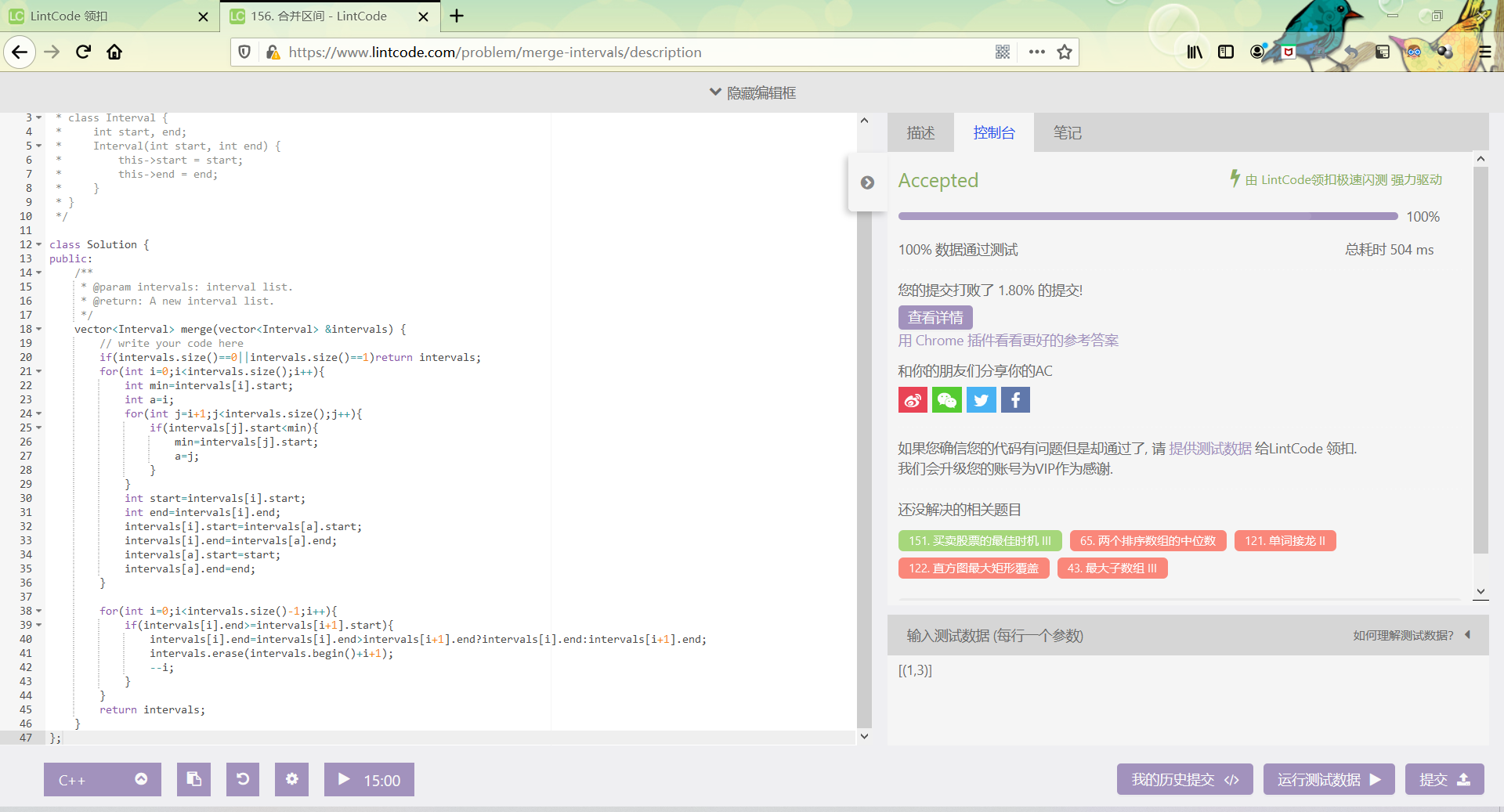
数据结构暑假作业报告

20191001985 11J191 王浩

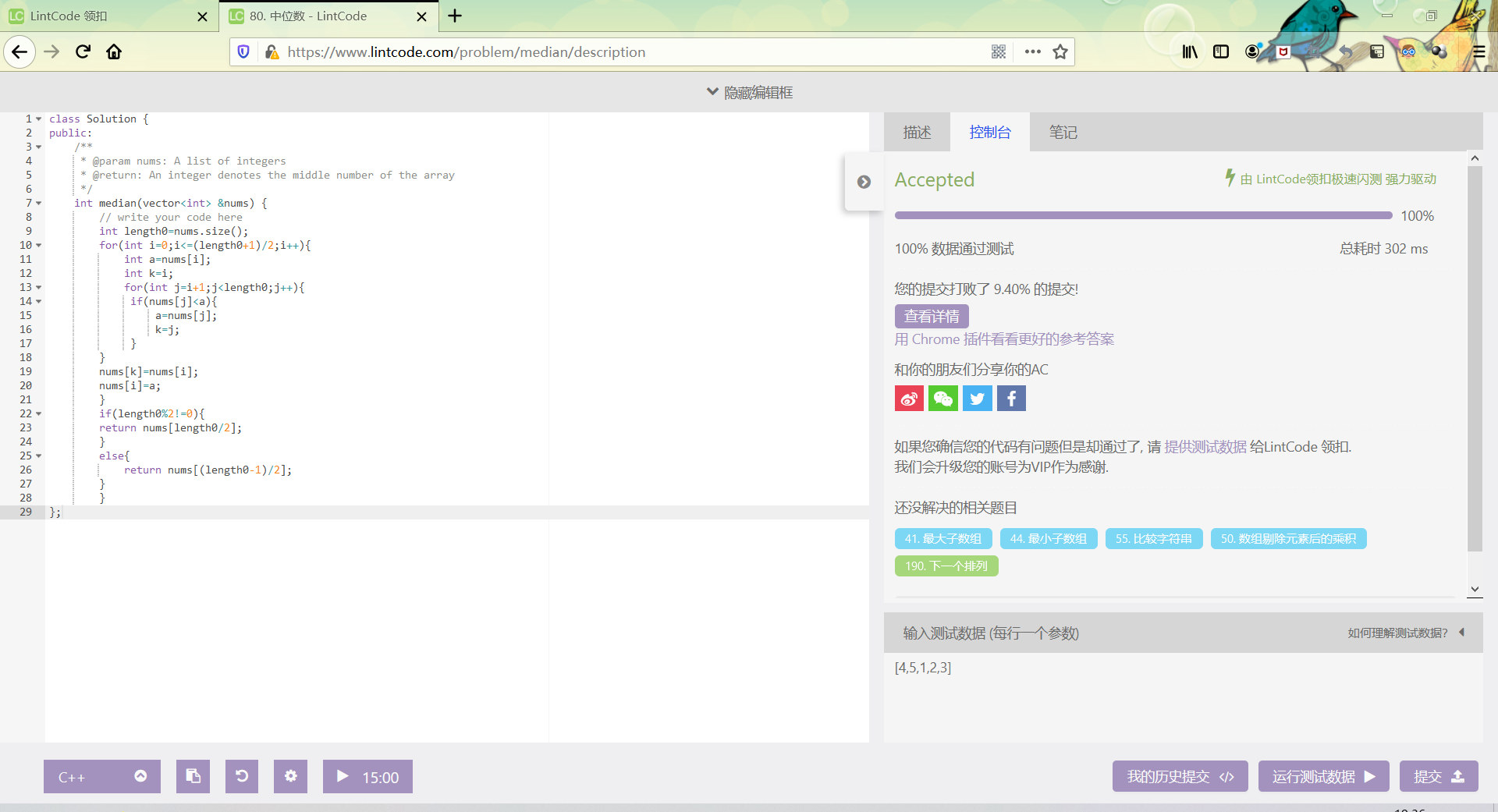
**数组**

1

**合并区间**

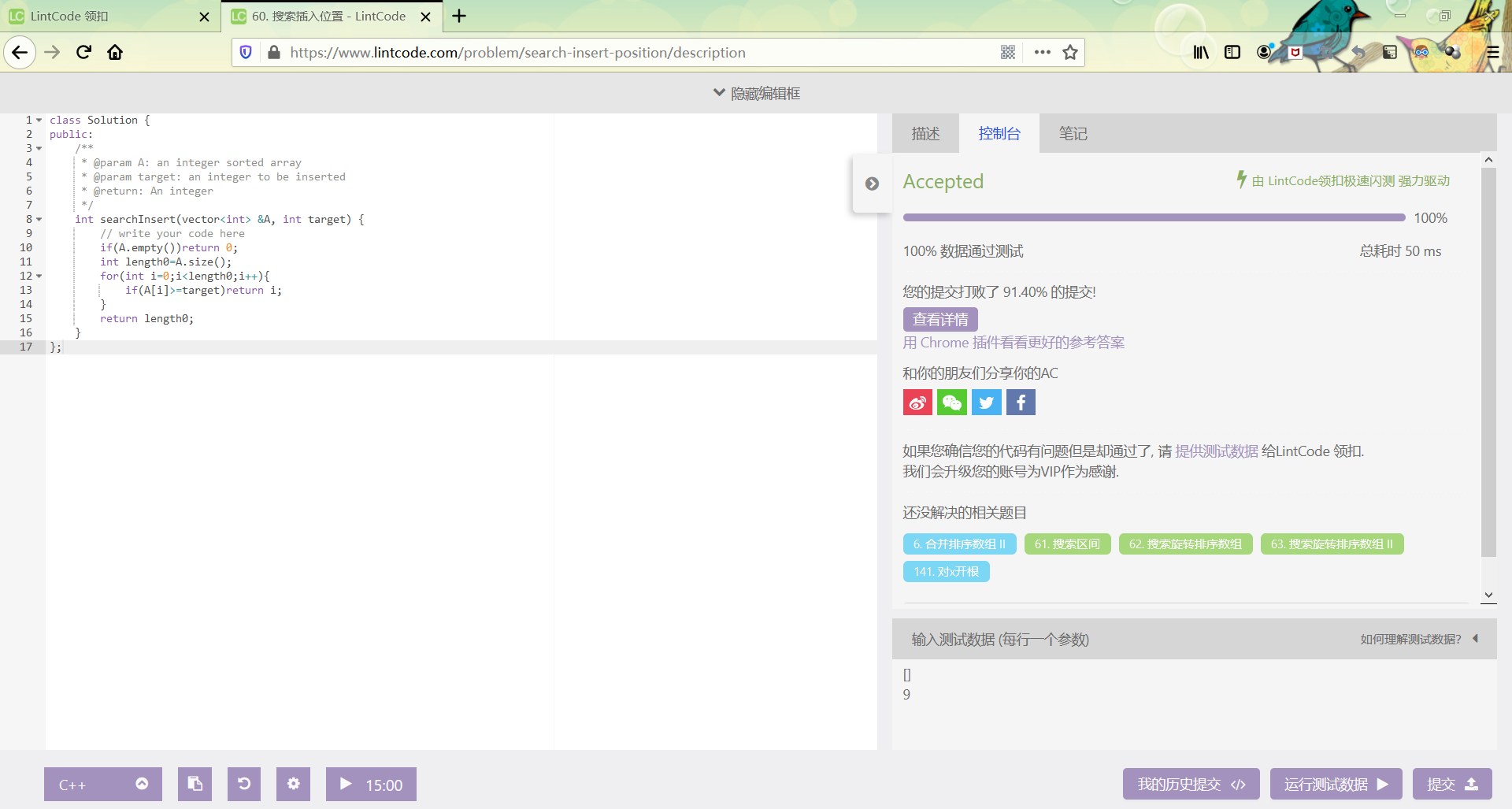
实现思路：对每个区间按照start由大到小排序，再通过一个循环对每次循环对应的前后量的区间进行比较，若能合并，则删除后面的区间，前面的区间改变为合并之后的区间。但是这样的运行时间较长。

