跟着PDF刷。

**141. Linked List Cycle I(判断环是否存在)**

利用两个指针，快慢指针，快指针一次走两步，慢指针一次走一步，如果一段时间后两者相遇，则必定有环存在。

要注意快指针的下一个指针是否为空指针，否则fast.next.next会出现编译错误。

**142. Linked List Cycle II(找到环开始的地方)**

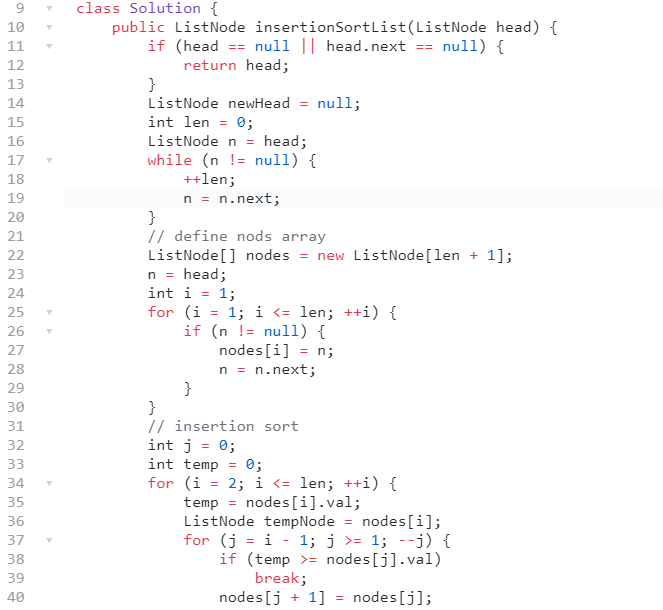
快慢指针相遇后，将慢指针定位头指针，两个指针皆一次走一步，相遇的地方就是环开始的地方。具体数学原理看PDF。

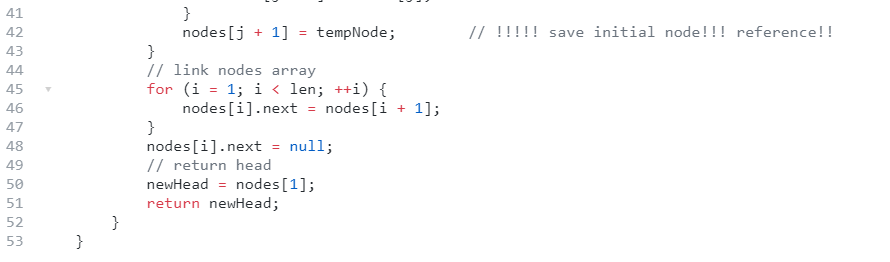
这两个题目在书本上都有，比较简单。

**147. Insertion Sort List(利用插入排序排序链表)**

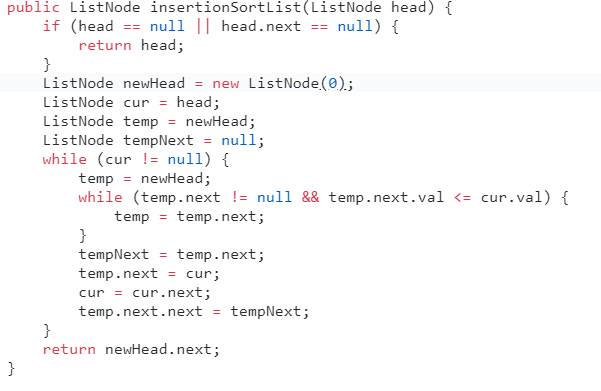
插入排序是稳定的排序，时间复杂度O(n^2)，原理是对于数组下标从2到len的元素(数组长度为len+1,从1开始)，往前寻找合适的位置，每个元素后移。

