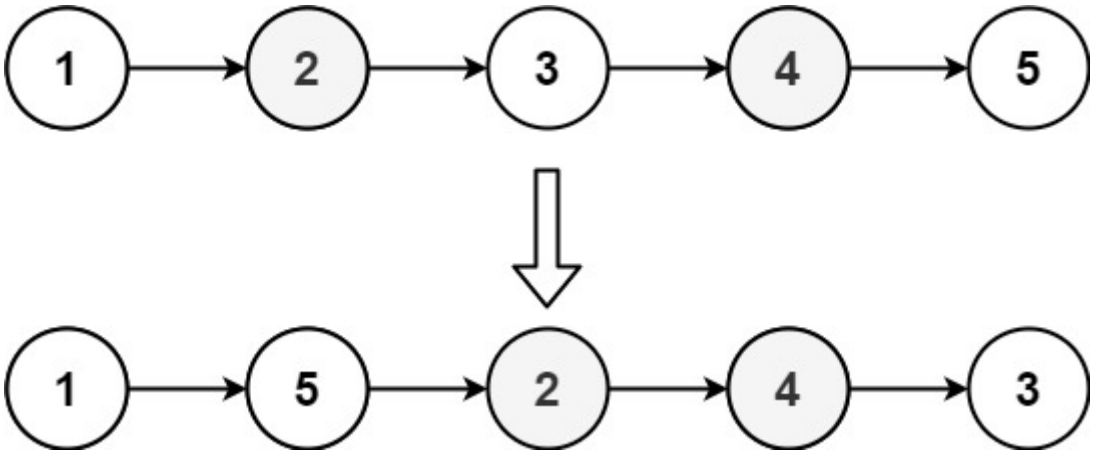
You may not modify the values in the list's nodes. Only nodes themselves may be changed.

Example 1:



Input: head = [1,2,3,4]
Output: [1,4,2,3]

Example 2:



Input: head = [1,2,3,4,5]
Output: [1,5,2,4,3]

Constraints:

- The number of nodes in the list is in the range `[1, 5 * 104]`.
- `1 ≤ Node.val ≤ 1000`

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next;
}
 * }
 */

class Solution {
    public ListNode reverse(ListNode head){
        if(head == null){
            return null;
        }
        ListNode newHead = null;
        ListNode curr = head;
        ListNode nextNode = null;

        while(curr ≠ null){
            nextNode = curr.next;
            curr.next = newHead;
            newHead = curr;
            curr = nextNode;
        }

        return newHead;
    }

    public void merge (ListNode list1, ListNode list2){
        while(list2 ≠ null){
            ListNode nextNode = list1.next;
            list1.next = list2;
            list1 = list2;
            list2 = nextNode;
        }
    }

    public void reorderList(ListNode head) {
        if (head == null || head.next == null){
            return;
        }
        ListNode slow = head;
        ListNode fast = head;
```

```
        ListNode prev = head;

        while(fast ≠ null && fast.next ≠ null){
            prev = slow;
            fast = fast.next.next;
            slow = slow.next;
        }

        prev.next = null;
        ListNode list1 = head;
        ListNode list2 = reverse(slow);
        merge(list1, list2);
    }
}
```