**2、中位数**



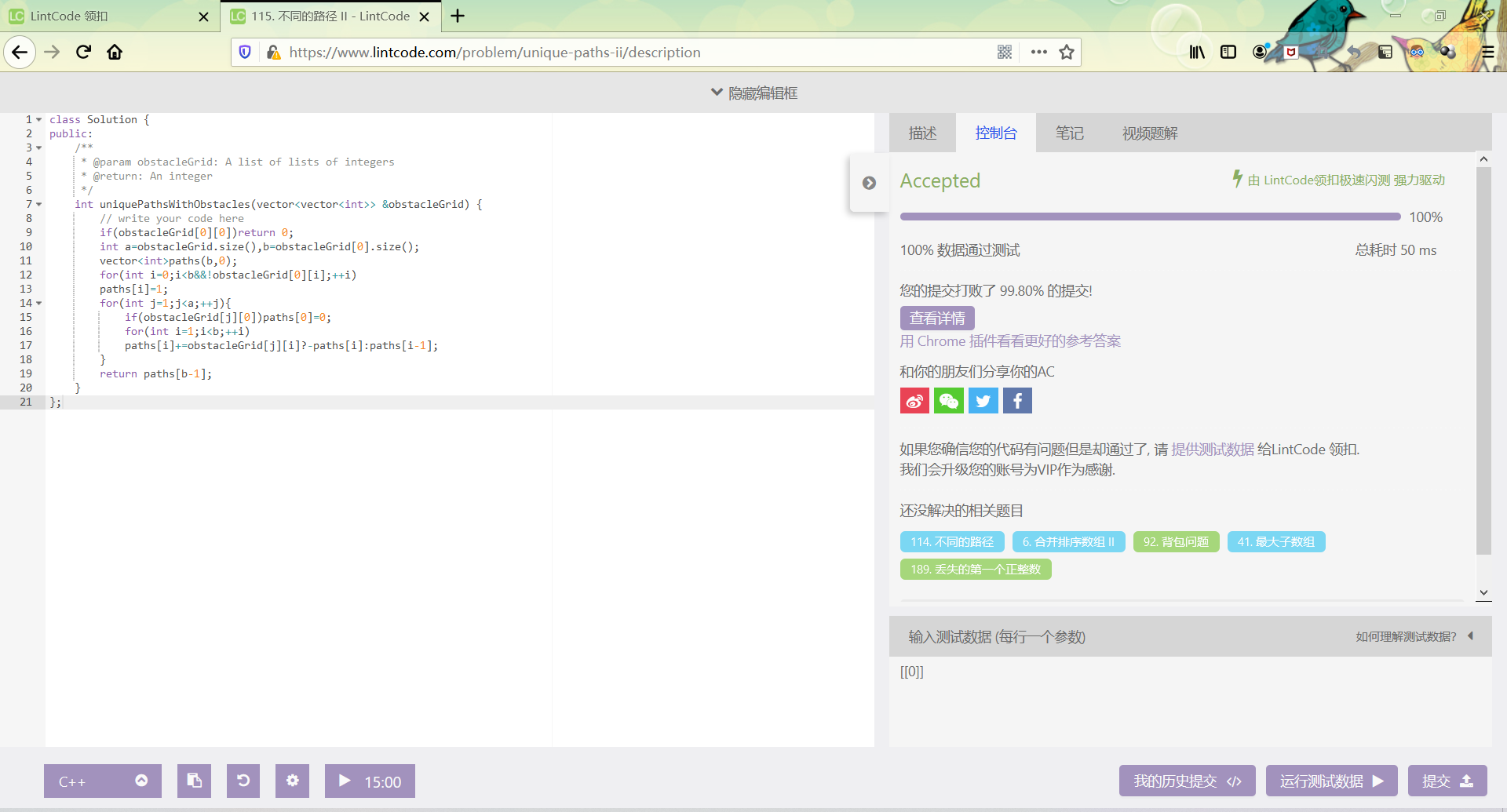
实现思路：对前一半的数进行排序，这里用的是插入排序，每次循环找出最小的，然后与前面的位置进行交换，当前一半的数据全部排序完后，再根据数组长度是奇数还是偶数返回值

**3、搜索插入位置**

****

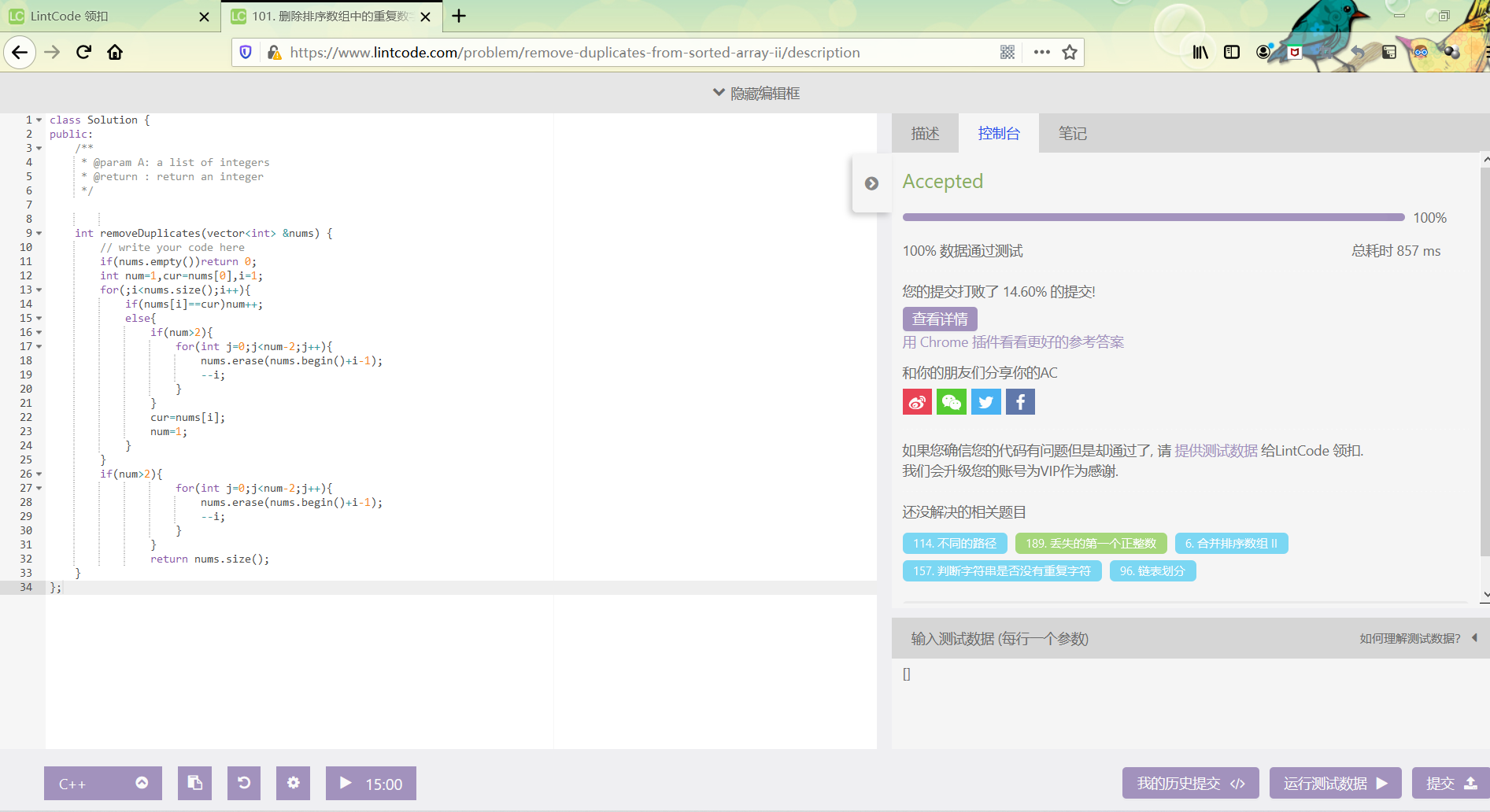
思路：因为是排序数组，则进行一次循环，直到数组中的数大于target，则直接插入。若数组长度为0，则直接返回0.若没有大于target的值，则插入到最后。

**4、不同的路径2**



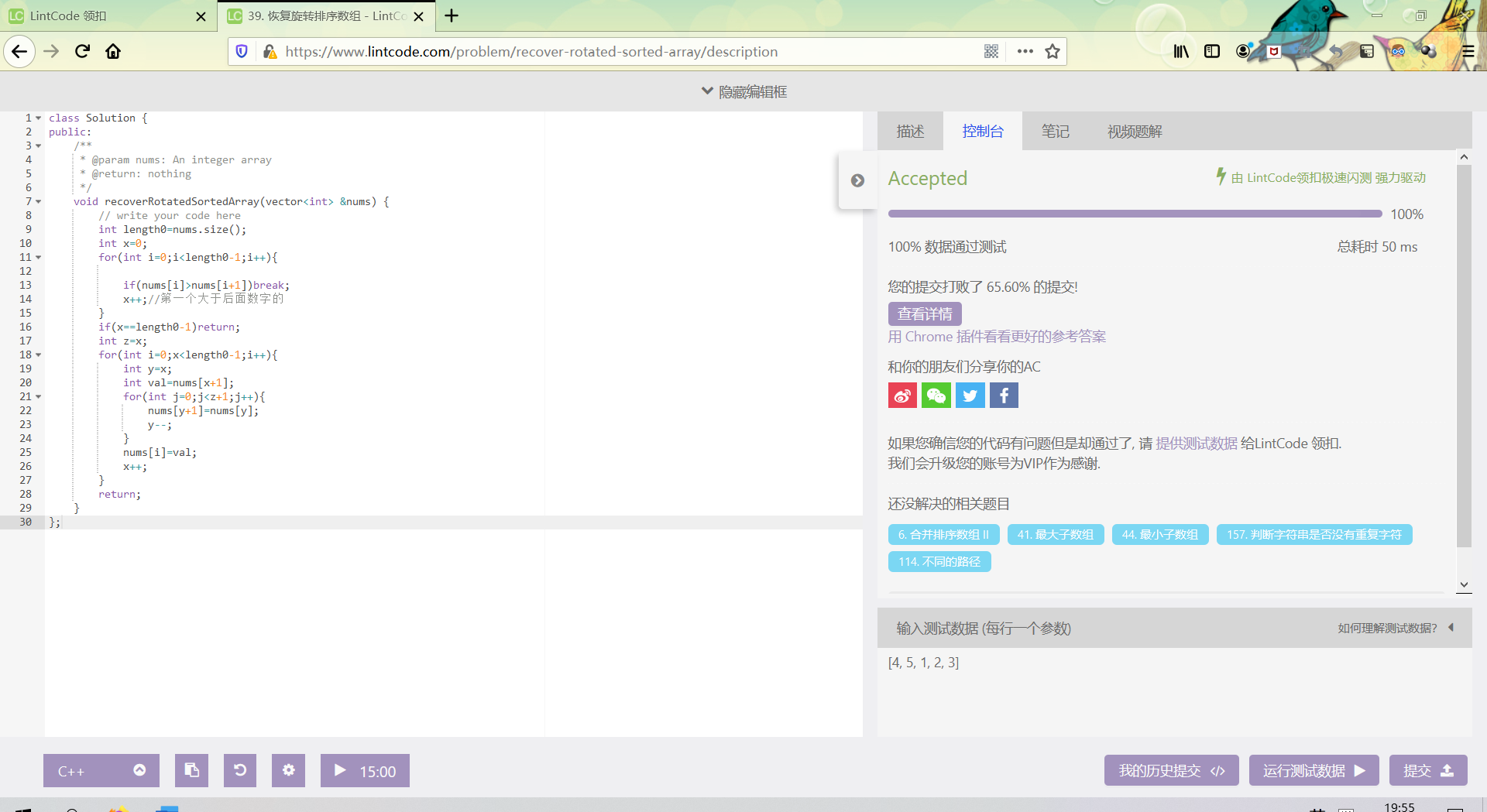
思路：这道题对我来说比较有难度，用自己的方法总是超时。现在用的是九章算法中的答案，但是具体怎样实现没看懂。

**4、删除排序数组中的重复数字2**



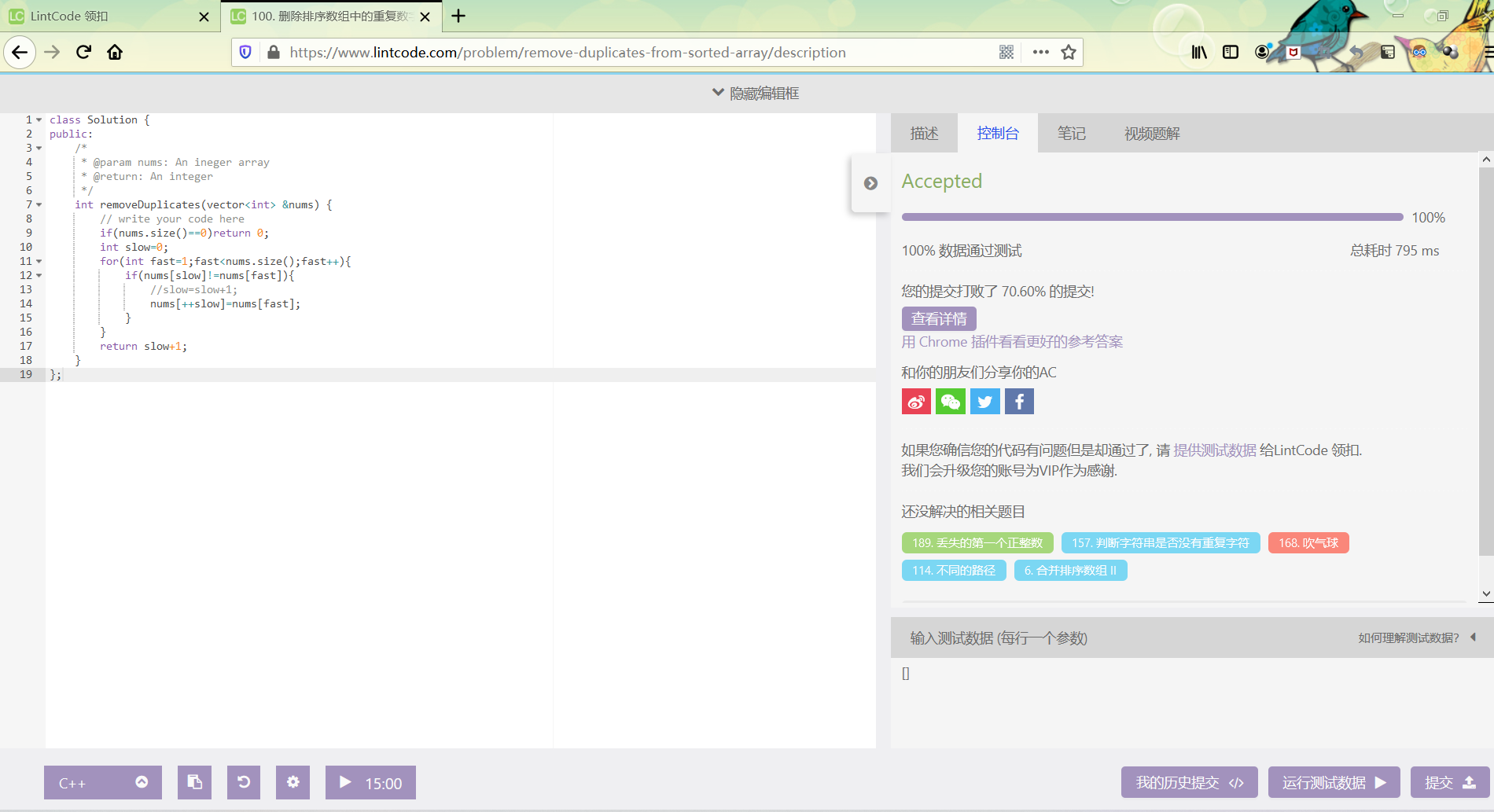
思路：若数组长的为0则直接返回该数组。若不为0，则用cur保存当前重复的数字，用num保存重复数字的个数，用一次循环，若num>2则删除大于2的部分，在这里我用的是erase()函数，开始我用的是自己的函数，通过用一个循环将后面的数组元素整体前移来实现删除，但是时间过长。