My own solution如下，时间复杂度O(n^2)，空间复杂度O(n)：





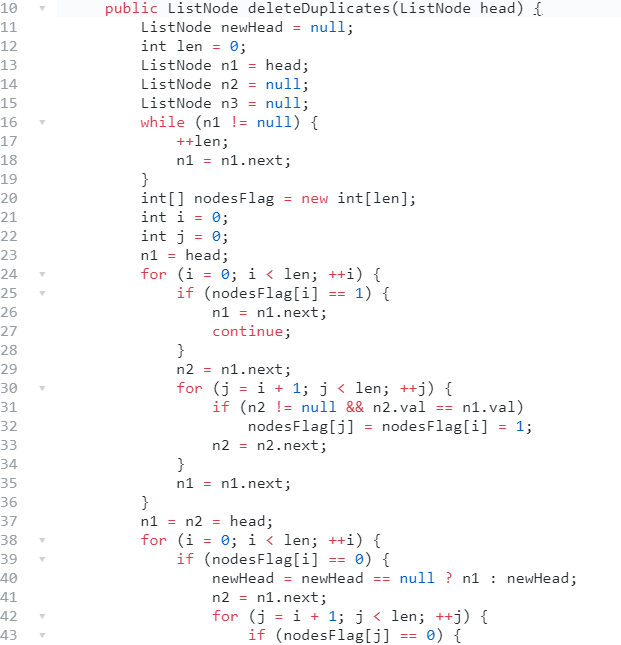
PDF的方法是设立一个新的节点(新的头节点)，对于原链表从头至尾遍历，每个节点都插入到新链表合适的位置，空间复杂度为O(1)。

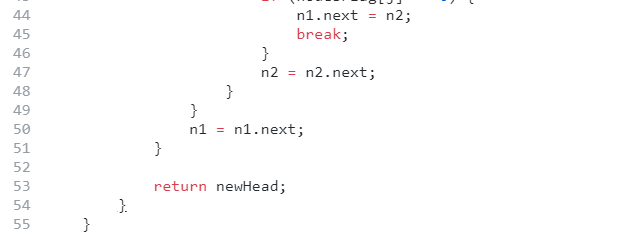


新的头节点，是一个假设节点(“dummy”)，如果没有它，则新链表头节点可能不断变化；其次，开辟新的“空间”也是我没有想到的，我惯性思维是在原链表进行操作。

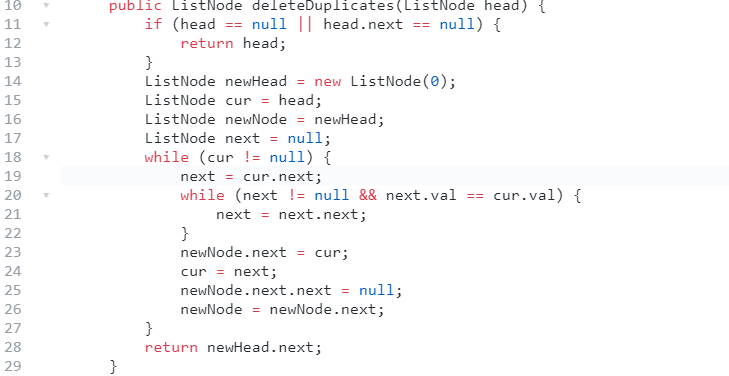
**83. Remove Duplicates from Sorted List(**Given a sorted linked list, delete all duplicates such that each element appear only once.**)**

