#LeetCode143

https://leetcode.com/problems/reorder-list/description/

Yifeng Zeng

#题目描述

Reorder List

#思路报告

我们假定要reorder的list是dummy->1->2->3->4->5->…->n-1->n->Null
最primitive的想法是：

1. 先遍历一边list得到有n个点，先把1拿出来，并且把2记录下来，再从2找到n, 把n放到1后面，得到dummy->1->n->Null
2. 把2拿出来，并把3记录下来，再次3找到n-1，把他们连起来得到dummy->1->n->2->n-1->Null。
3. 重复上述步骤直到中点

这个时间复杂度应该是O(n^2)。那么我们怎么优化呢。

首先，我们找n, n-1, n-2…每一次都是遍历list，所以是O(n^2)，那么如果我们有一个list是n->n-1->n-2->…那么我们就可以很方便的找到下一个candidate，所以我们想到了reverse linked list。而我们只需要翻转后半部的linked list，所以我们需要找到中点。找中点的操作我使用最常见的快慢指针，这里我就不介绍了。

代码如下：

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
class Solution {
    public void reorderList(ListNode head) {
        if (head == null) {
            return;
```

```
        }

        // find mid
        ListNode mid = findMid(head);
        // System.out.println(mid.val);

        // reverse
        ListNode tail = reverseList(mid.next);
        // tail will be the start of the second half
        mid.next = null; // actuall we don't need this
        // tail is now the start of the reversed second half
        // print(head);
        // print(tail);

        // now merge two list head & tail
        ListNode dummy = new ListNode(0);
        ListNode cur = dummy;
        while (head != null && tail != null) {
            cur.next = head;
            head = head.next;
            cur = cur.next;
            cur.next = tail;
            tail = tail.next;
            cur = cur.next;
        }
        cur.next = head;

        // return dummy.next;
    }

    private ListNode findMid(ListNode head) {
        ListNode slow = head;
        ListNode fast = head.next;
        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
        }
        return slow;
    }

    private ListNode reverseList(ListNode head) {
        ListNode prev = null;
        while (head != null) {
            ListNode next = head.next;
            head.next = prev;
            prev = head;
            head = next;
        }
        return prev;
    }

    private void print(ListNode head) {
        while (head != null) {
            System.out.print(head.val + " ");
            head = head.next;
        }
```

```
            System.out.println();
        }
    }
```

#套路总结

- 程序的模块化很重要，单独把findMid和reverseList函数提出来这样主函数会看起来比较清晰。
- 当找n,n-1,n-2...的时候前面的遍历都是重复的，所以想到反过来能不能行，所以想到了reverse后面一半。