<https://leetcode.com/problems/reorder-list/>

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

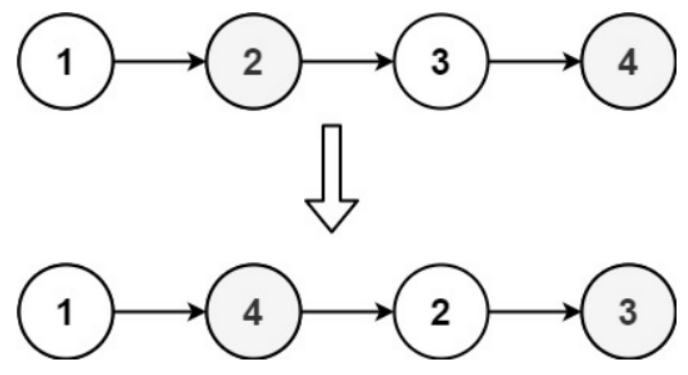
L0 → L1 → … → Ln - 1 → Ln

*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.

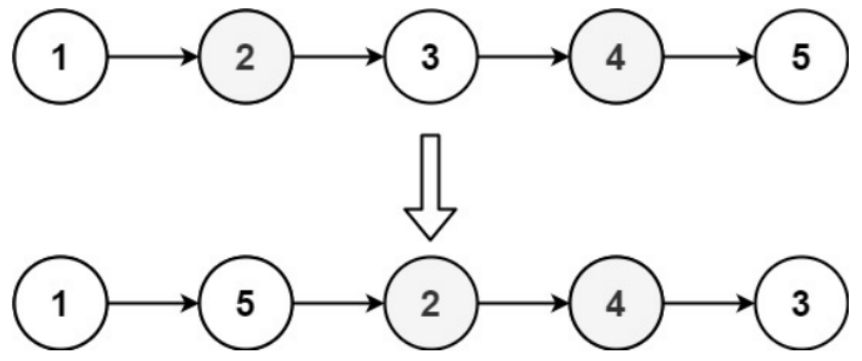
**Example 1:**



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

**Attempt 1: 2023-02-20**

**Solution 1: Fast and Slow pointer + Reverse Linked List + Connect two sub-lists inplace (10 min)**

/\*\*

\* 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 void reorderList(ListNode head) {

if(head == null || head.next == null) {

return;

}

// Find the pre-node of second half head

ListNode slow = head;

ListNode fast = head;

while(fast.next != null && fast.next.next != null) {

fast = fast.next.next;

slow = slow.next;

}

// 'slow' in this style will always point to pre-node of second half head

ListNode secondHalfHead = slow.next;

// Cut the connection between first and second half

slow.next = null;

// Reverse second half

ListNode revSecondHalfHead = reverse(secondHalfHead);

// Since no return required, have to connect two sublists inplace, no dummy head

while(head != null && revSecondHalfHead != null) {

ListNode node1 = head.next;

ListNode node2 = revSecondHalfHead.next;

revSecondHalfHead.next = head.next;

head.next = revSecondHalfHead;

head = node1;

revSecondHalfHead = node2;

}

}

private ListNode reverse(ListNode secondHalfHead) {

ListNode prev = null;

ListNode cur = secondHalfHead;

while(cur != null) {

ListNode next = cur.next;

cur.next = prev;

prev = cur;

cur = next;

}

return prev;

}

}

**Refer to**

<https://leetcode.com/problems/reorder-list/solutions/44992/java-solution-with-3-steps/comments/155674>

public void reorderList(ListNode head) {

if (head == null) {

return;

}

// Find the middle node

ListNode slow = head, fast = head.next;

while (fast != null && fast.next != null) {

slow = slow.next;

fast = fast.next.next;

}

// Reverse the second half

ListNode head2 = reverse(slow.next);

slow.next = null;

// Link the two halves together

while (head != null && head2 != null) {

ListNode tmp1 = head.next;

ListNode tmp2 = head2.next;

head2.next = head.next;

head.next = head2;

head = tmp1;

head2 = tmp2;

}

}

private ListNode reverse(ListNode n) {

ListNode prev = null;

ListNode cur = n;

while (cur != null) {

ListNode tmp = cur.next;

cur.next = prev;

prev = cur;

cur = tmp;

}

return prev;

}