**5、恢复旋转排序数组**

****

思路：先通过一次循环找到旋转排序的起点，再通过一次循环将起点及之后的数字整体前移。具体实现过程是每次将前面的数据整体后移一个单位，从第一个数组空间开始空出来前面的一个数组空间供后面的元素插入。

**6、删除排序数组中的重复数字**

****

思路：使用快慢指针，在一次循环中，若快指针不等于慢指针所指的值，则用快指针所指的值覆盖慢指针下一个值，这样当慢指针所指的重复值全部被后面的值覆盖后，覆盖前面的值会在接下来的时间里 也被覆盖。

**7、最长连续上升子序列**

****

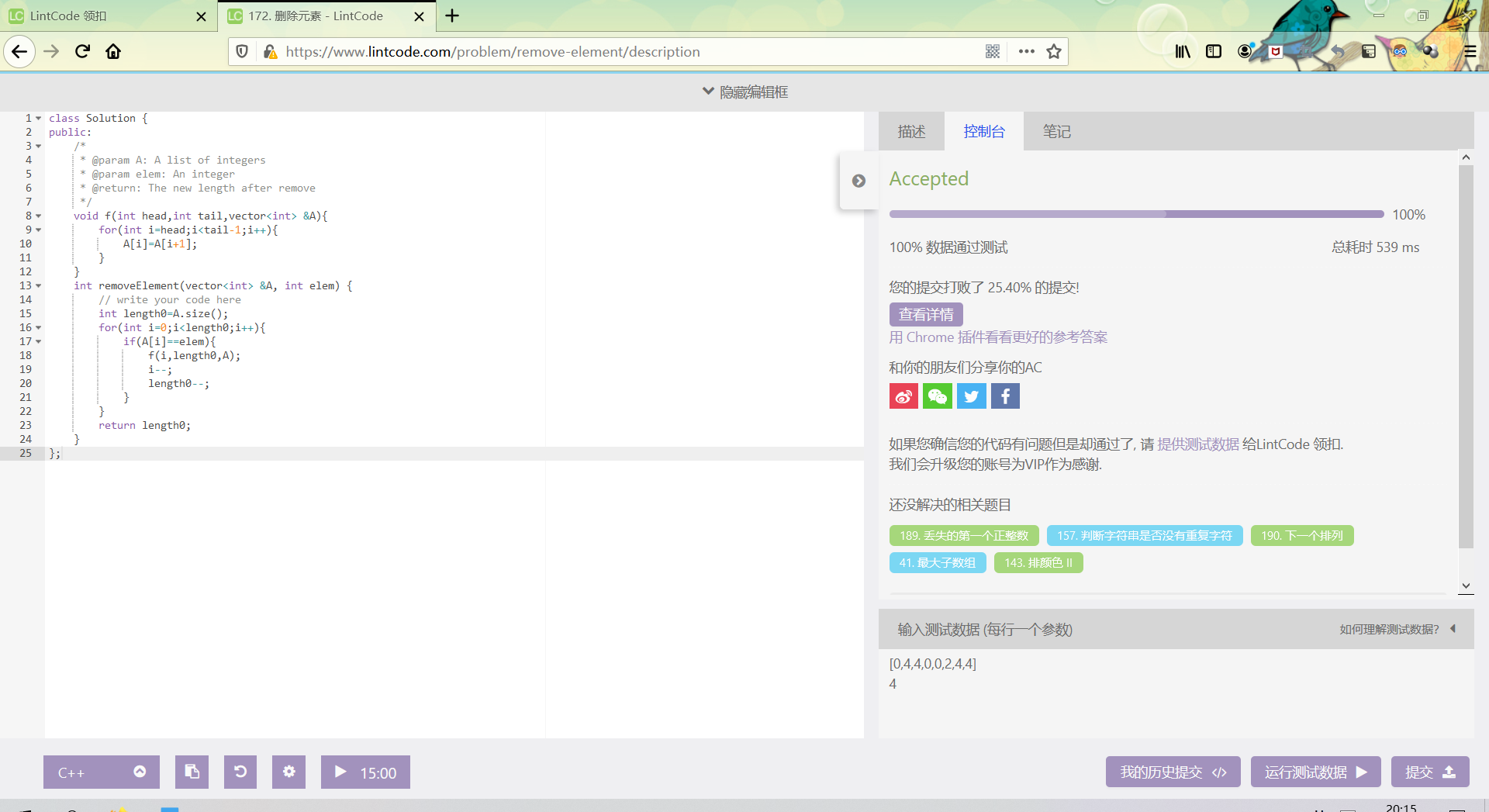
思路：用rise和fall两个值分别储存最大上升子序列和下降子序列的长度，这样可以在一次循环中完成。开始时我只用一个值储存长度，这样则需要大于一次循环和其它不必要的赋值、比较，实现速度垫底。

**8、第三大的数**

****

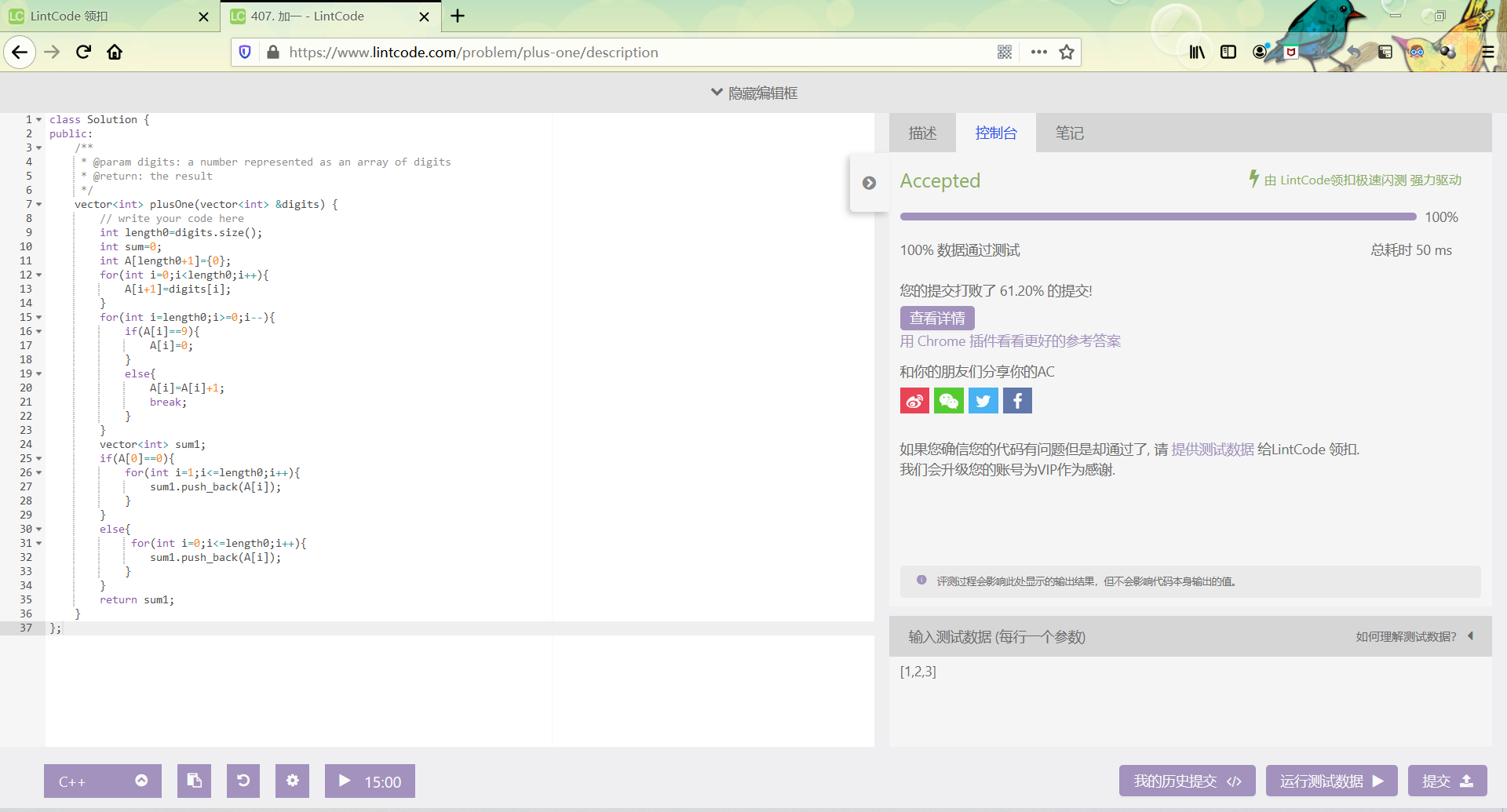
**思路：**用三个变量来存放1、2、3大的元素，遍历数组，找到第三大的数，没有则返回第一大的数，忽略重复的数。

**9、删除元素**

****

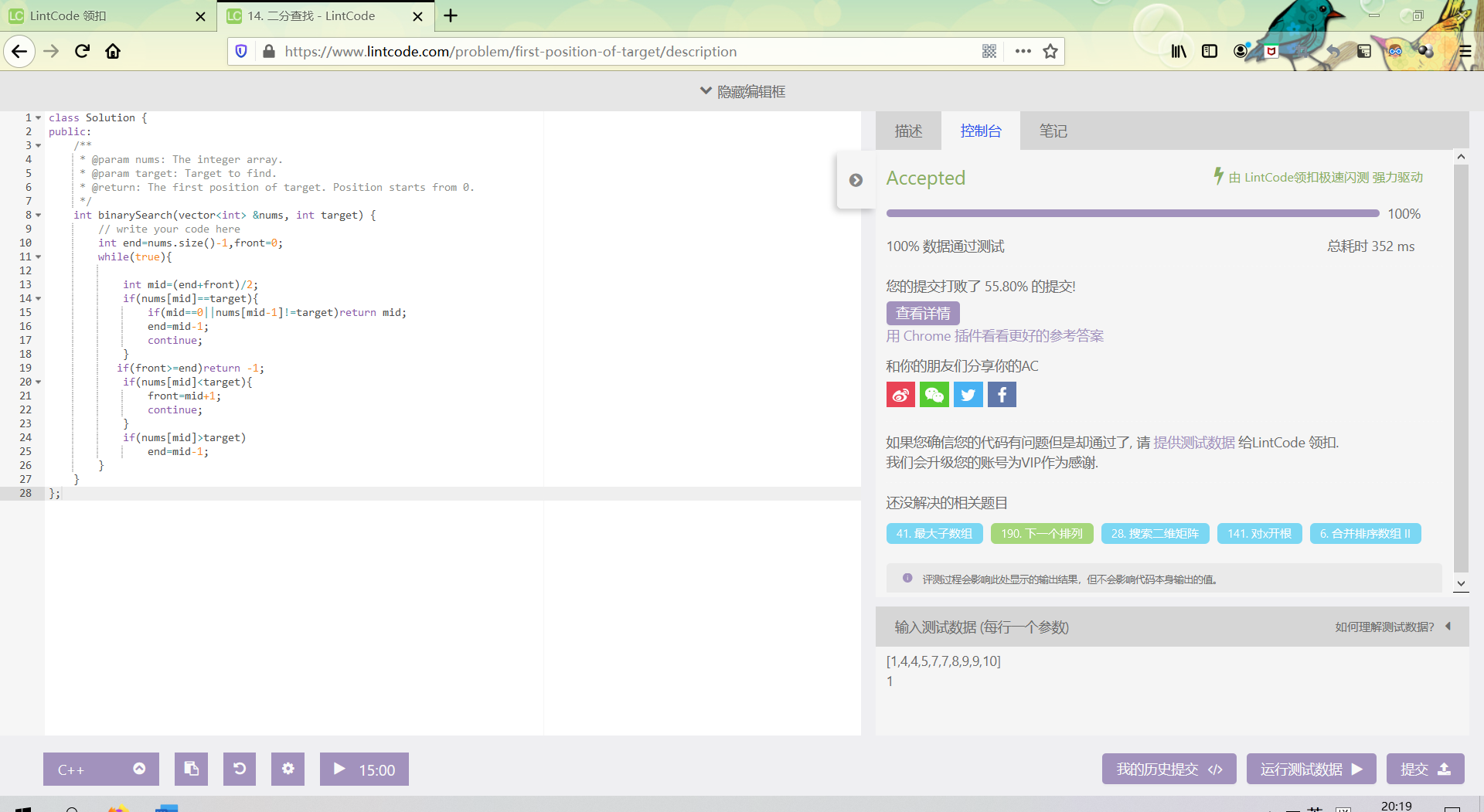
思路：一次遍历，找到target，其后的元素整体前移一个长度即可，具体实现过程是每次将前面的数据整体后移一个单位，从第一个数组空间开始空出来前面的一个数组空间供后面的元素插入。

