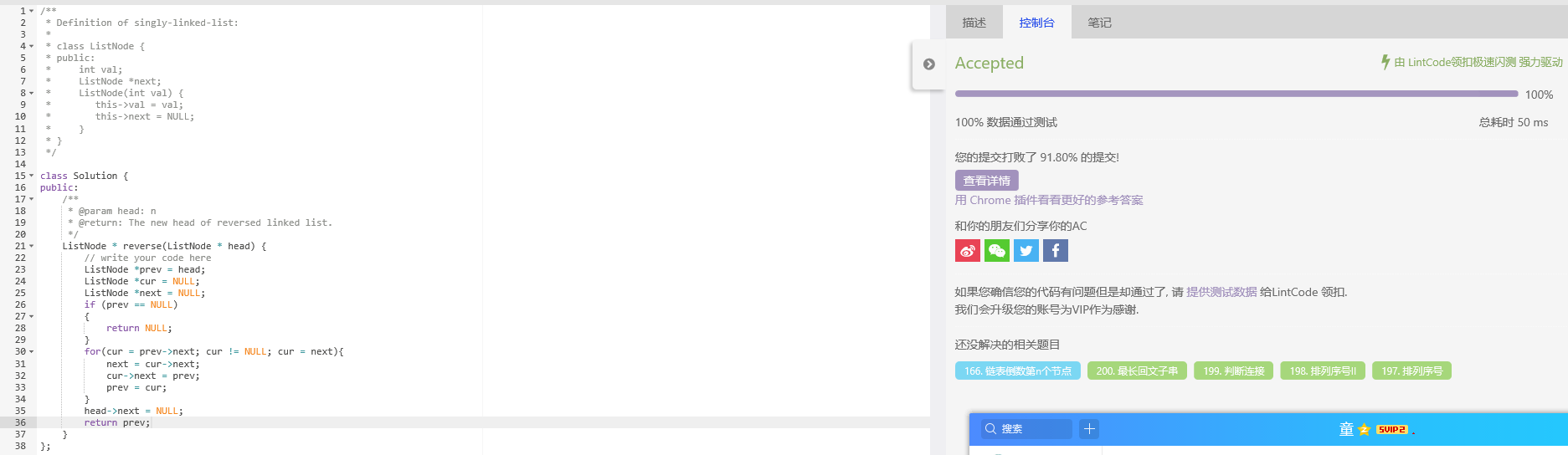
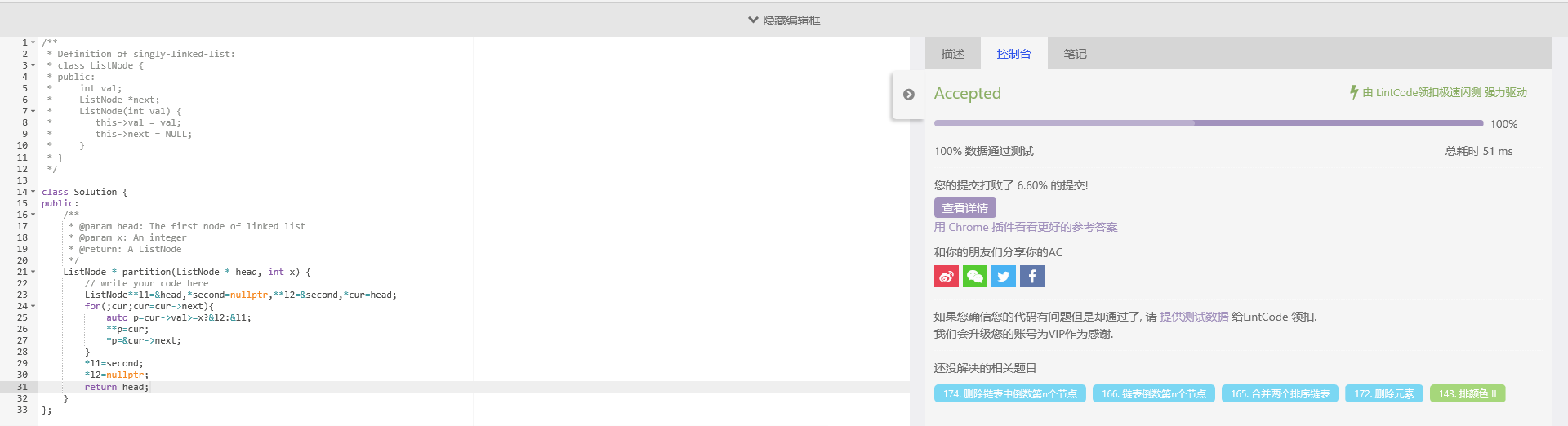
**暑期算法训练报告**

翻转链表



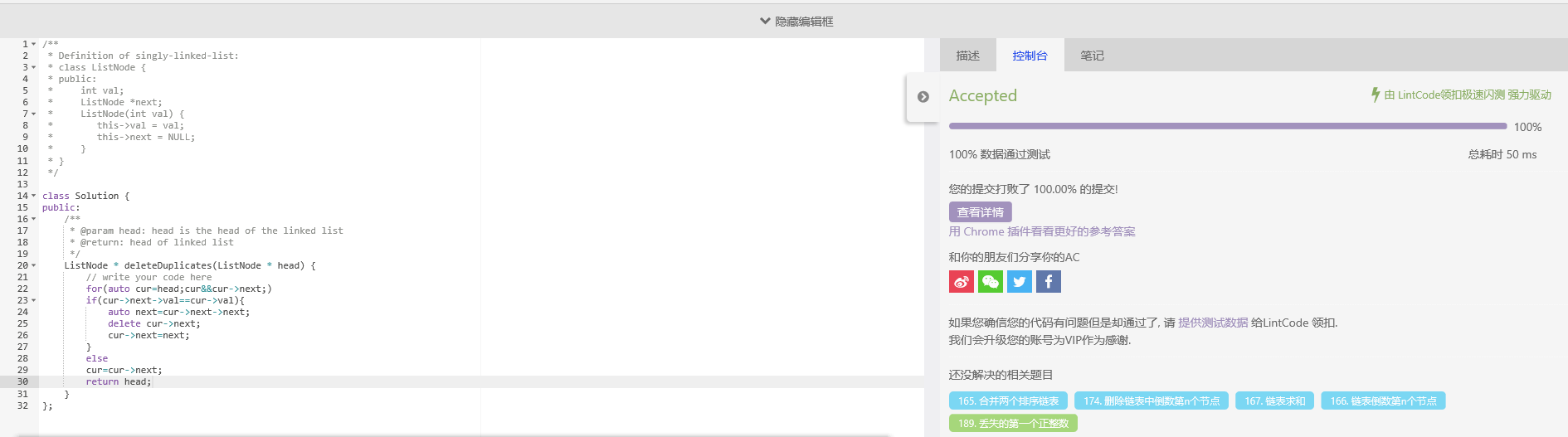
此题采用了原地翻转的算法，即遍历链表中所有节点，当当前节点仍有效时（不为NULL），则令当前节点的下一节点（curr->next）等于当前节点的上一节点（last）完成此部分的翻转，最后更新当前节点（curr）与上一节点（last）完成遍历。