自法，声明一个标记数组，与链表对应，如果重复，则标记为1，否则为0。利用两层循环标记后，再遍历链表，将标记为0的节点连接：





根据上一题的触发后改进如下：



才发现其实不要假设表头(newHead)，之前有是因为插入排序可能是表头变换，这次是不会的…

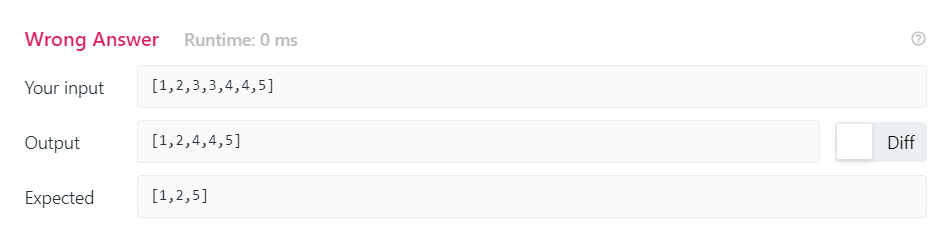
PDF的做法更简单更明了…只需在原链表判断并删除即可。

**82. Remove Duplicates from Sorted List II(删掉所有数值重复的节点)**

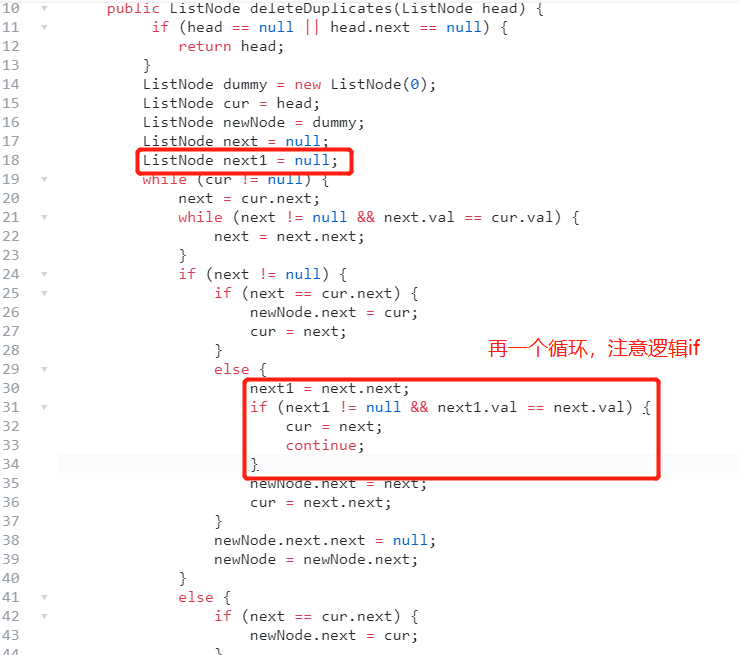
本来我刚开始做这题的思路跟上题一样，空间复杂度O(n)，时间复杂度O(n^2)，改进如下：

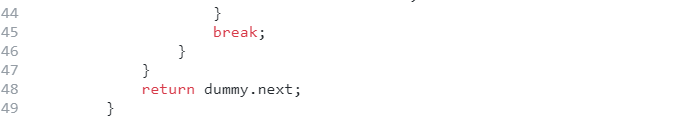


发现不对：



需要再加一层循环判断：





空间复杂度O(1)，时间复杂度O(n)。

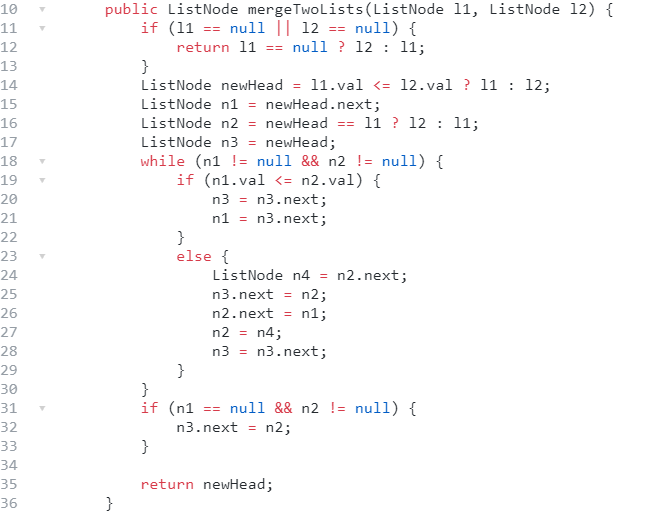
PDF的做法跟我的差不多，但更明白一些，while判断条件是p&&p->next，思路差不多，但代码流程更简便。