**10、加一**



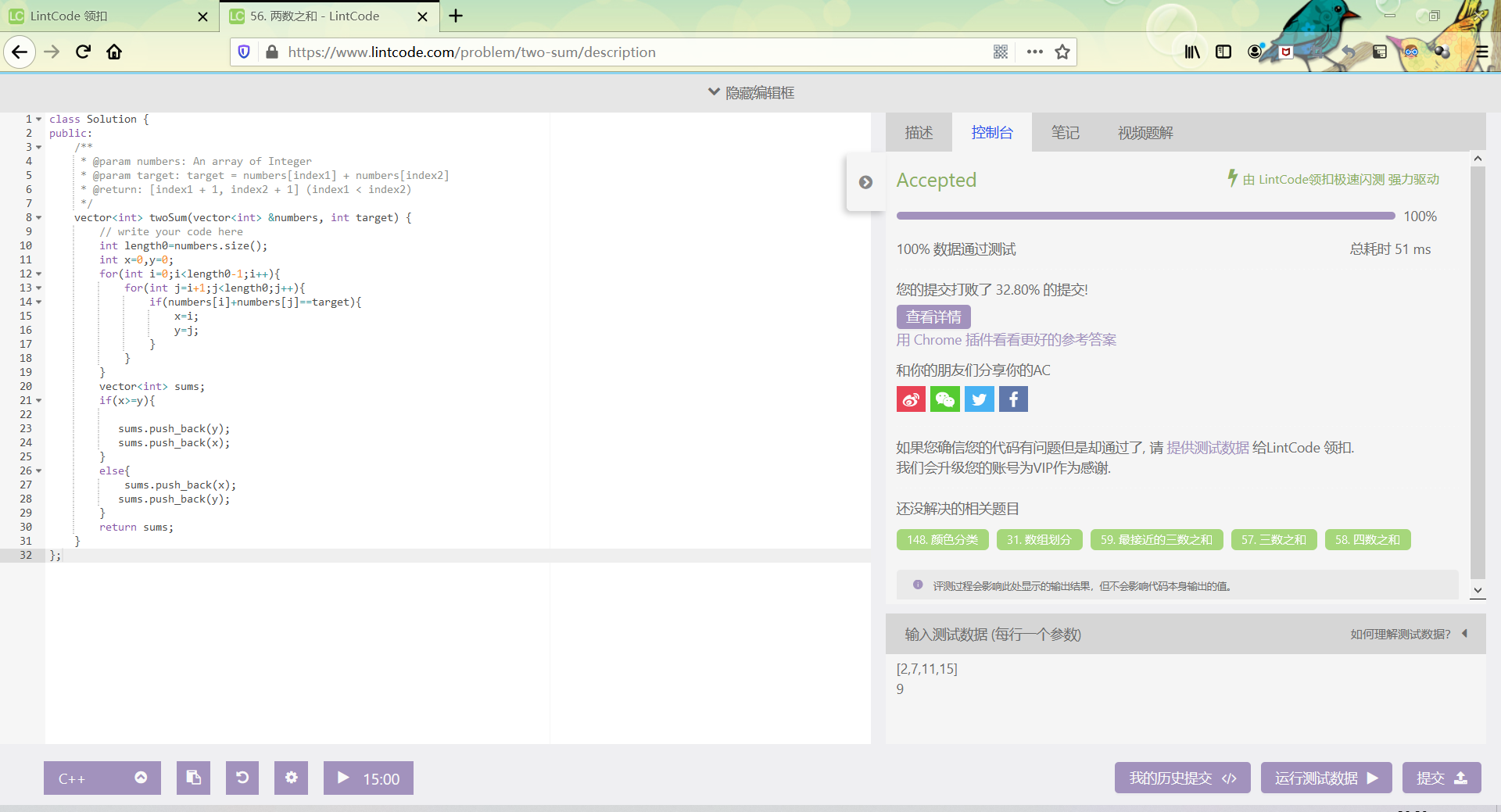
思路：对每个数组元素加一，若等于10则赋值为0，下一个元素进位1。需要用到一个循环，将所有节点相加。当其中一个链表结束了之后需要跳出循环，重开一个循环对另一个链表操作。

**11、二分查找**

****

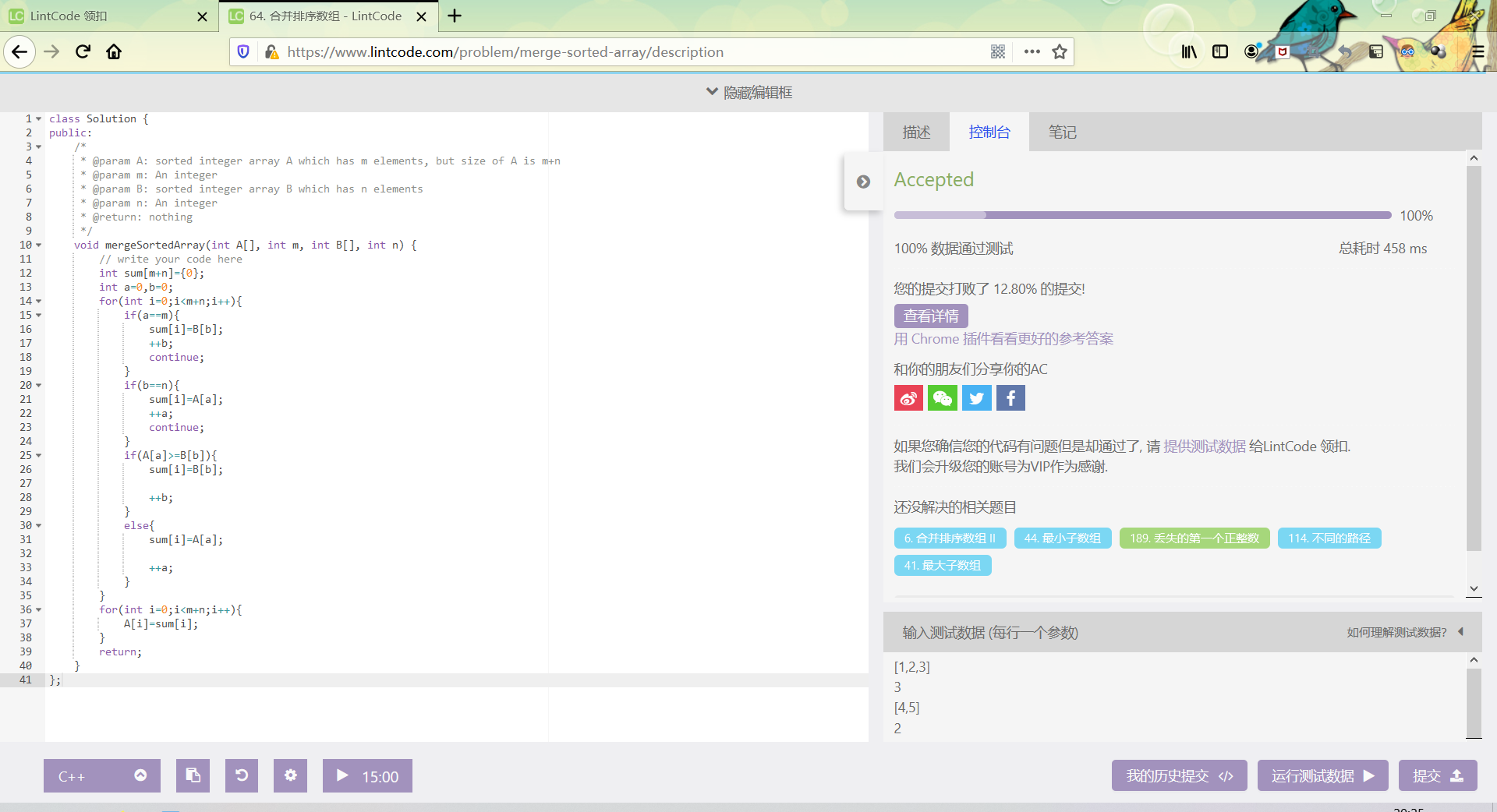
思路：因为是排序数组，则比较target与中位数的大小，大于则继续依此法搜索前半部分，小于则搜索后半部分，依次进行，最后找到target。

**12、两数之和**

****

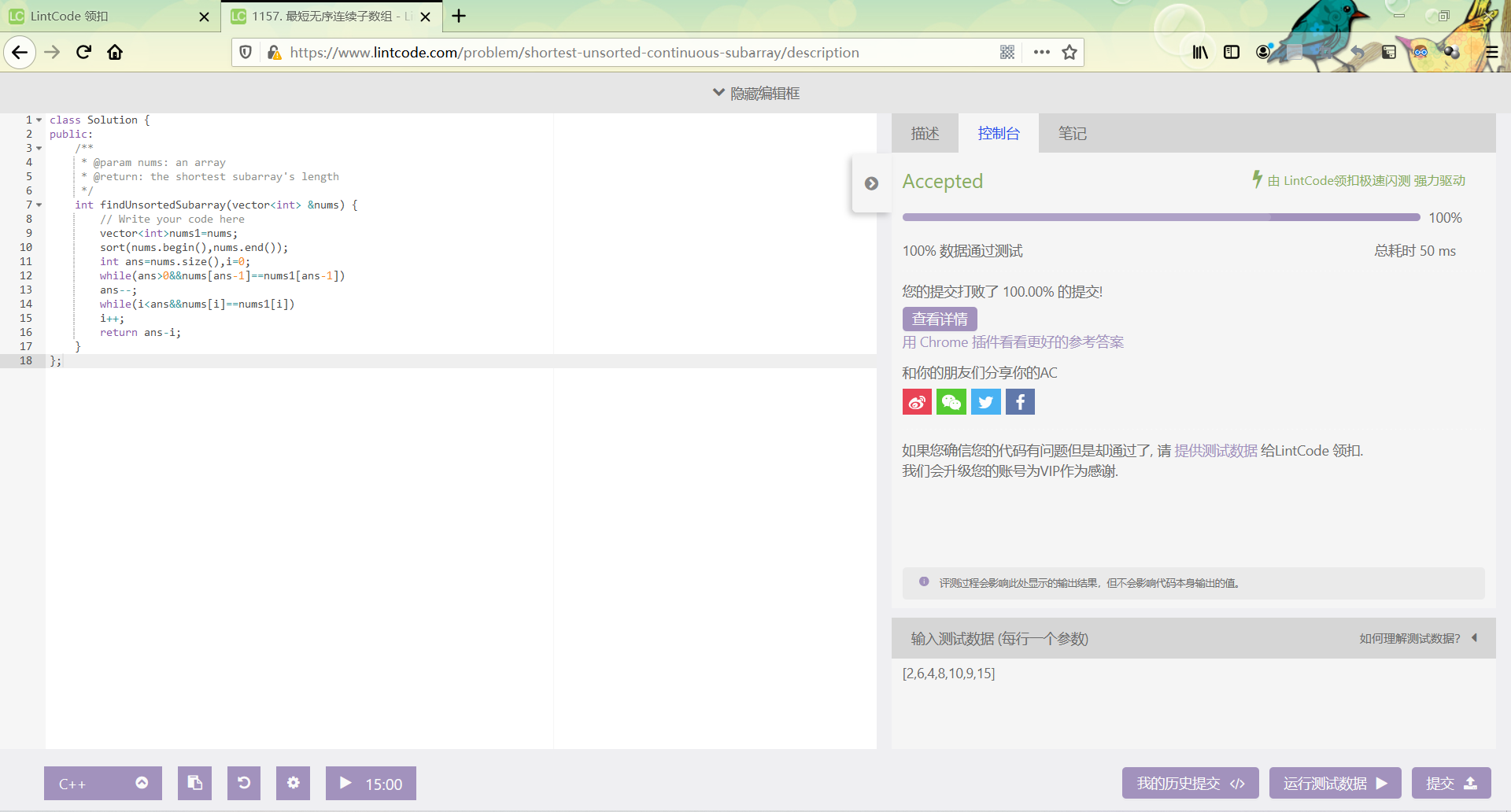
思路：只需要通过两个嵌套的循环找到符合条件的数，再由小到大输出即可。里层循环的起点应该是外层循环当前循环所在的下一个，以缩短运行时间并防止重复。