链表划分



此题算法为新建两个节点类型单链表分别用于代表两个链表的头节点，新建两个节点分别用于存放节点数据，通过遍历一次单链表（head）分别找到大于和小于等于目标数的节点并存入两个链表。最后把两个链表连接即完成此题。

删除排序链表中的重复元素



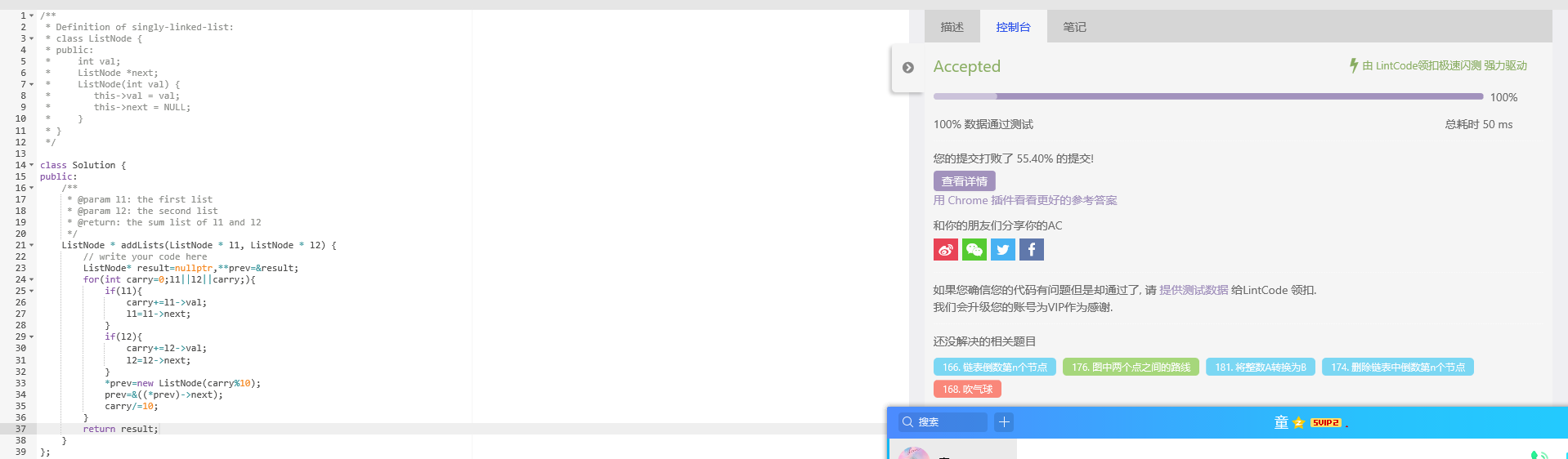
此题算法为遍历列表中的所有节点，因为此题中为排序列表，故相同值的节点总是排在一起的，而一定是从第二个节点开始才有可能删除的，故不需要对第一个节点讨论，对符合条件的节点可直接在原列表中删除。

合并两个排序链表



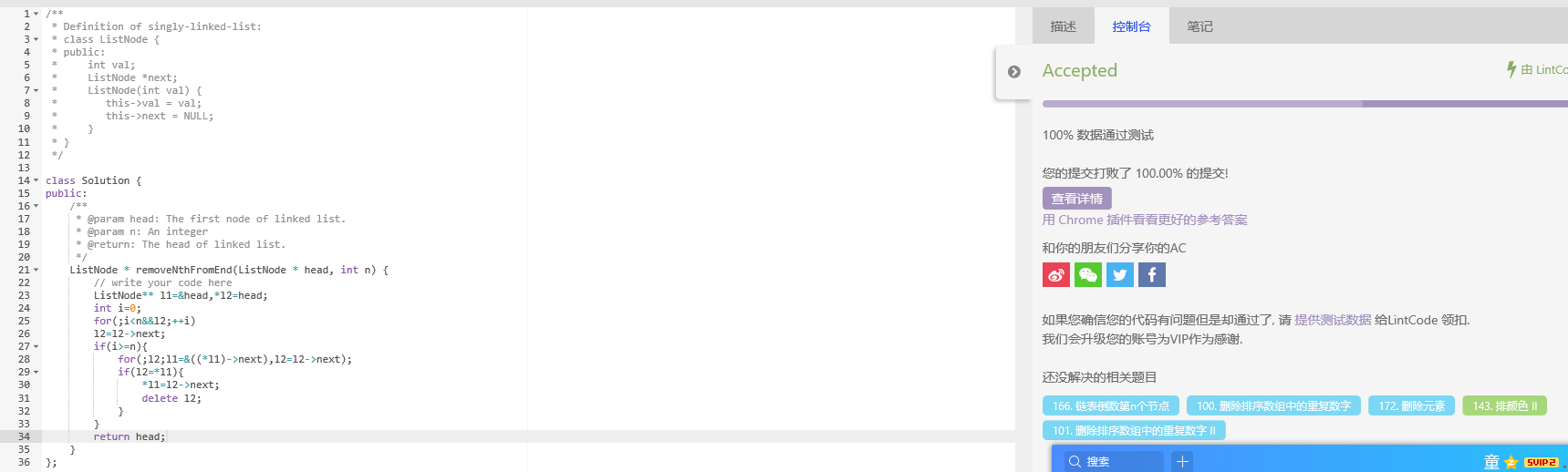
此题算法为新建一个链表（head）用于存放结果，当两个链表（l1、l2）都不为空时，判断当前两链表的节点值大小，将较小的节点接入head中，之后将此较小节点的链表后移一位，继续判断直至遍历完其中一个链表。

