Reorder List

Given a singly linked list L: $L_0 \rightarrow L_1 \rightarrow \ldots \rightarrow L_{n-1} \rightarrow L_n$,
reorder it to: $L_0 \rightarrow L_n \rightarrow L_1 \rightarrow L_{n-1} \rightarrow L_2 \rightarrow L_{n-2} \rightarrow \ldots$

You must do this in-place without altering the nodes' values.

For example,
Given `{1,2,3,4}`, reorder it to `{1,4,2,3}`.

Solution 1

This question is a combination of **Reverse a linked list I & II** . It should be pretty straight forward to do it in 3 steps :)

```
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;
            }
        }
```

written by wanqing original link here

Solution 2

```
// O(N) time, O(1) space in total
void reorderList(ListNode *head) {
    if (!head || !head->next) return;

    // find the middle node: O(n)
    ListNode *p1 = head, *p2 = head->next;
    while (p2 && p2->next) {
        p1 = p1->next;
        p2 = p2->next->next;
    }

    // cut from the middle and reverse the second half: O(n)
    ListNode *head2 = p1->next;
    p1->next = NULL;

    p2 = head2->next;
    head2->next = NULL;
    while (p2) {
        p1 = p2->next;
        p2->next = head2;
        head2 = p2;
        p2 = p1;
    }

    // merge two lists: O(n)
    for (p1 = head, p2 = head2; p1; ) {
        auto t = p1->next;
        p1 = p1->next = p2;
        p2 = t;
    }

    //for (p1 = head, p2 = head2; p2; ) {
    //    auto t = p1->next;
    //    p1->next = p2;
    //    p2 = p2->next;
    //    p1 = p1->next->next = t;
    //}
}
```

written by shichaotan original link here

Solution 3

```
# Splits in place a list in two halves, the first half is >= in size than the seco
nd.
# @return A tuple containing the heads of the two halves
def _splitList(head):
    fast = head
    slow = head
    while fast and fast.next:
        slow = slow.next
        fast = fast.next
        fast = fast.next

    middle = slow.next
    slow.next = None

    return head, middle

# Reverses in place a list.
# @return Returns the head of the new reversed list
def _reverseList(head):

  last = None
  currentNode = head

  while currentNode:
    nextNode = currentNode.next
    currentNode.next = last
    last = currentNode
    currentNode = nextNode

  return last

# Merges in place two lists
# @return The newly merged list.
def _mergeLists(a, b):

    tail = a
    head = a

    a = a.next
    while b:
        tail.next = b
        tail = tail.next
        b = b.next
        if a:
            a, b = b, a

    return head


class Solution:

    # @param head, a ListNode
    # @return nothing
    def reorderList(self, head):
```

```
        if not head or not head.next:
            return

        a, b = _splitList(head)
        b = _reverseList(b)
        head = _mergeLists(a, b)
```

written by riccardo original link here

From Leetcoder.