**13、合并排序数组**

****

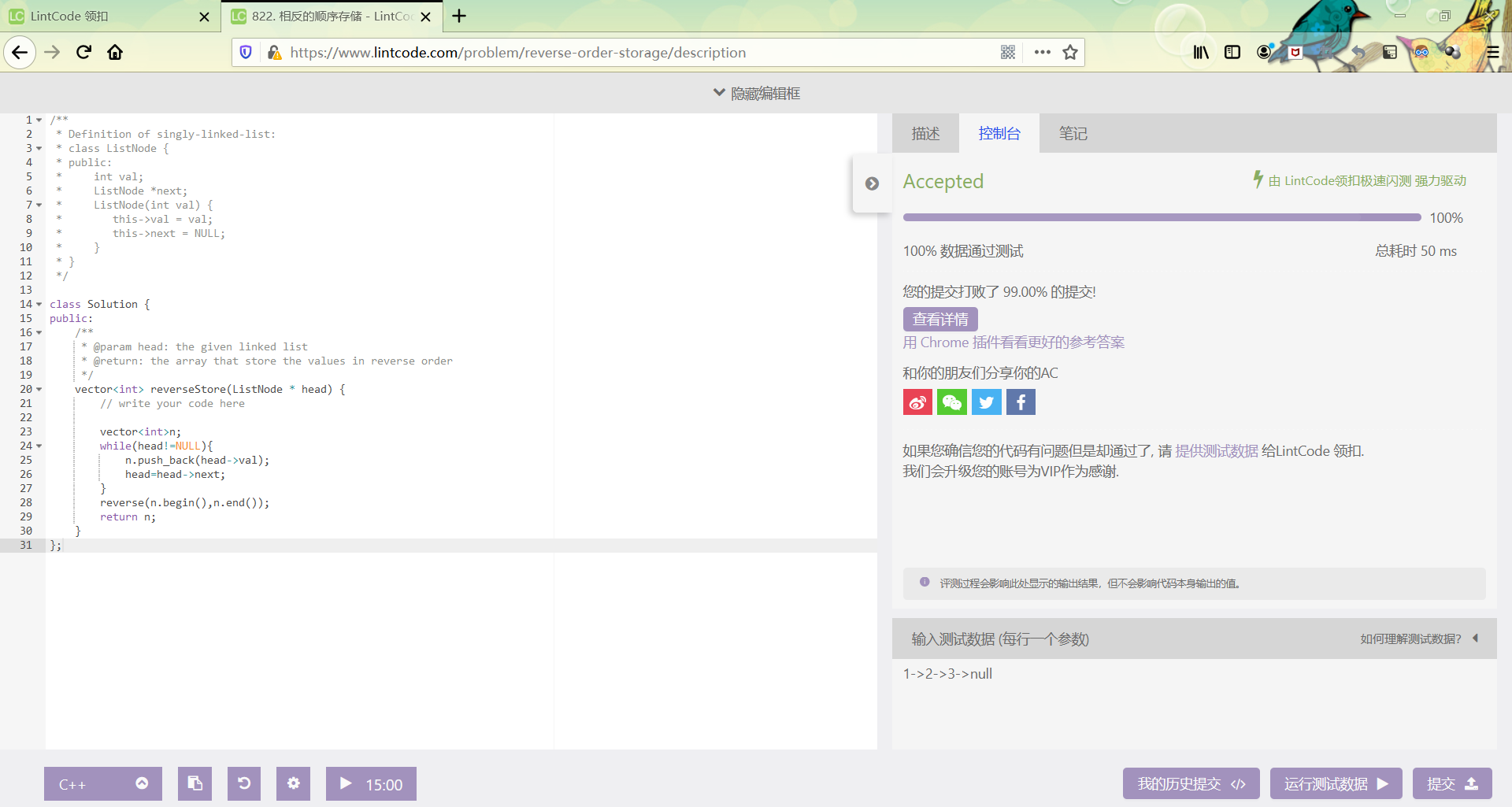
思路：新建一个数组，分别从两个数组中拿出数来比较，将较小的数放在前面的位置，一次放入新建的数组，一个数组空后，将另一个数组整体放入即可。再对A数组赋值。