链表求和



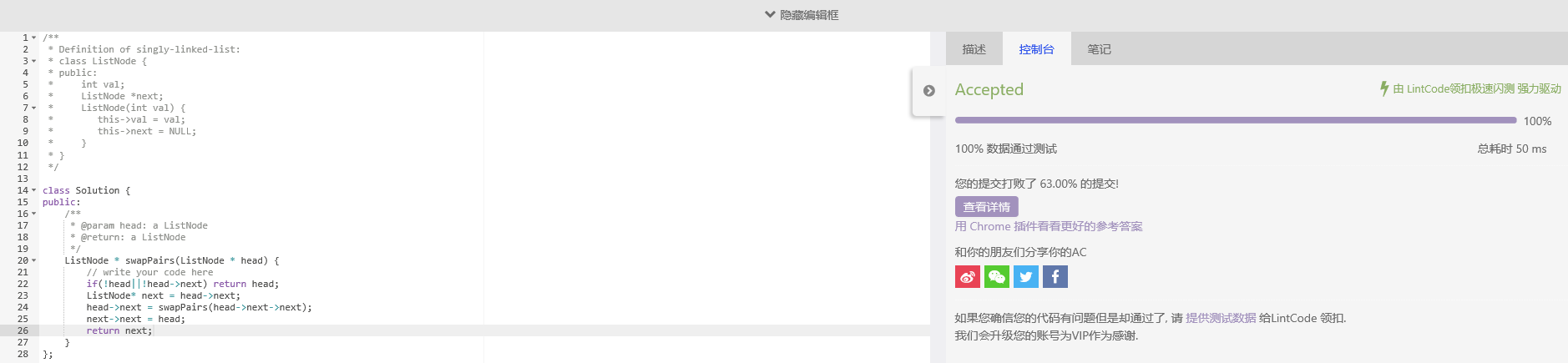
新建一个链表，然后从头节点开始对两数进行相加，用carry表示进位，相加后的值从头结点开始储存在新链表中，这样在一次循环中即可完成。

删除链表中倒数第n个节点



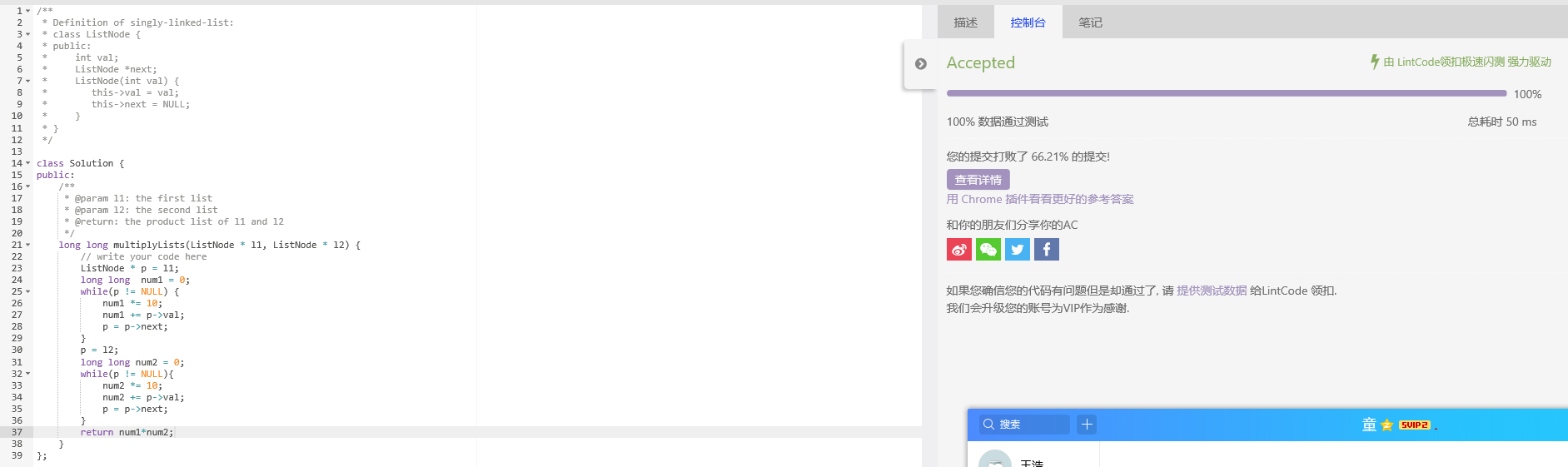
一次循环得出链表长度，再进行一次循环，找到倒数第n个节点，删除即可。

两两交换链表中的节点



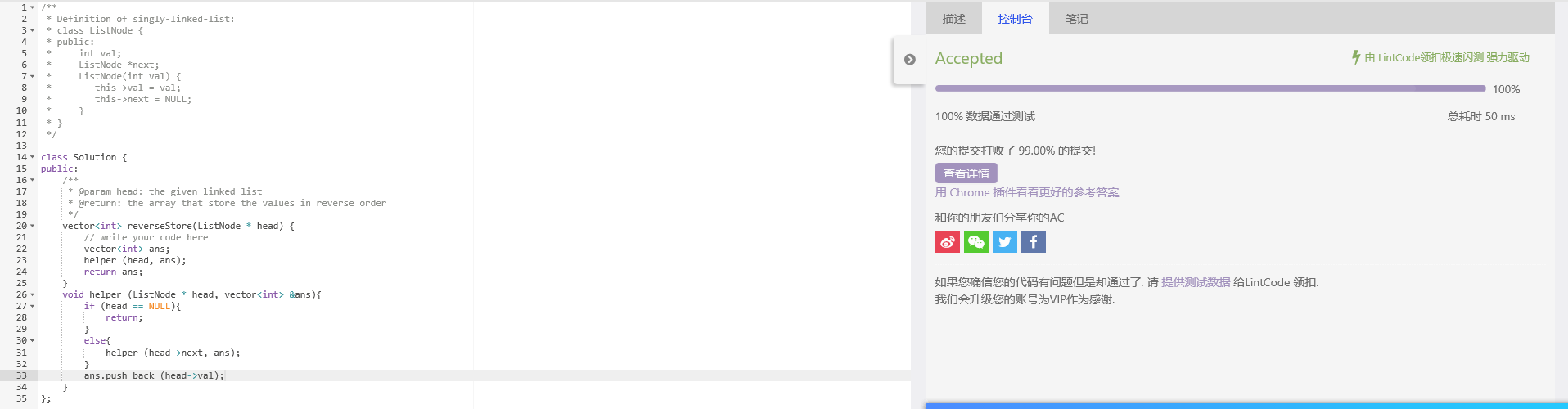
先设定递归推出条件，，也就是到最后剩下一个空节点或者是单个节点的情况下，返回节点本身。然后先把第二个节点保存下来，将第三个节点进行递归，返回的结果被第一个结点指向。然后之前保存的第二个节点指向第一个节点，返回第二个节点即可。