**21. Merge Two Sorted Lists(合并两有序链表并返回新的有序链表)**

书上做过，注意的是要有一个主链表和副链表和代码流程。当while循环后记得处理剩余链表。

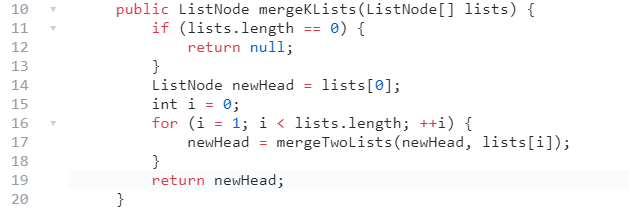
PDF思路也差不多。

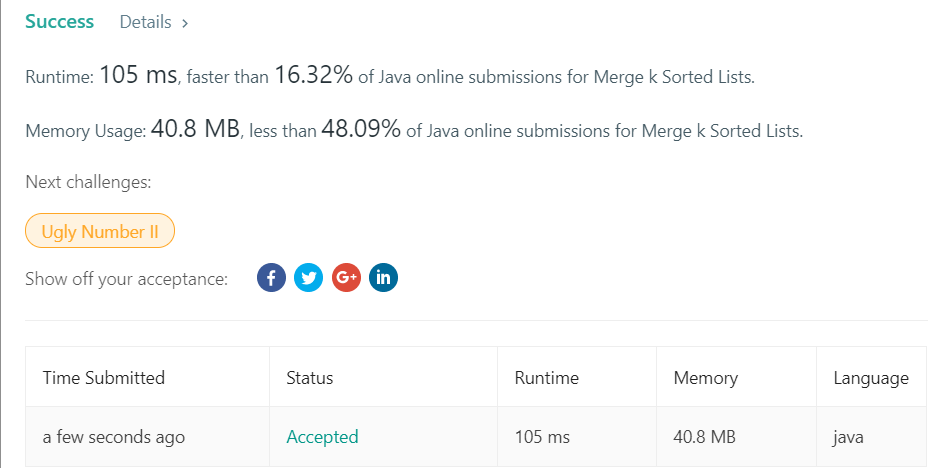
自法：



**23. Merge k Sorted Lists(合并k个有序链表)**

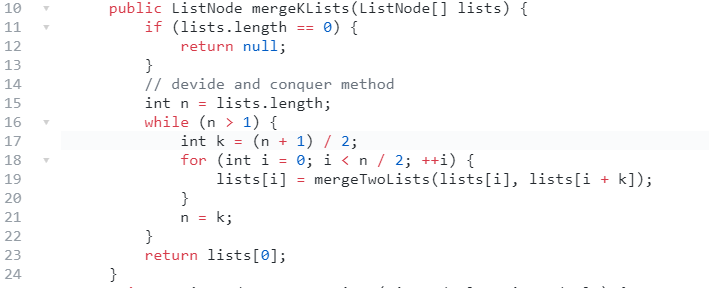
这题虽然是hard难度，但因为有上一题合并两条链表的启下，就很简单了：





然而运行时间很长…

PDF的方法是 divide and conquer 的方法，首先两两合并，然后再将先前合 并的继续两两合并。时间复杂度为O(NlgN)。



