**143. Reorder List**

[**https://leetcode.com/problems/reorder-list/**](https://leetcode.com/problems/reorder-list/)

1. **Listen**

**Problem Description:**

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.

**Input**:

head of a singly **linked list**

L0 → L1 → … → Ln - 1 → Ln

**Goal**:

Reorder the list to be on the following form

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

**Return**:

head of reordered list

1. **Example**

**Constraints:**

* The number of nodes in the list is in the range [1, 5 \* 104].
* 1 <= Node.val <= 1000
* You may **not modify** the **values** in the list's nodes. Only the position (next pointers) may be changed.

**Test Cases:**

* linked list is even length
* linked list is odd length

**Edge Cases:**

* linked list is empty
* single node in linked list

**Questions & Assumptions:**

* none

**Example 1**

Diagram

Description automatically generated

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

**Output:** [1,4,2,3]

**Example 2**

Diagram

Description automatically generated

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

**Output:** [1,5,2,4,3]

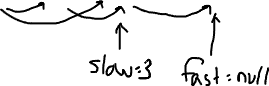
1. **Brute Force**

**Solution 1:**

* Traverse to middle of list.
* Use a stack.
* Put the second half of nodes onto the stack.
* Create a pointer pointing to the head of the list called curr.
* Assign curr’s next pointer to the top of the stack.
* Pop the stack and iterate the list pointer twice.

Diagram

Description automatically generated



Stack:

Diagram

Description automatically generated

Diagram

Description automatically generated

First half of list:

Diagram

Description automatically generated

Combine for every node in the first half of the list, append the top of the stack

List:

Diagram

Description automatically generated

Stack:

Diagram

Description automatically generated

List:

Diagram

Description automatically generated

Stack:

empty

We can do this in O(n/2 + n/2) = O(n) time and O(n/2) = O(n) space.

1. **Optimize**

We can save on the unnecessary space by doing in without an extra data structure.

We complete this in three parts

1. Traverse to the middle of the list (using two-pointers slow and fast)
2. Reverse the second half of the list (have one pointer to beginning and end of list)
3. Combine two lists into one alternating list

We can do this in O(n + n + n) time and O(1) space

1. **Implement**

public void reorderList(ListNode head) {

if(head==null||head.next==null) return;

//Find the middle of the list

ListNode p1=head;

ListNode p2=head;

while(p2.next!=null&&p2.next.next!=null){

p1=p1.next;

p2=p2.next.next;

}

//Reverse the half after middle 1->2->3->4->5->6 to 1->2->3->6->5->4

ListNode preMiddle=p1;

ListNode preCurrent=p1.next;

while(preCurrent.next!=null){

ListNode current=preCurrent.next;

preCurrent.next=current.next;

current.next=preMiddle.next;

preMiddle.next=current;

}

//Start reorder one by one 1->2->3->6->5->4 to 1->6->2->5->3->4

p1=head;

p2=preMiddle.next;

while(p1!=preMiddle){

preMiddle.next=p2.next;

p2.next=p1.next;

p1.next=p2;

p1=p2.next;

p2=preMiddle.next;

}

}