两数相乘



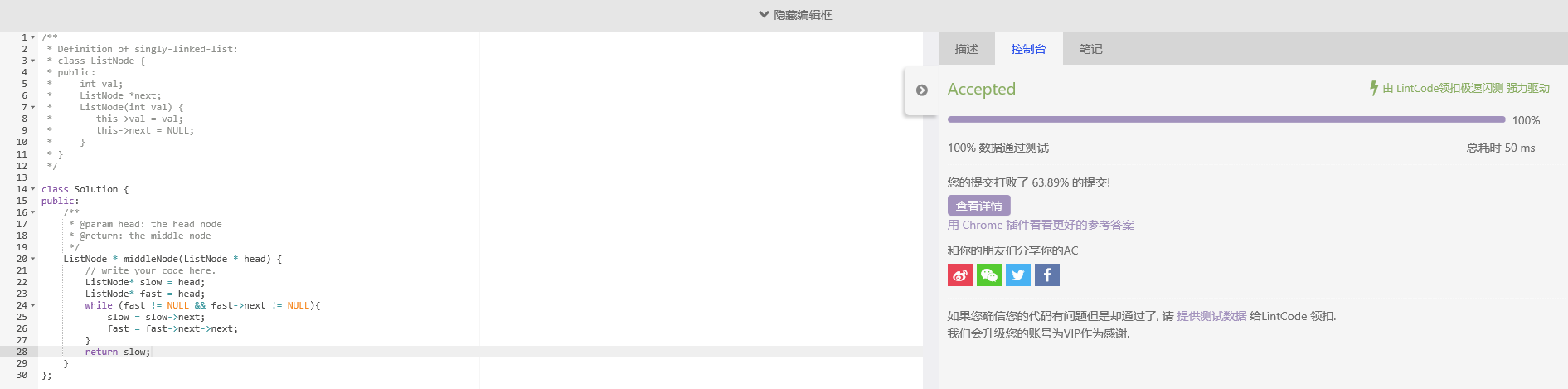
按照链表遍历得到两数相乘返回结果

相反的顺序存储



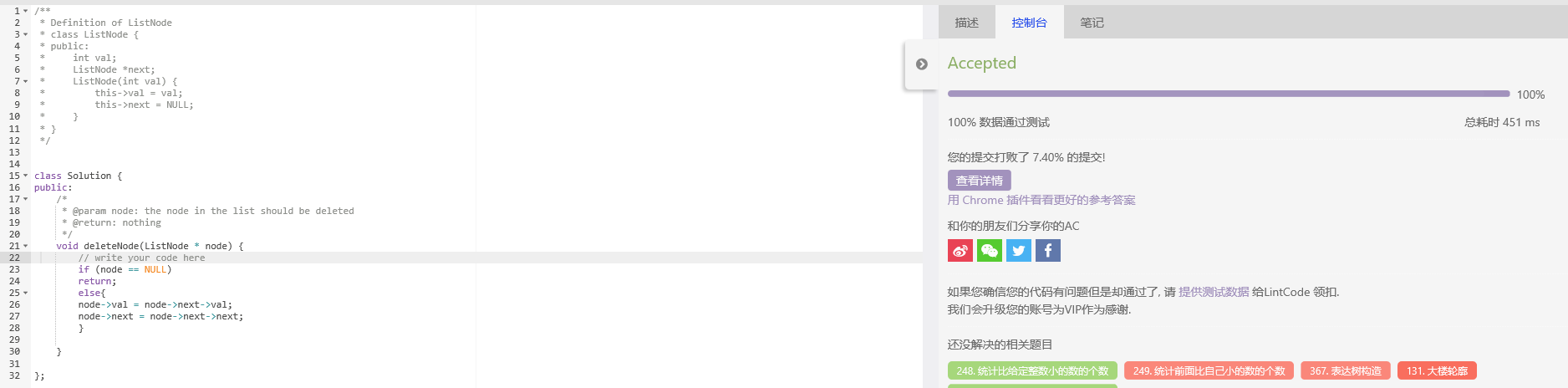
先正序存储在数组中，再翻转数组

链表的中间结点



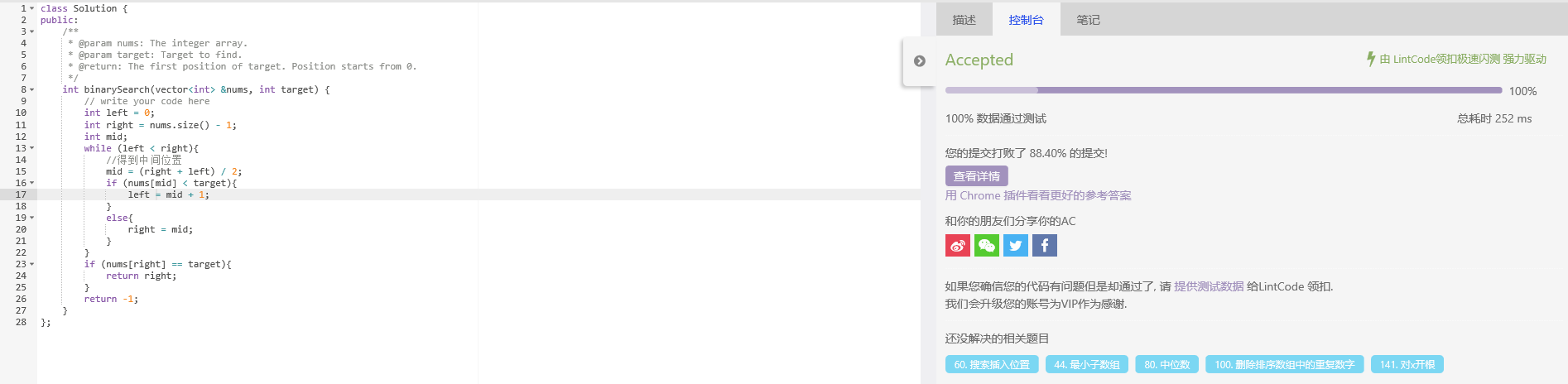
当用慢指针slow遍历列表时，让另一个指针fast的速度是它的两倍；当fast到达列表的末尾时，slow必然位于中间