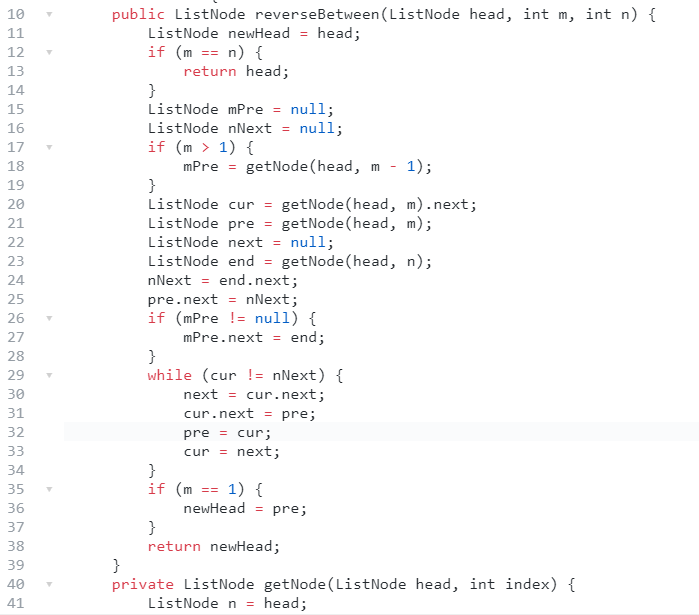


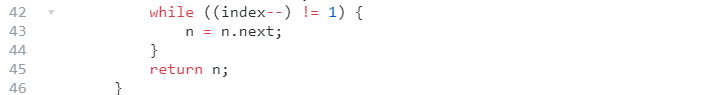
分成两半，一半跟一半合并，再减少长度，继续分半合成。

**92. Reverse Linked List II(反转部分链表)**

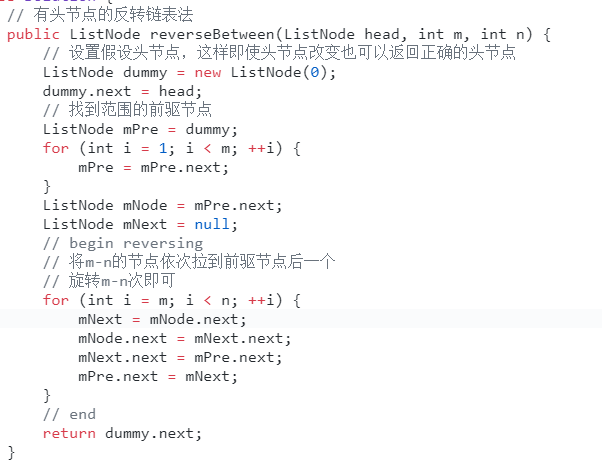
书上有原题，注意的是要记录范围的第一个节点的前一个节点，保证首尾相连。

这是利用三个指针反转链表的方法，不需要头指针。



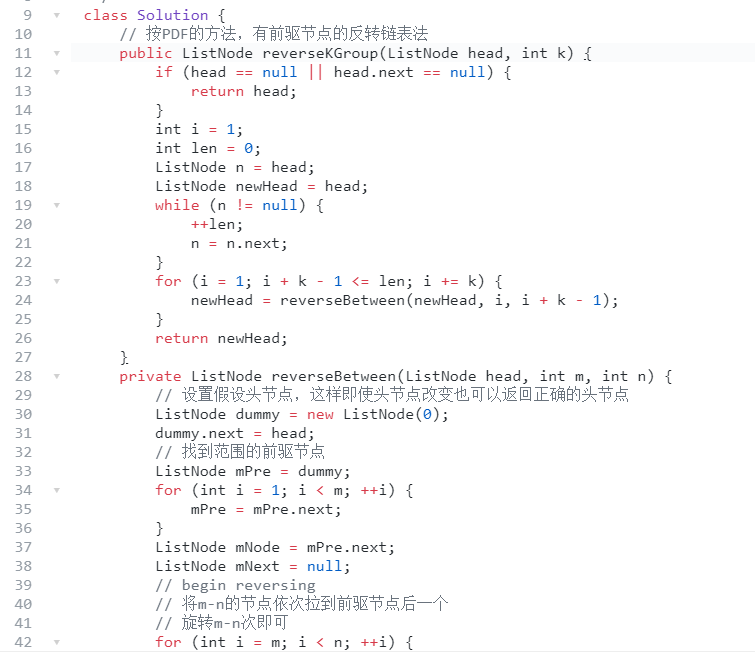


PDF上的思路是需要头指针，两个指针移动，代码如下：

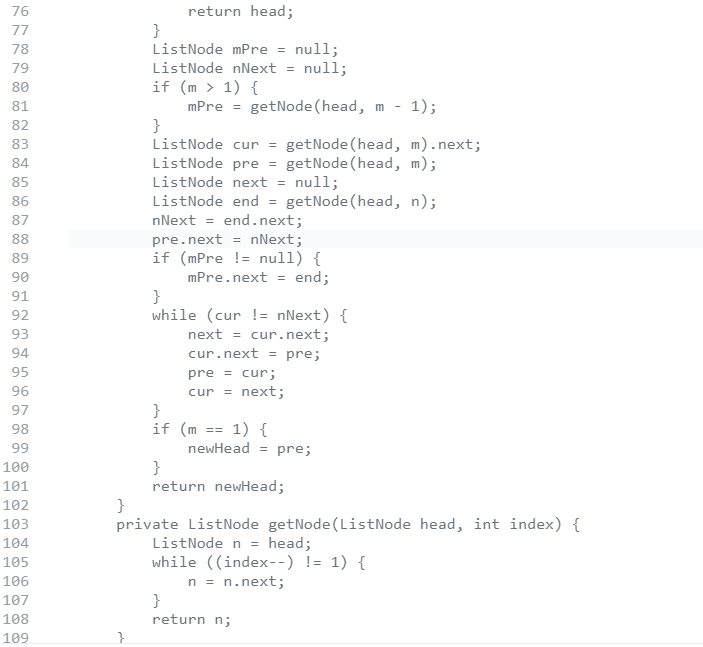