**14、最短连续无序子数组**

****

思路：新建一个和原数组一样的数组，对这个数组进行排序，这里用的是sort函数。再从头、尾开始遍历，直到找到和原数组数不一样的数，得到无序子序列的头尾下标。

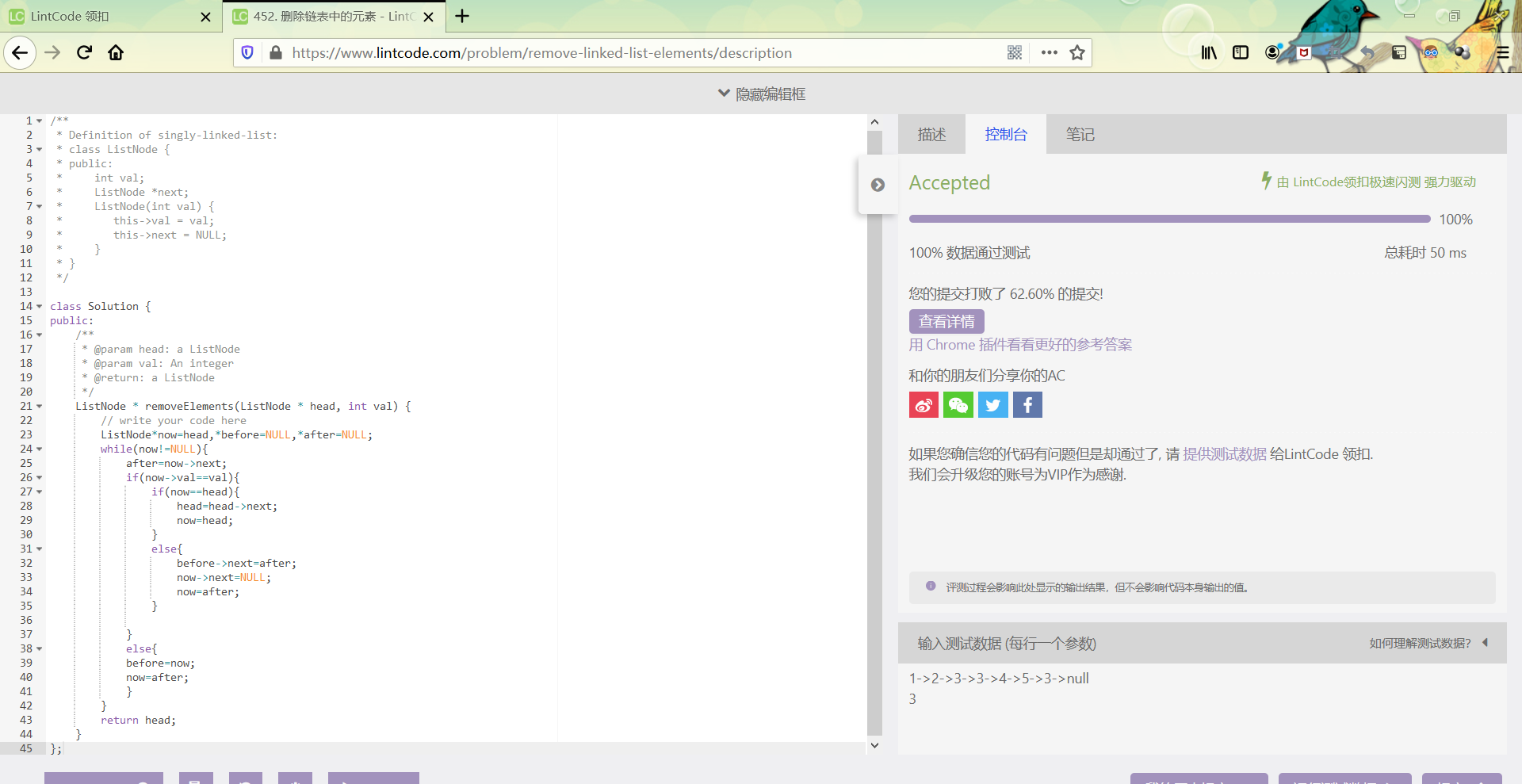
**15、相反的顺序存储**

****

思路：建立一个vector数组，按照相反的顺序push\_back，再翻转。

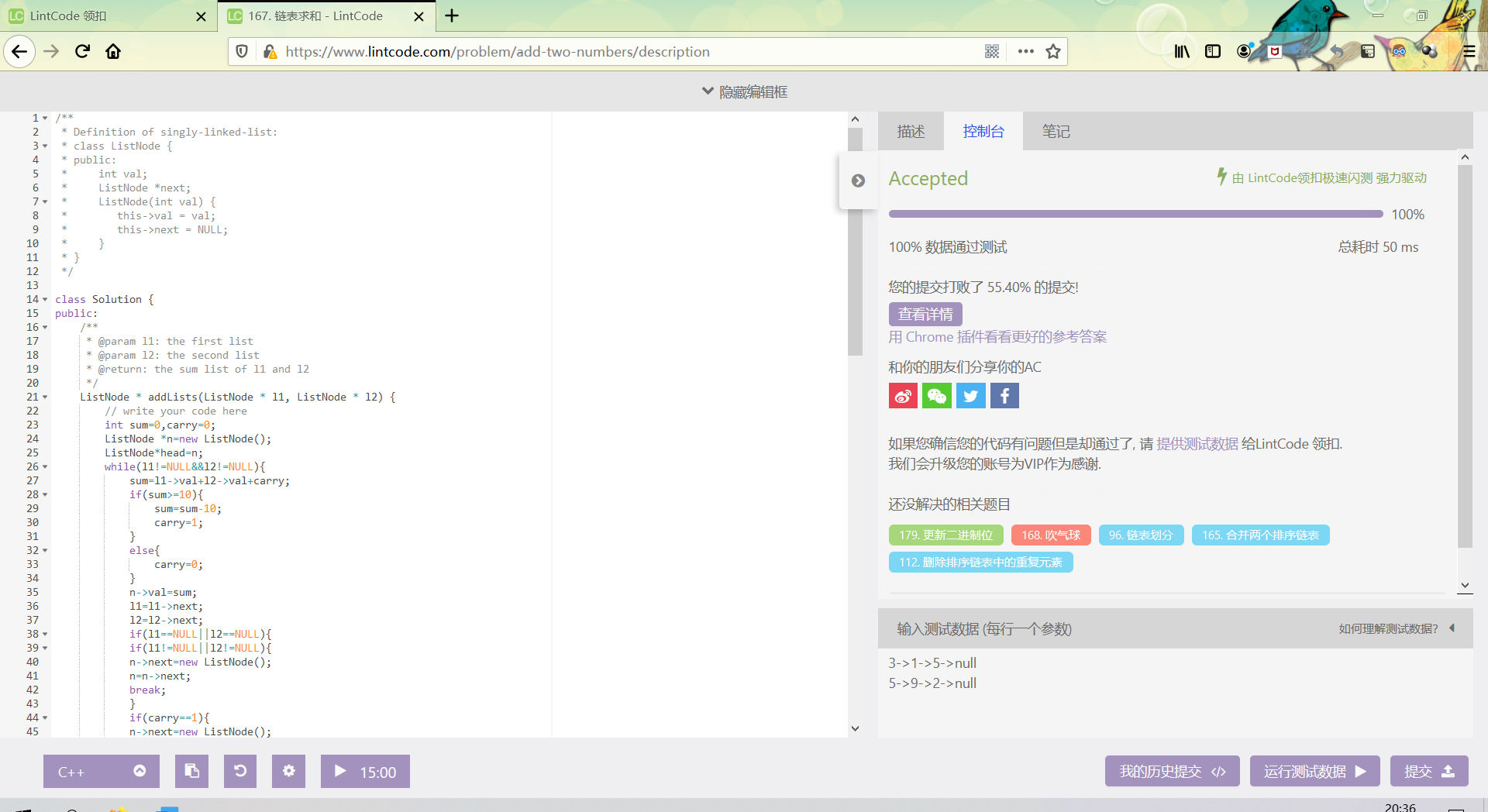
**链表**

**16、删除链表中的元素**

****

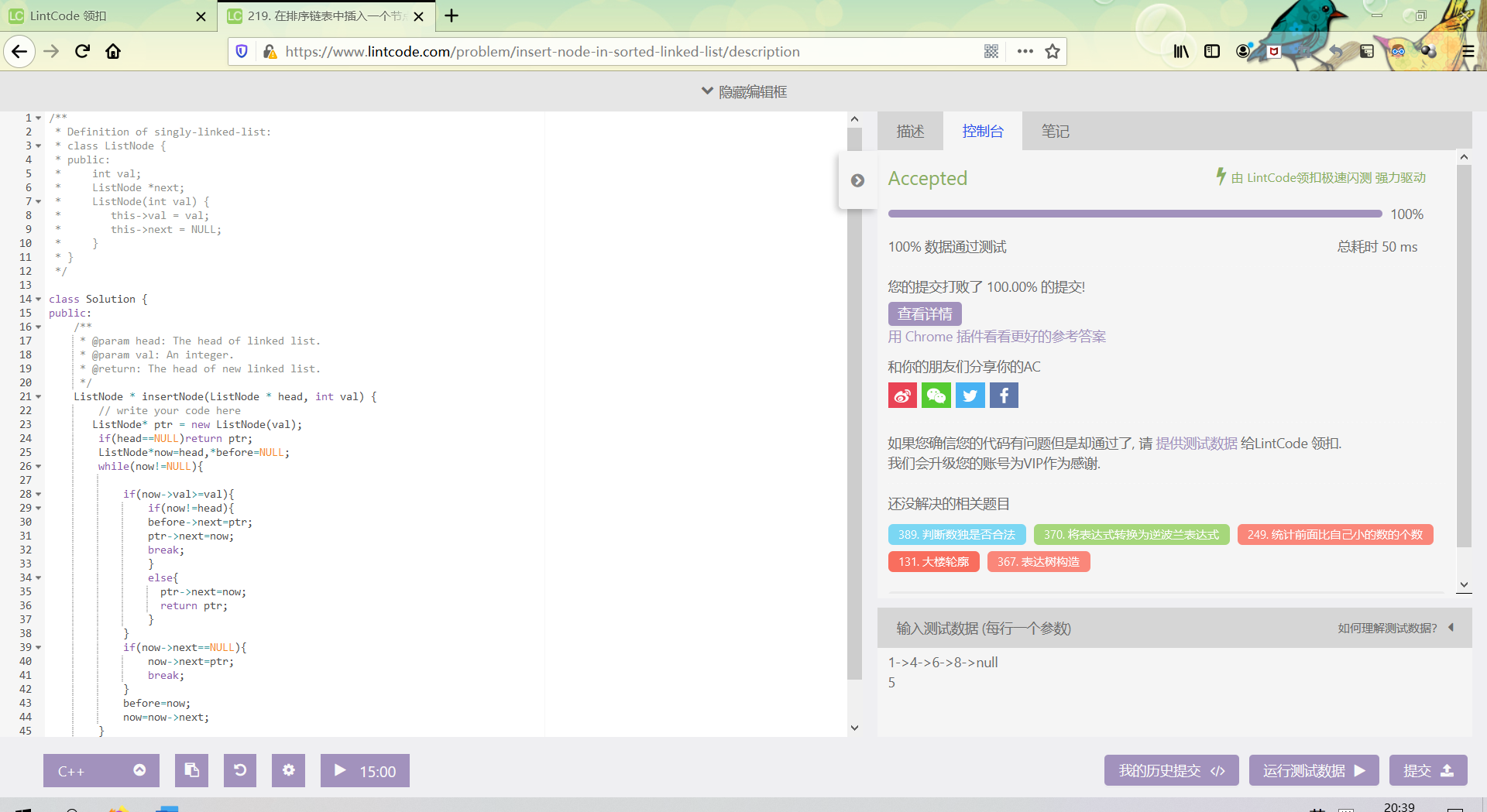
思路：用三个指针before,now,after指向前中后三个节点，判断中节点，若删除则将before的next指向after，这样实现删除操作，不删除则将三个指针整体后移一次，注意，before一开始指向null。也可以用哑节点。

**17、链表求和**

****

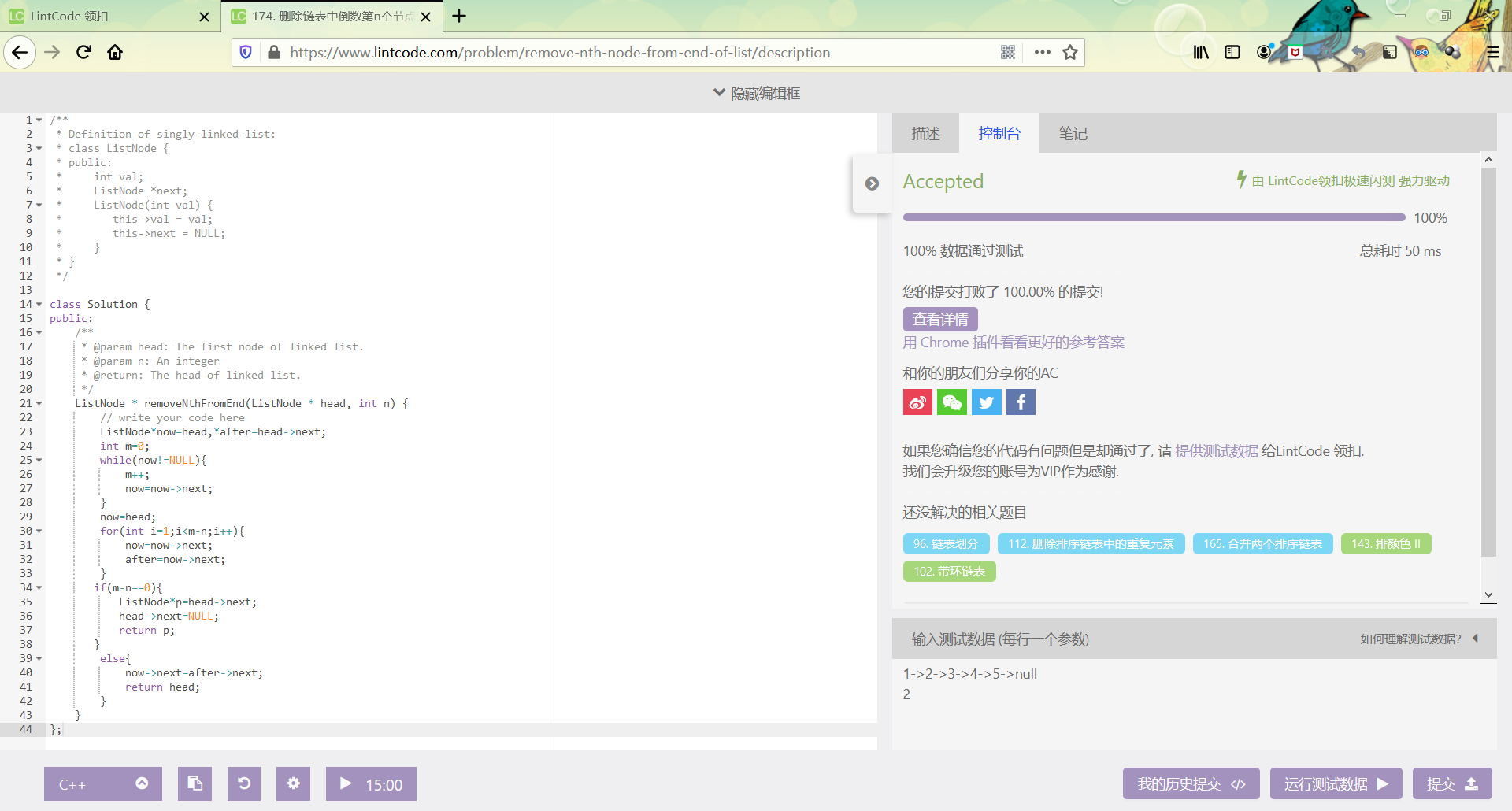
思路：新建一个链表，然后从头节点开始对两数进行相加，用carry表示进位，相加后的值从头结点开始储存在新链表中，这样在一次循环中即可完成。

**18、在排序链表中插入一个节点**

****

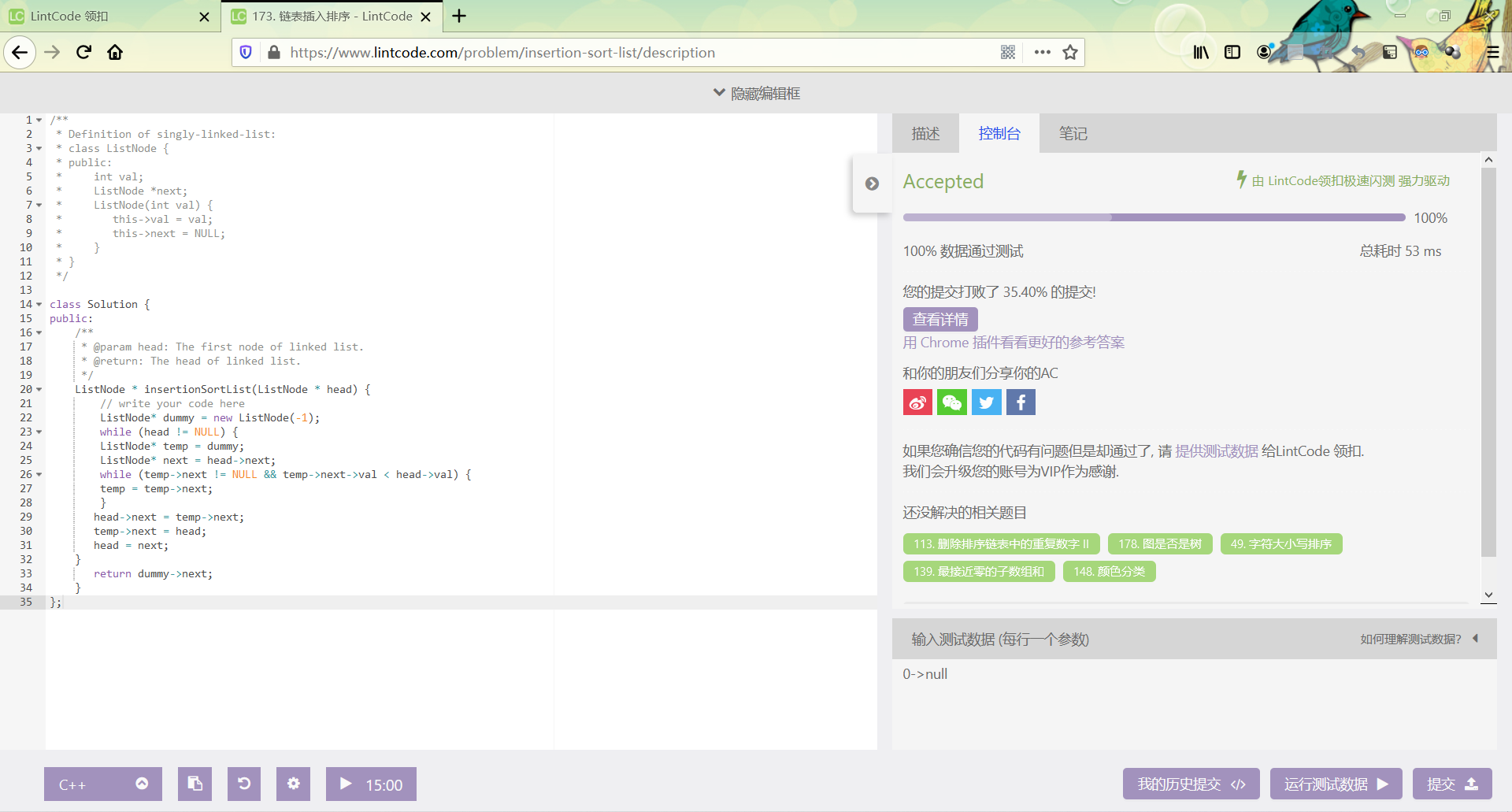
**思路：**用一次循环找到符合条件的节点，用before和now分别指向前中两个节点，直接改变节点指向即可。