在O（1）时间复杂度删除链表节点



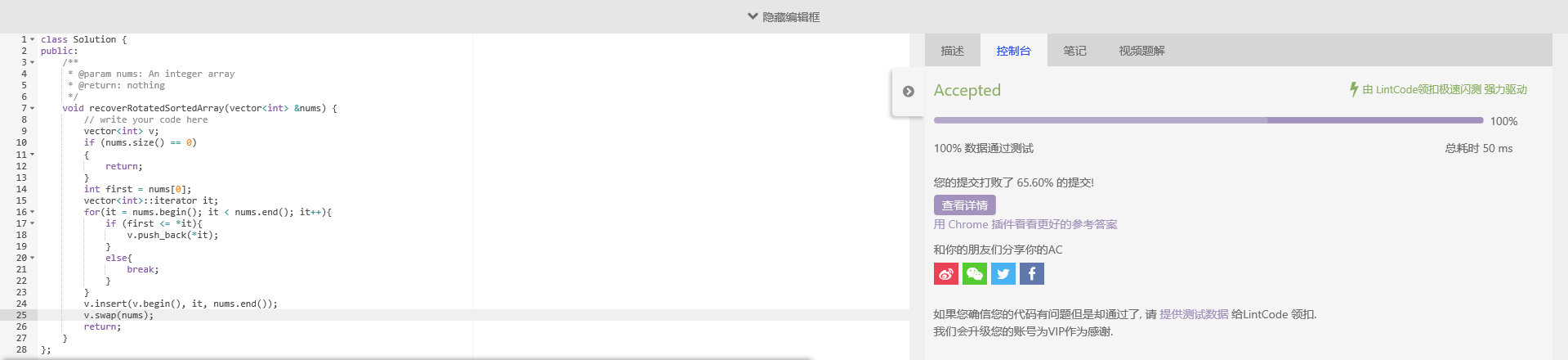
把删除的的node的值改成它的下一个node的值，它的next改成它的下一个的next即可

二分查找



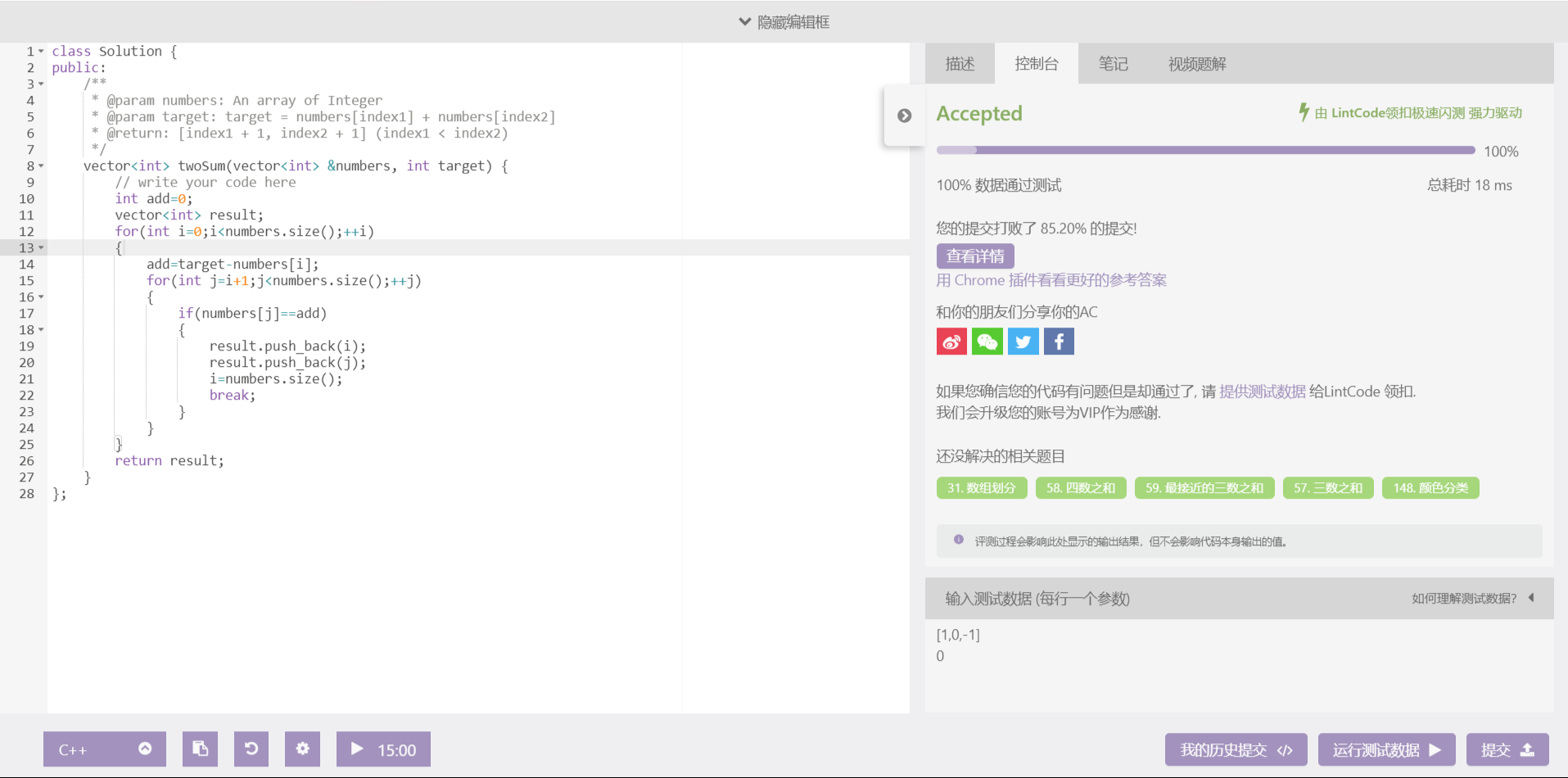
因为是排序数组，则比较target与中位数的大小，大于则继续依此法搜索前半部分，小于则搜索后半部分，依次进行，最后找到target。