**25. Reverse Nodes in k-Group(从头至尾按k对逆序)**

承接上题对链表部分逆序，两种反转链表的方法，两种解法：







**24. Swap Nodes in Pairs(从头至尾成对逆序链表)**

比较简单，不说明了。

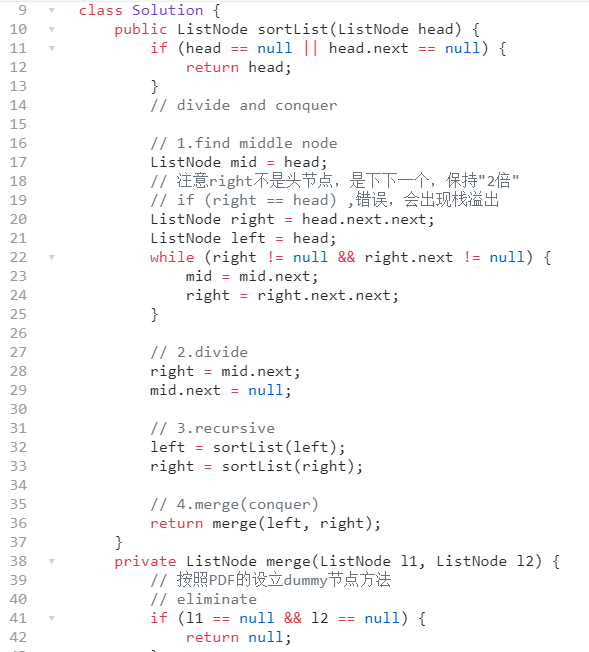
**148. Sort List(排序，要求空间O(1)，时间O(nlgn))**

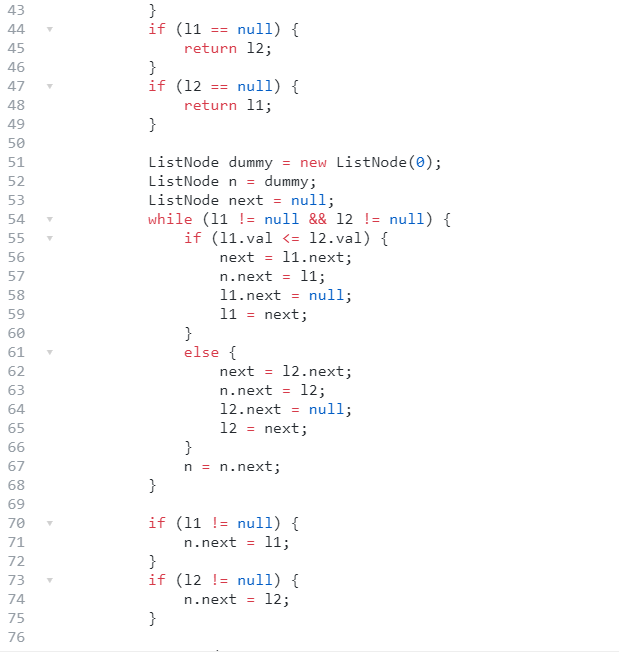
自己的思路：高级排序中归并排序不行，只能要堆排序，但堆排序要建树，如何解决呢？恐怕不行。

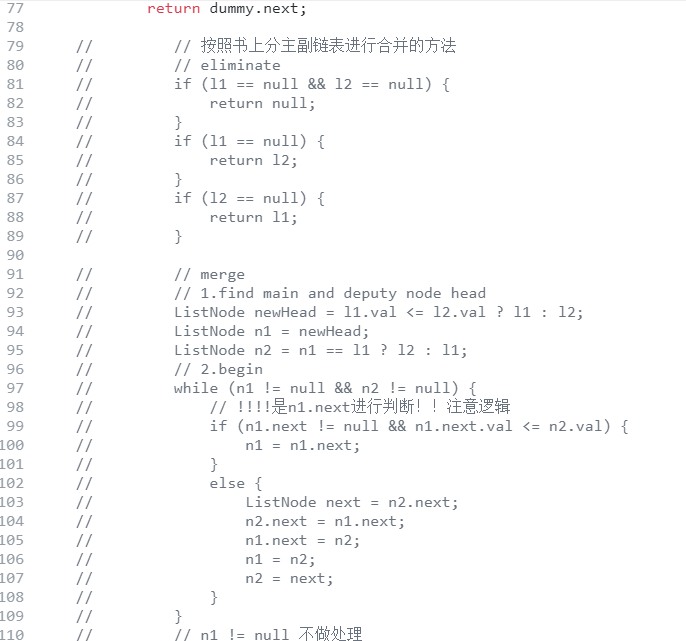
根据PDF的思路：按照divide and conquer(分治法)，将链表一分为二，一分为二…然后将两个链表合并，这样就满足了时间复杂度和空间复杂度。