**19、删除倒数第n个节点**

****

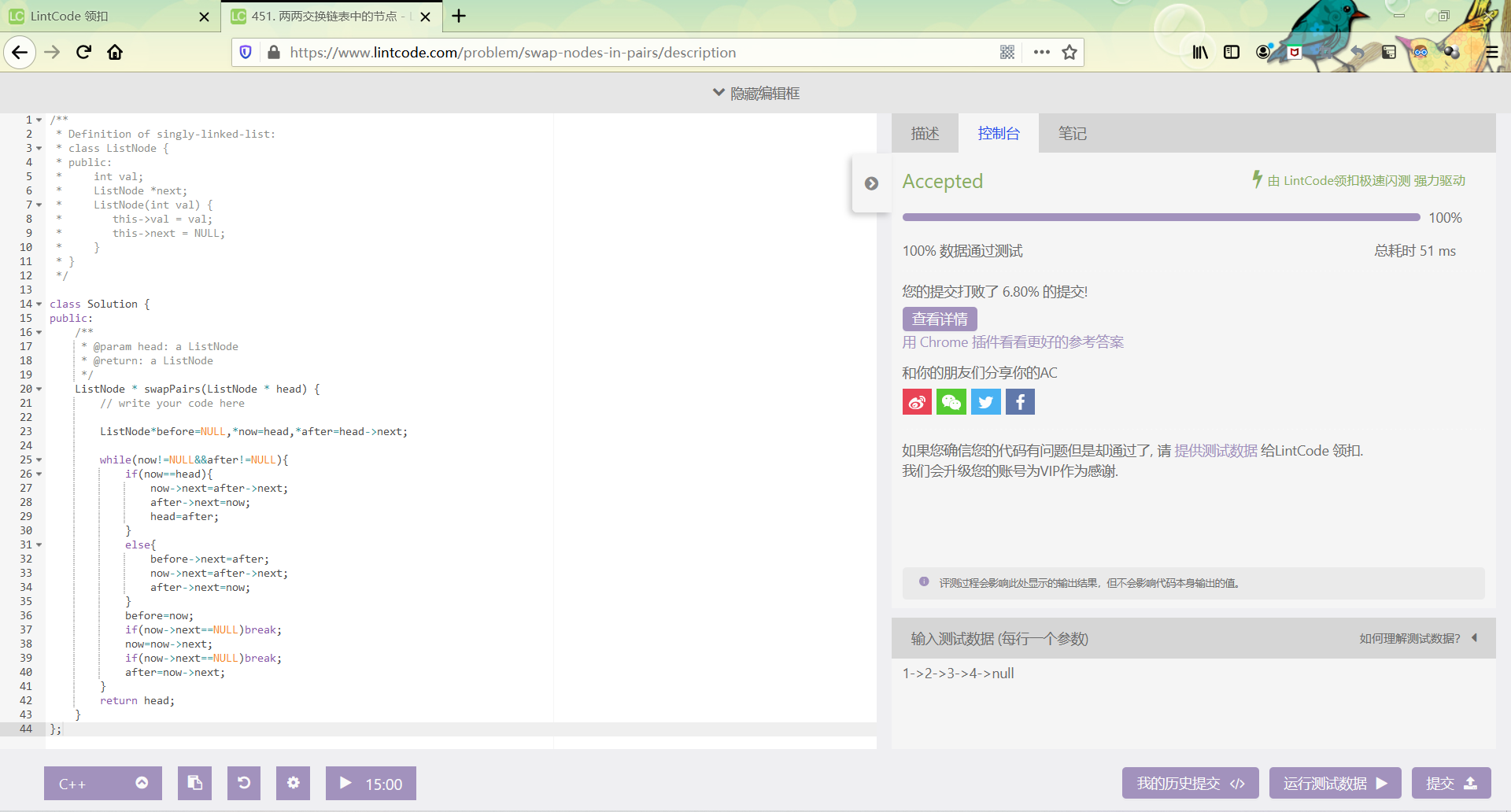
思路：一次循环得出链表长度，再进行一次循环，找到倒数第n个节点，删除即可。也可以使用快慢指针，快指针位于慢指针之后的第n个位置，当快指针指向最后一个节点时，慢指针指向倒数第n个节点。

**20、链表插入排序**

****

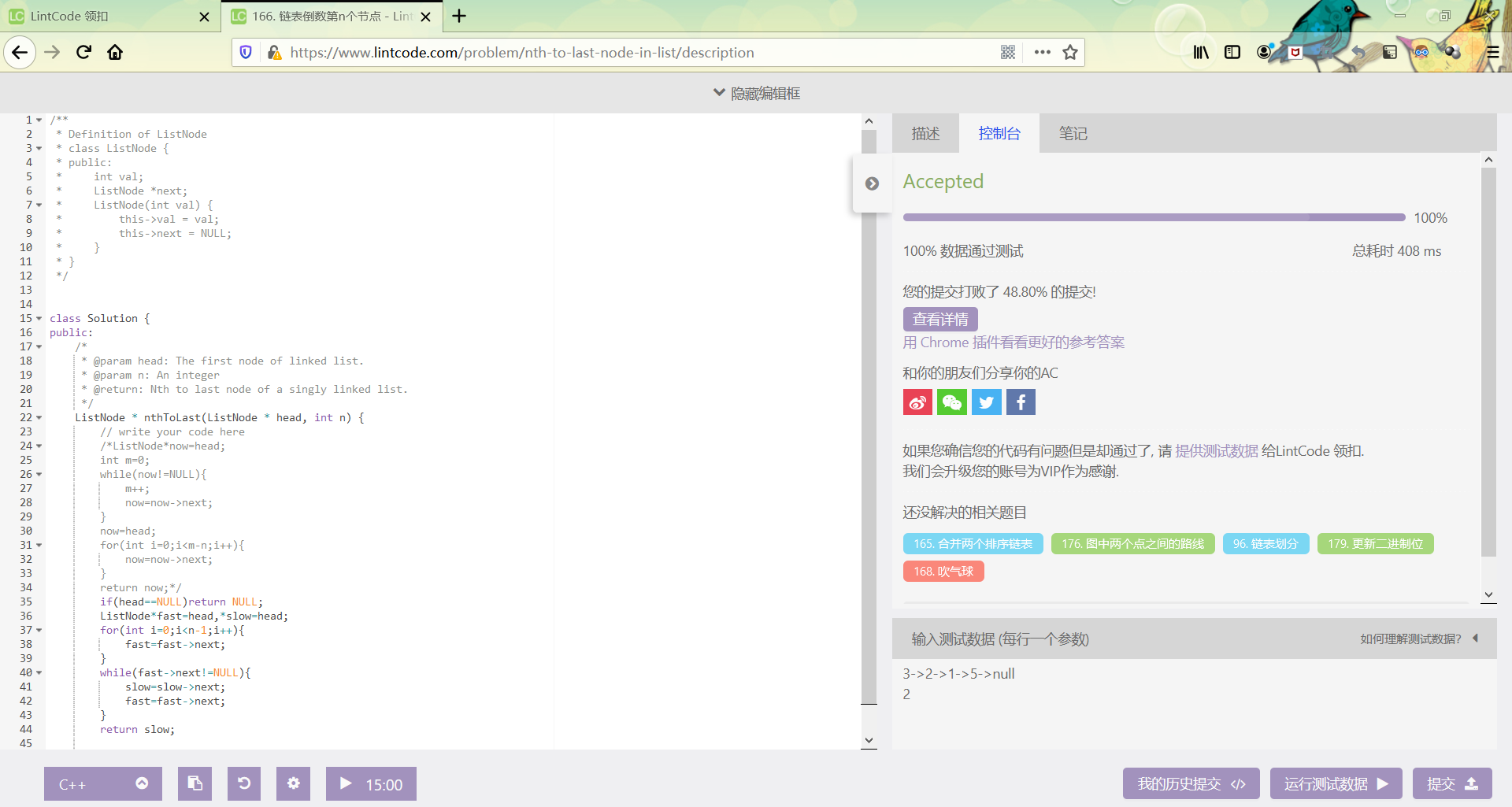
思路：新建一个哑节点，可以避免讨论是否为第一个节点。方法就是普通的插入排序。

**21、两两交换节点**



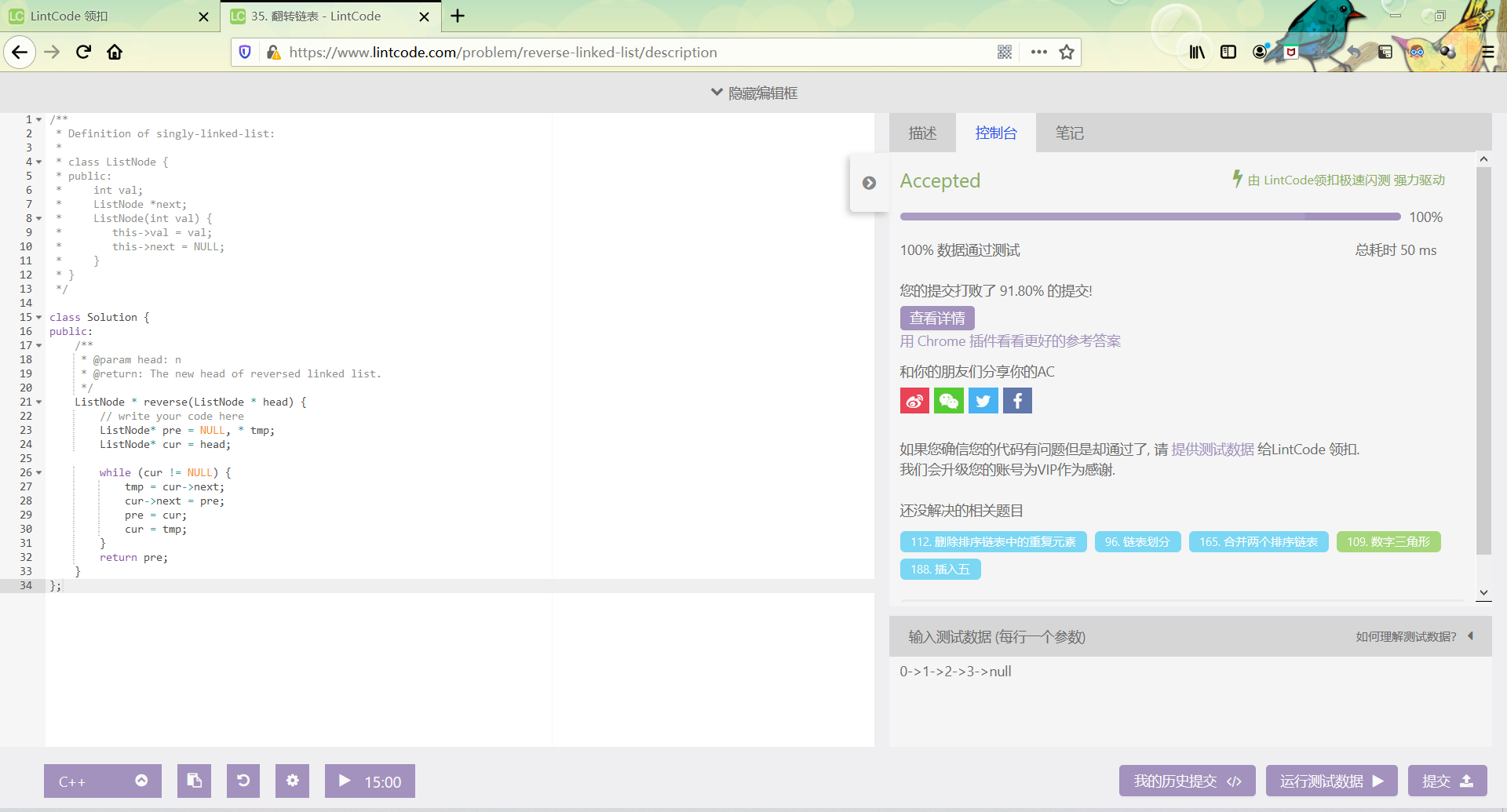
思路：用三个指针指向前中后三个节点，两两更改next即可。结束后now指针指向后第二个节点，用一次循环即可。

**22、链表倒数第n个节点**

****

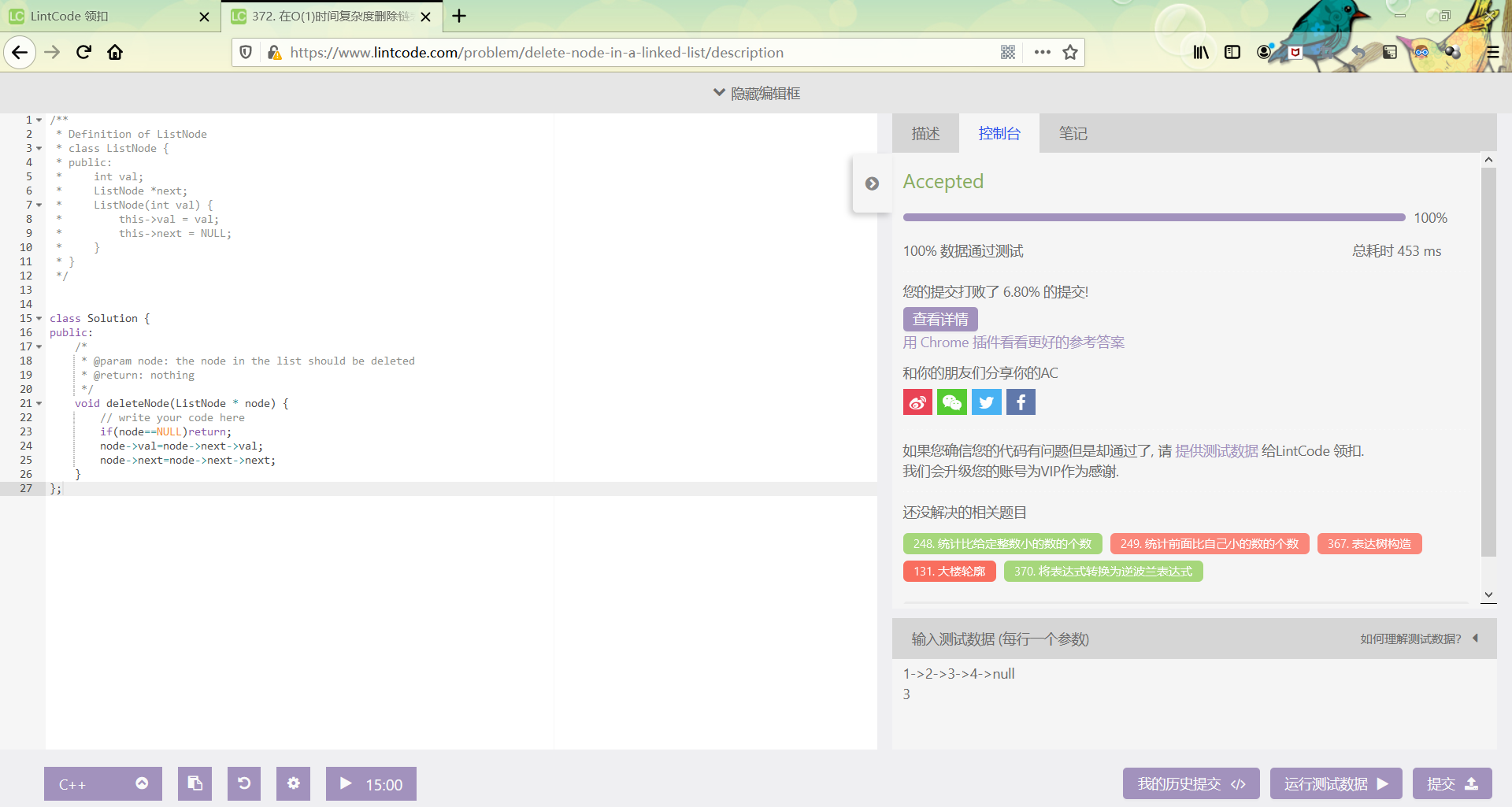
思路：快慢指针，慢指针在快指针前第n个节点，快指针指向最后一个节点时，慢指针指向倒数第n个指针。