恢复旋转排序数组

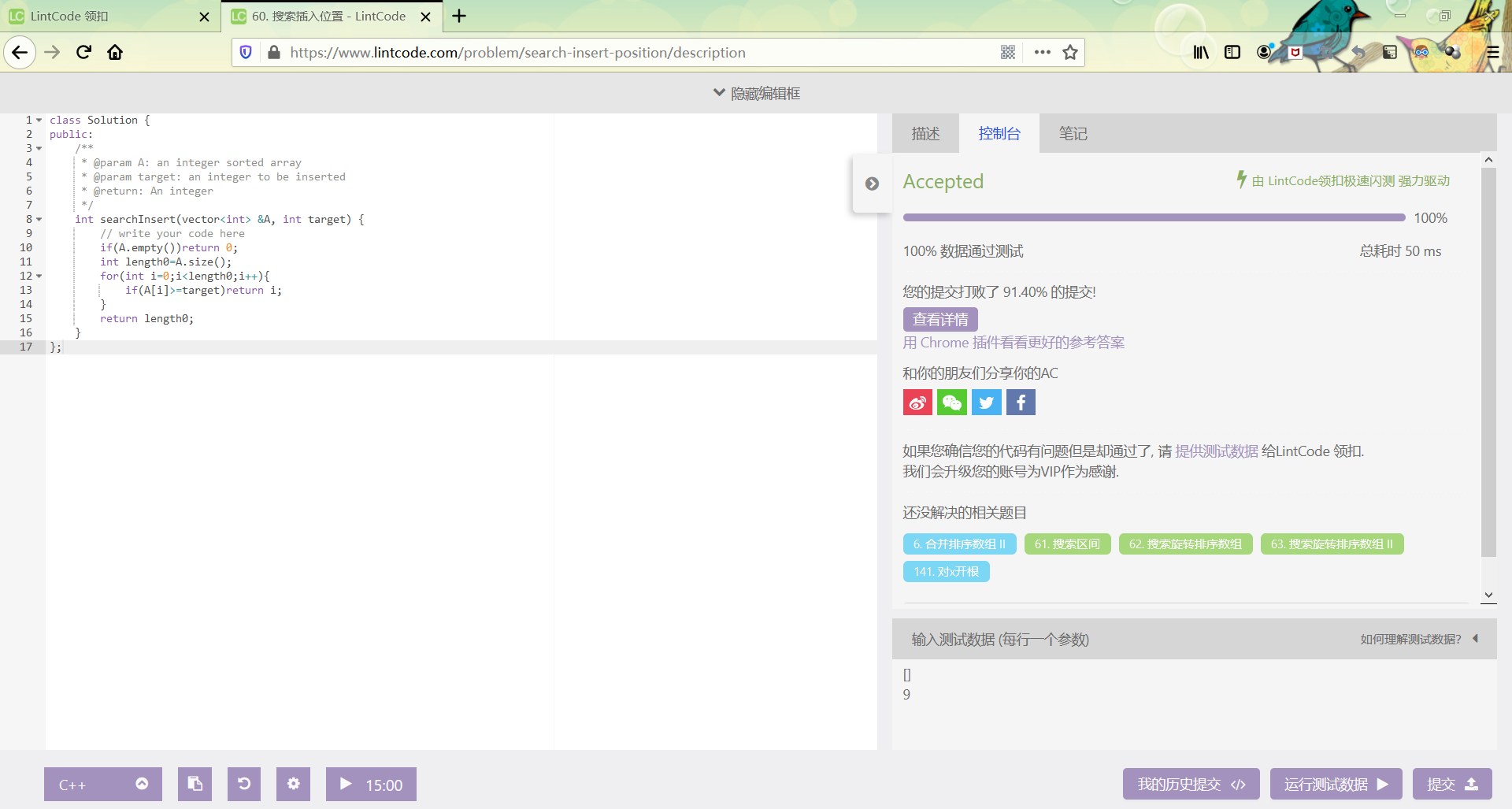


先通过一次循环找到旋转排序的起点，再通过一次循环将起点及之后的数字整体前移。具体实现过程是每次将前面的数据整体后移一个单位，从第一个数组空间开始空出来前面的一个数组空间供后面的元素插入。

两数之和

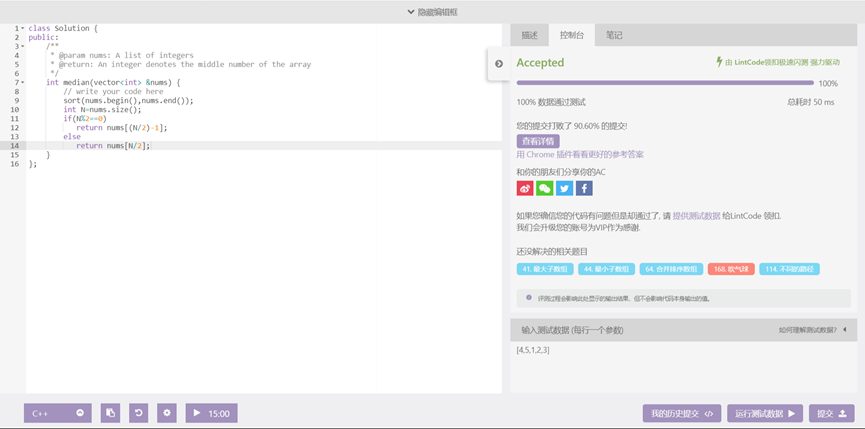
此题算法为嵌套循环遍历数组找到两个满足条件的数。首先新建数组（result）用于保存结果，之后循环遍历数组寻找第一个数（numbers[ i ]），因为有负数存在的可能，故当第一个数大于目标数时无法判断其正确与否。在循环中内嵌循环用于寻找第二个数(numbers[ j ]，当满足条件时先后将找到的两个下标尾插到结果数组中，最后跳出两个循环（跳出内层循环用break，将i改为numbers.size()跳出外层循环）完成此题。

搜索插入位置

****

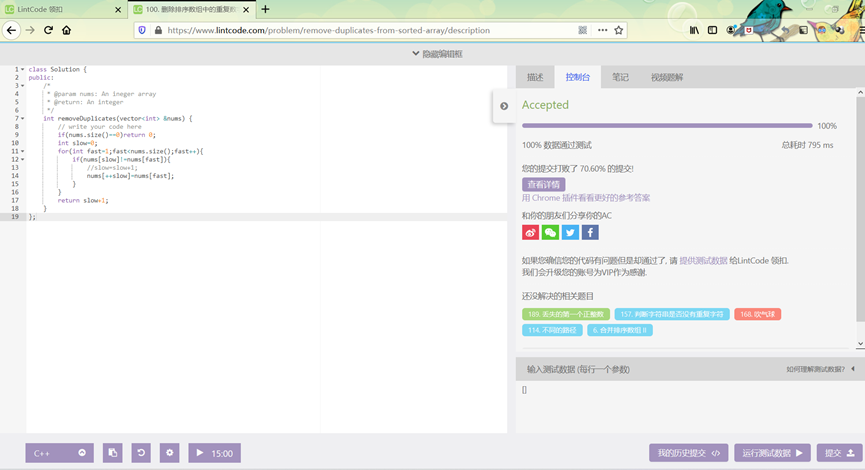
因为是排序数组，则进行一次循环，直到数组中的数大于target，则直接插入。若数组长度为0，则直接返回0.若没有大于target的值，则插入到最后。

