Documentation for the "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 in 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, 50,000].
- 1 <= Node.val <= 1000

Solution Approach

To reorder the list as specified, follow these steps:

1. **Find the Middle of the List**
   - *Objective:* Identify the middle of the list to divide it into two halves.
   - *Method:* Use two pointers, slow and fast. Move slow by one step and fast by two steps until fast reaches the end of the list. When fast reaches the end, slow will be at the middle of the list.

2. **Reverse the Second Half of the List**
   - *Objective:* Reverse the order of nodes in the second half of the list.
   - *Method:* Starting from the node after the middle (the beginning of the second half), reverse the linked list using standard reversal techniques. This involves re-linking nodes in reverse order until the end of the second half is reached.

3. **Merge the Two Halves**
   - *Objective:* Merge the first half and the reversed second half into the desired reordered form.
   - *Method:* Use two pointers, first and second, to traverse the first half and the reversed second half respectively. Alternately link nodes from the two halves until all nodes are processed.

Detailed Steps

1. **Finding the Middle:**
   - Initialize two pointers, slow and fast, both starting at the head.
   - Move slow by one step and fast by two steps in each iteration.
   - When fast reaches the end, slow will be at the midpoint of the list.

2. **Reversing the Second Half:**
    - Set slow.next to None to separate the two halves.
    - Reverse the list starting from slow.next using iterative reversal techniques.
    - The new head of the reversed list will be assigned to second.

3. **Merging the Halves:**
    - Initialize first at the head and second at the head of the reversed second half.
    - Alternately link nodes from first and second while updating pointers.

Complexity Analysis

- **Time Complexity:** O(n), where n is the number of nodes in the list. The solution involves a single pass to find the middle, another pass to reverse the second half, and a final pass to merge the two halves.

- **Space Complexity:** O(1), as the solution uses only a few pointers and does not require extra space proportional to the input size.