**23、反转链表**

****

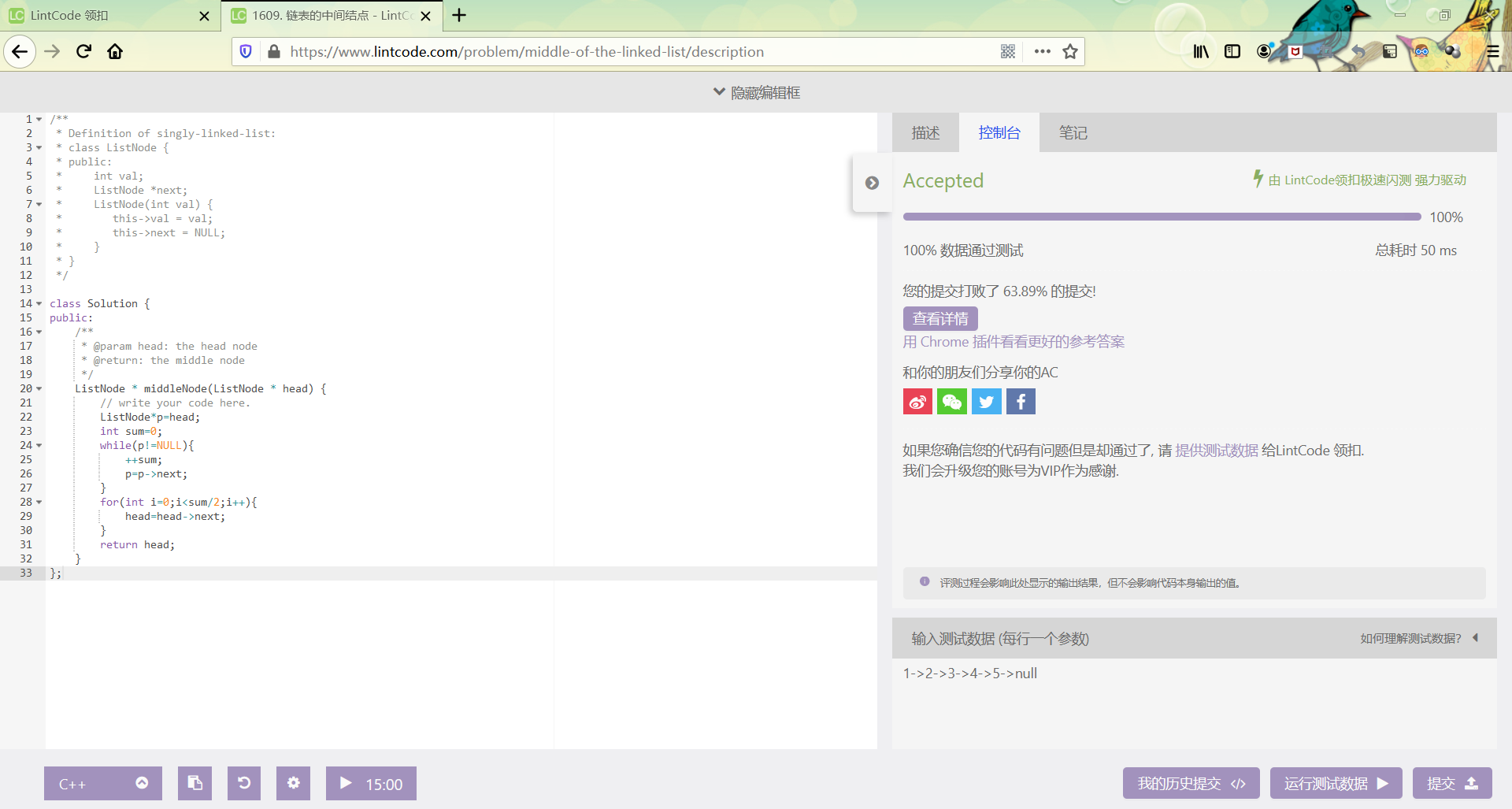
思路：从首节点开始，将该节点的next指向前一个节点，在一次循环中实现。需要三个指针，pre保存前一个节点,cur保存现节点，numb用于暂时存储下一个节点。最后返回原本最后一个节点。

**24、在O(1)复杂度删除链表节点**

****

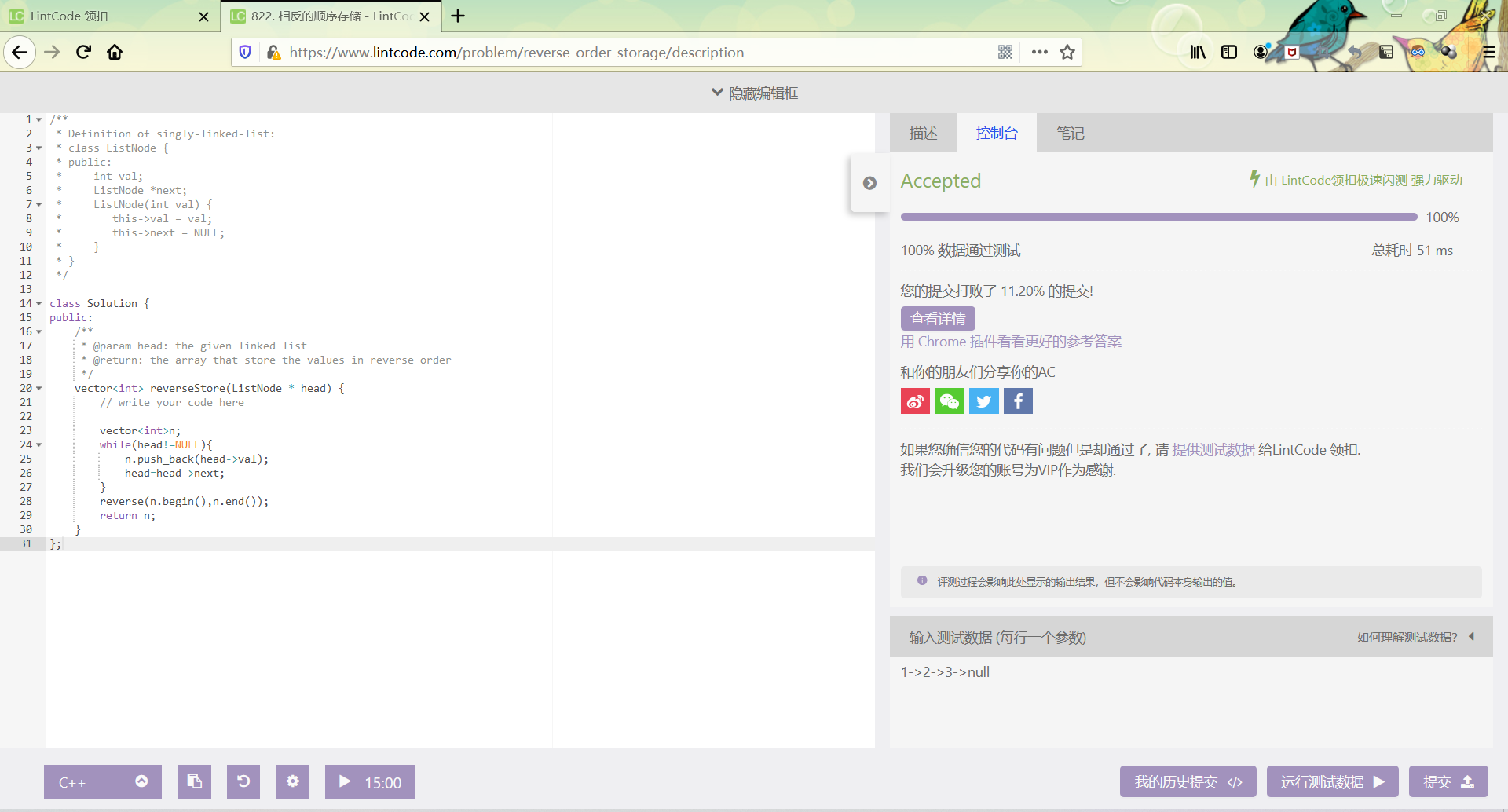
思路：将本节点赋值为后一个节点，next指向后一个节点的后一个节点。因为只给了一个节点，没有前一个节点，所以只能用这个方法。

**25、链表的中间节点**

****

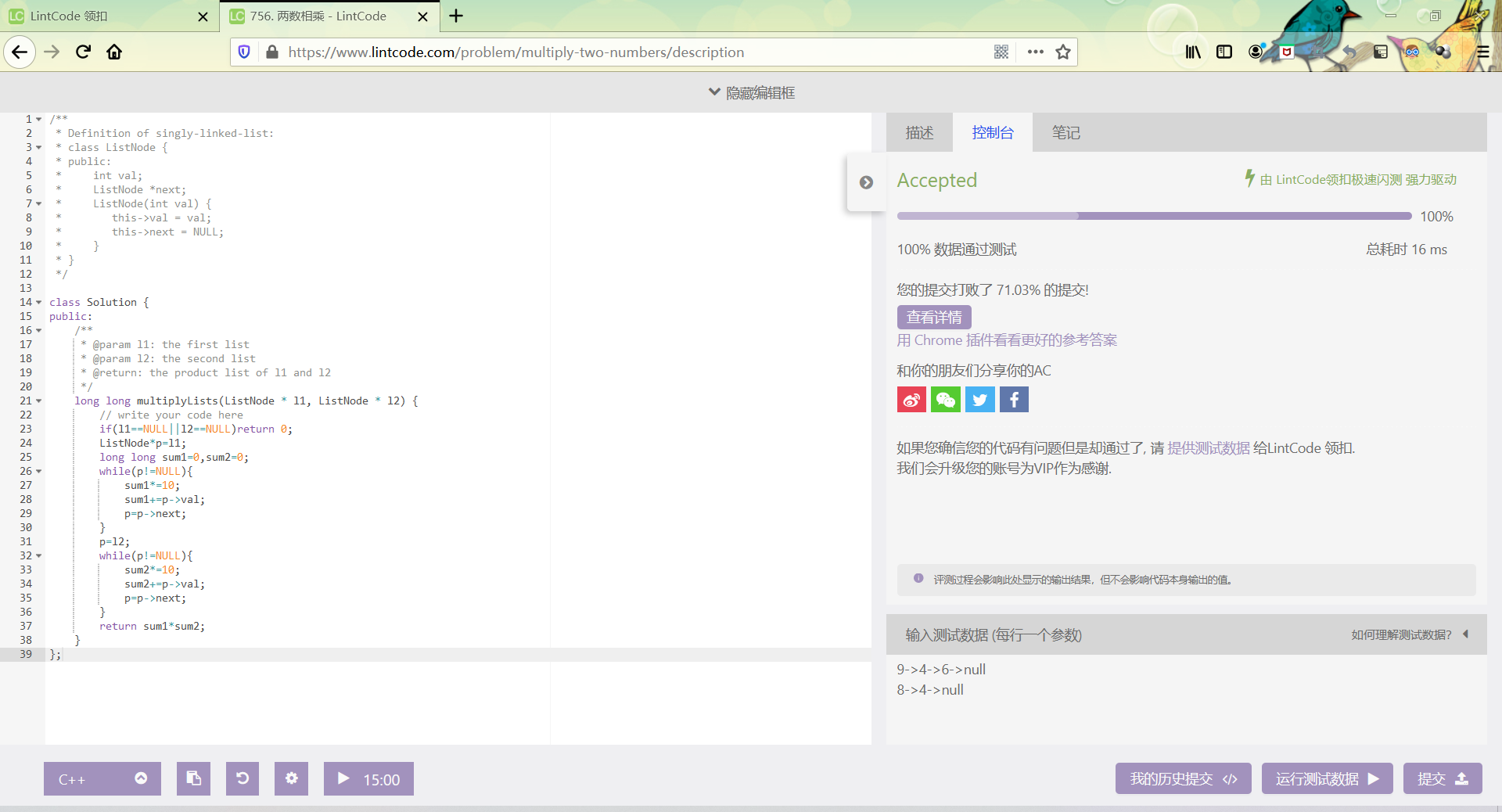
思路：一次循环得出长度，第二次循环使指针指向中间节点，返回中间节点

**26、相反顺序存储**

****

思路：先正序存储在数组中，再翻转数组

**27、两数相乘**

****

思路：两次循环用\*10的方法得到链表中存储的数，再相乘