中位数



此题算法为对已知数组排序之后输出对应位置的数据，但是自己编写排序的代码比较费力，运行时间也很多，之后找到了vector数组的排序函数sort，完成此题。

删除排序数组中的重复数字



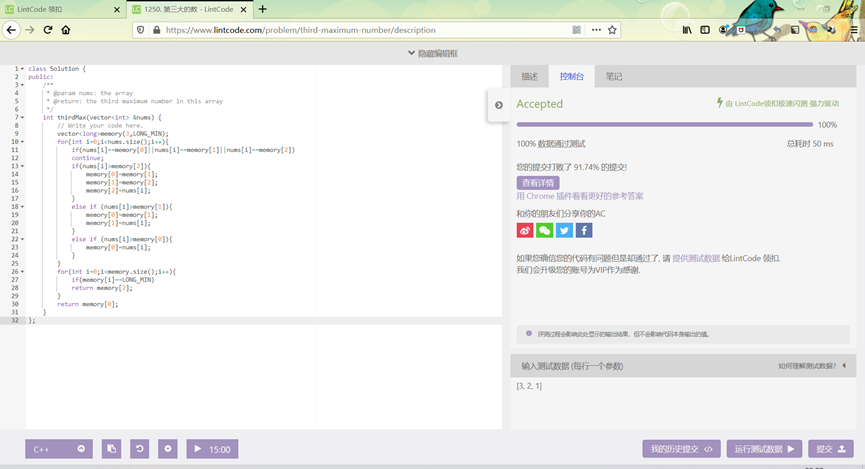
使用快慢指针，在一次循环中，若快指针不等于慢指针所指的值，则用快指针所指的值覆盖慢指针下一个值，这样当慢指针所指的重复值全部被后面的值覆盖后，覆盖前面的值会在接下来的时间里 也被覆盖。

合并排序数组



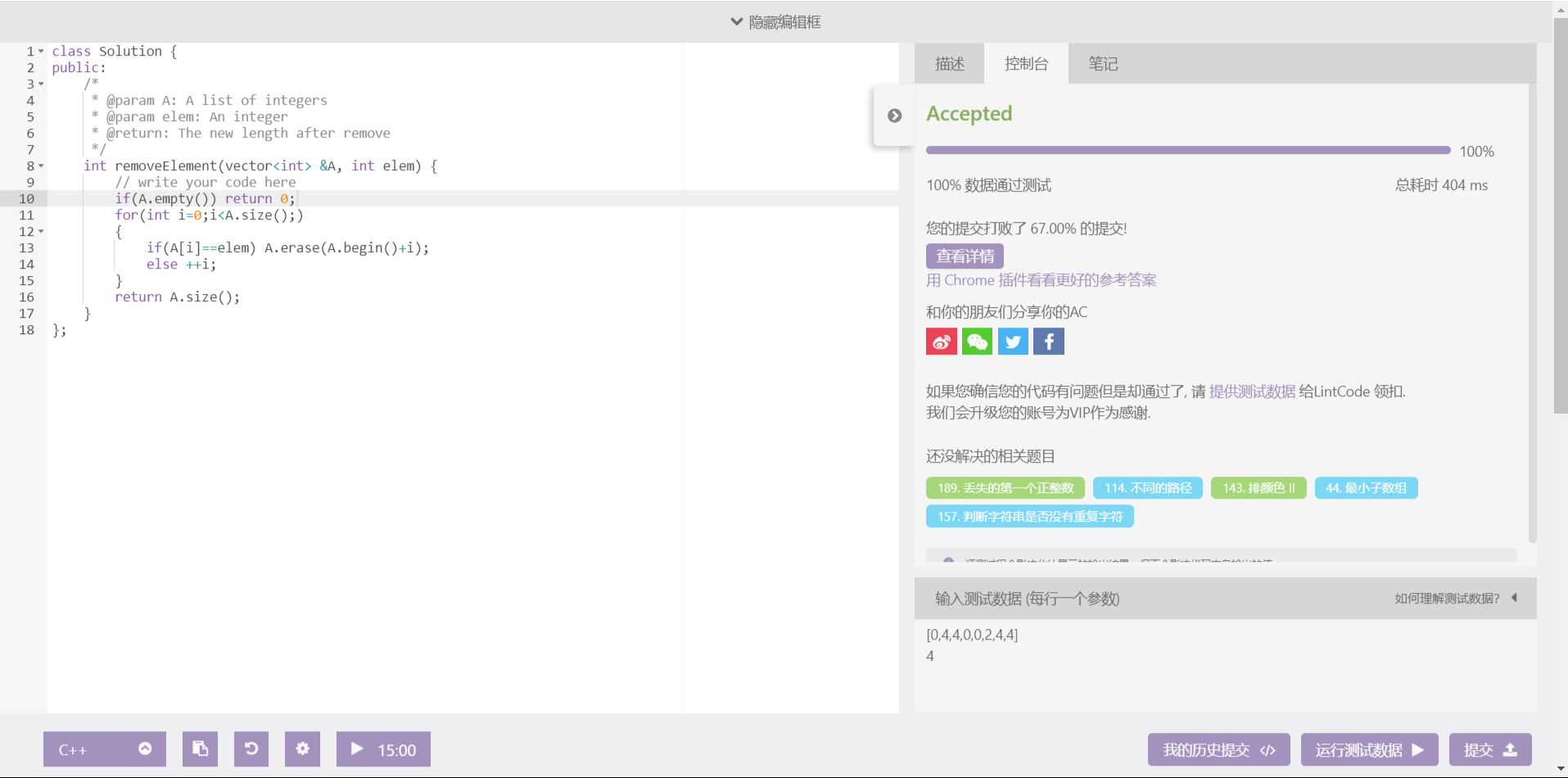
此题算法为从后向前遍历A数组和B数组，若B数组当前元素大于等于A数组的最后一个有效元素（A数组的长度不是其有效长度，是A数组有效长度与B数组长度的和m+n），则将B数组的当前元素写入A数组的当前位置，反之将A数组的当前元素写入A数组的当前位置（先排列最大的元素）完成此题。

第三大的数



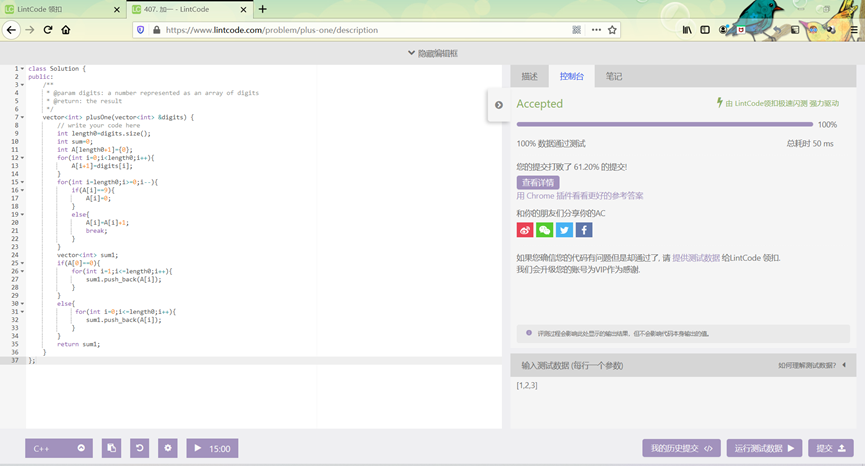
用三个变量来存放1、2、3大的元素，遍历数组，找到第三大的数，没有则返回第一大的数，忽略重复的数。