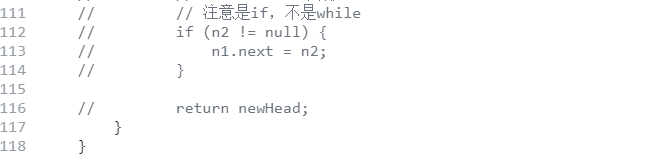
我发现自己对于分治，贪婪，动态规划等算法一点都不了解，需要加强。

此外，合并算法分为书上和PDF的写法，个人觉得PDF的方法更简单明了，但更重要的是自己的逻辑，要慢慢分析。









**61. Rotate List(**rotate the list to the right by k places**)**

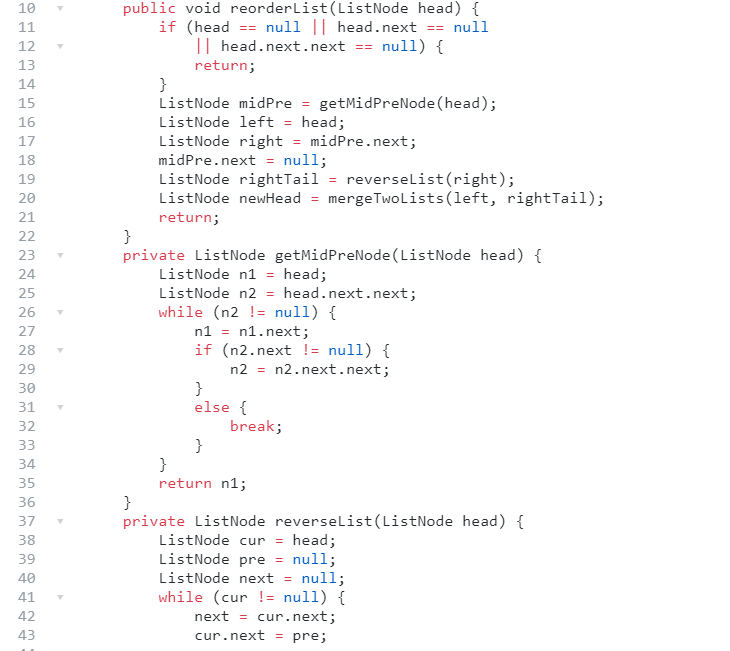
找规律即可。注意代码简洁。

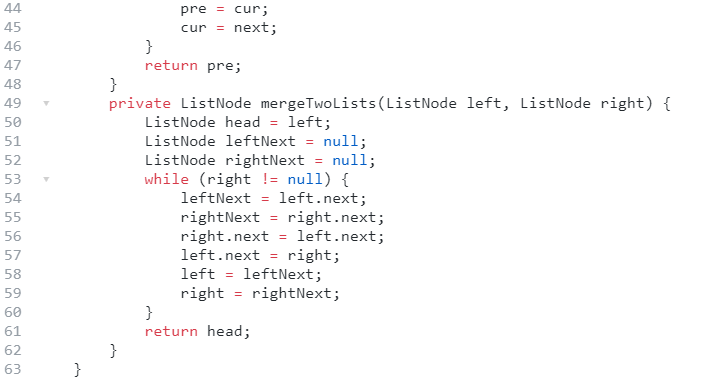
**143. Reorder List(前后半链表部分合并)**

综合题目，有空可以练练。

1. 找到中间节点；
2. 反转后半部分链表
3. 链表合并

自己代码如下：





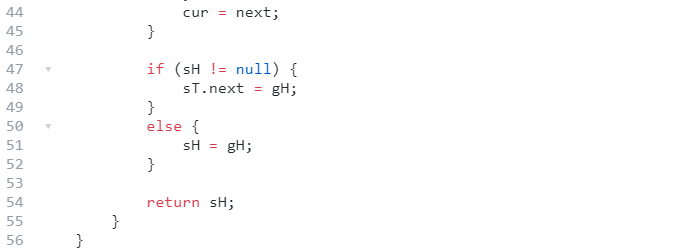
**86. Partition List(切分链表)**

一开始我想用在学校交的切分(快速排序)，然而比较复杂。

PDF：我们可以使用两个链表，p1和p2，以此遍历原链表，如果节点的值小于x，就挂载到p1下面，反之则放到 p2下面，最后将p2挂载到p1下面就成了。

总之，要开拓思维，也不要忘记做过的题目的解法。





注意最后两个链表合起来的时候要判断第一个链表为不为空。

**445. Add Two Numbers II**

书上有原题，思路是利用栈结构将链表逆序相加(栈的使用)