删除元素



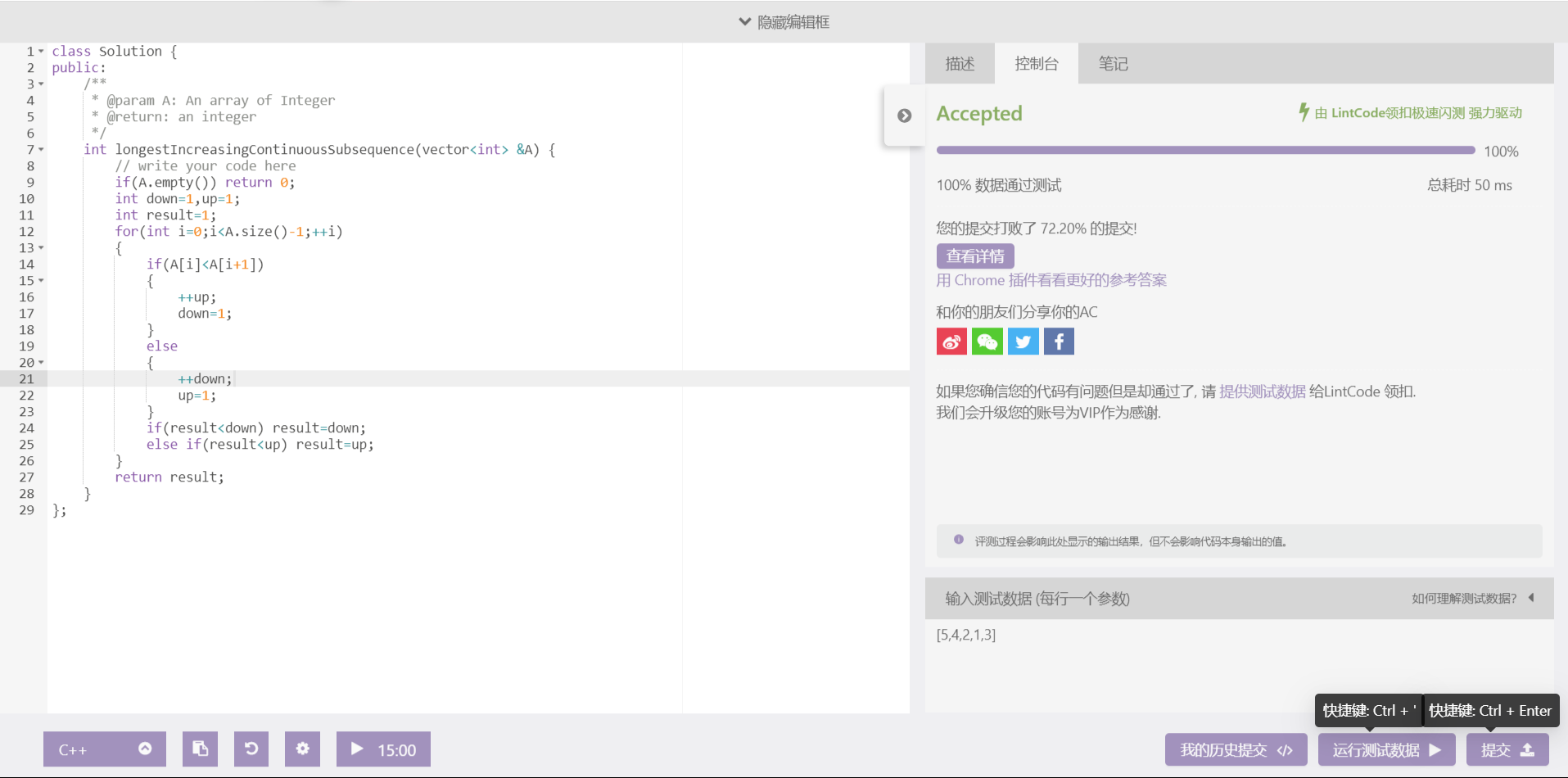
对每个数组元素加一，若等于10则赋值为0，下一个元素进位1。需要用到一个循环，将所有节点相加。当其中一个链表结束了之后需要跳出循环，重开一个循环对另一个链表操作。

加一



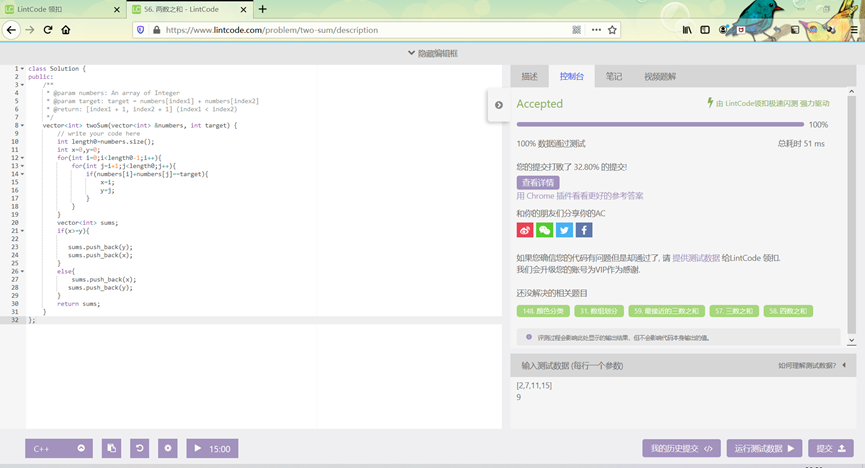
对每个数组元素加一，若等于10则赋值为0，下一个元素进位1。需要用到一个循环，将所有节点相加。当其中一个链表结束了之后需要跳出循环，重开一个循环对另一个链表操作。

最长上升连续子序列



此题算法为遍历数组并判断相邻两数的大小，找到最长的连续子序列。首先定义变量（down , up）用于保存从右向左和从左向右的上升序列，之后遍历数组，判断当前元素与下一个元素的大小，当当前元素大于下一元素时，说明当前序列是从右向左的上升序列，则自加down；对于up变量，此种上升序列已经打断了从左向右的序列，则更新up为1。在结束每一次循环的时候，判断up, down的大小，将更大的一个赋值给结果变量（result）。最后输出result完成此题。

两数之和



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。

最短无序连续子数组



此题算法为：首先新建数组（result）用于存放对已知数组（nums）排序之后的结果，并定义记录子数组位置的变量（begin, final）。之后同时正向遍历result , nums数组，找到begin；同时反向遍历result, nums数组，找到final。最后若begin , final值相等，说明此数组不需要重新排序，返回0即可；反之则返回（final-begin+1）即完成此题。