1.重复计数

给定一个整数数组，您的任务是计算重复数组元素的数量。 重复是被定义为两个或更多相同的元素。 例如，在数组[1, 2, 2, 3, 3, 3]中，两个2是一个重复，三个3是一个重复。你需要返回一个数组，包含所有非唯一（重复）值的数字，且顺序与给定的数组相同。

![7N6[HQS2Y]5]45[RZIK](RI](data:image/png;base64,)

思路：

1.定义一个新容器用于存放符合要求的数据

2.遍历一边数组，从第二个数据开始，与前一个比较，如果相等就计数2

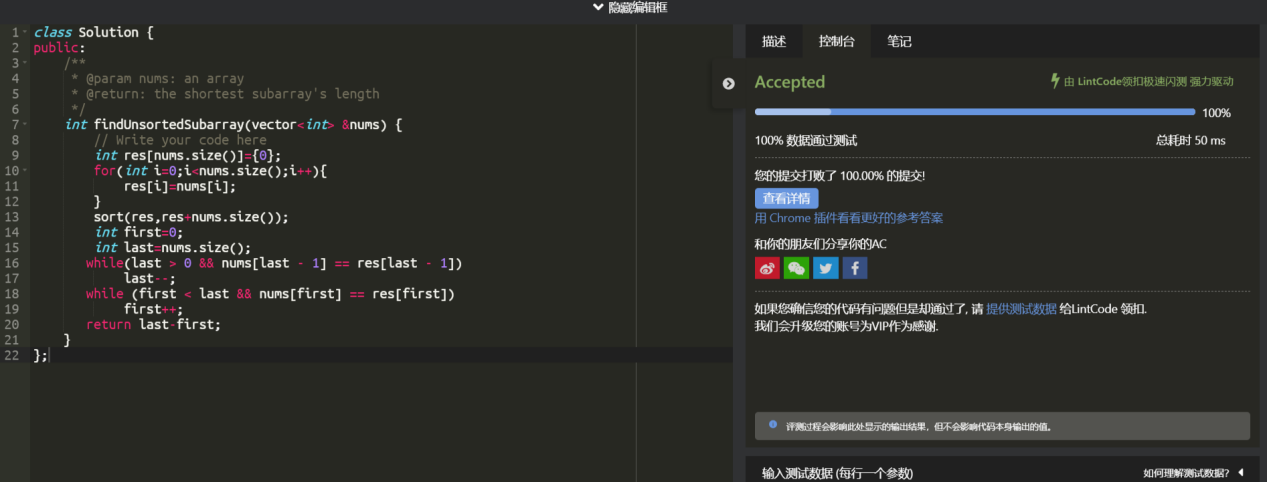
3.在每次比较后，判断计数如果为2就代表出现了两次，则放入新容器

4.注意如果次数多与2次，计数仍会增加，但只需判断重复了两次即可，既无动作

2.最短无序连续子数组

给定一个整数数组，你需要找到一个连续子数组，如果你只按升序对这个子数组进行排序，那么整个数组也将按升序排序。

你需要找到最短的这样的子数组并输出它的长度。



思路：

1.将所给数组放入常规数组中

2.对新数组sort排序，这样就得到了原数组的排序数组

3.只需要两者一一比较，找到最短的连续不相的对应序号，后者减前者即为答案

4.注意分两次循环，更新last和first

3.合并排序数组

合并两个排序的整数数组A和B变成一个新的数组。



思路：

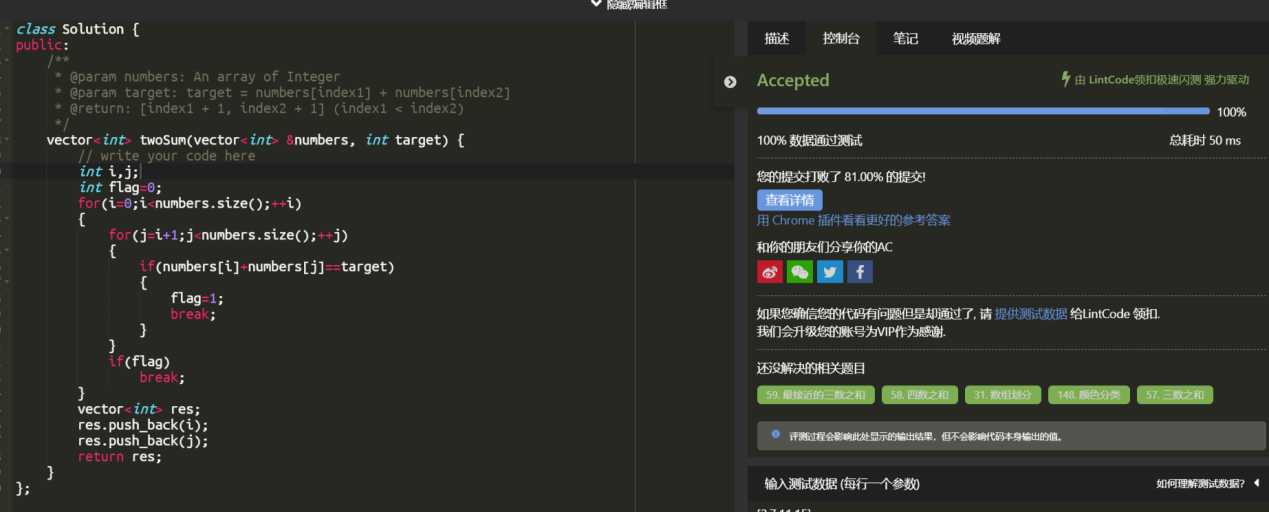
1.直接将后一个数组接入前一个数组

2.对前一个改动过的数组sort排序

4.两数之和

给一个整数数组，找到两个数使得他们的和等于一个给定的数 target。

你需要实现的函数twoSum需要返回这两个数的下标, 并且第一个下标小于第二个下标。注意这里下标的范围是 0 到 n-1。



思路：

1.提前定义两个常量代表数组两个数的下标

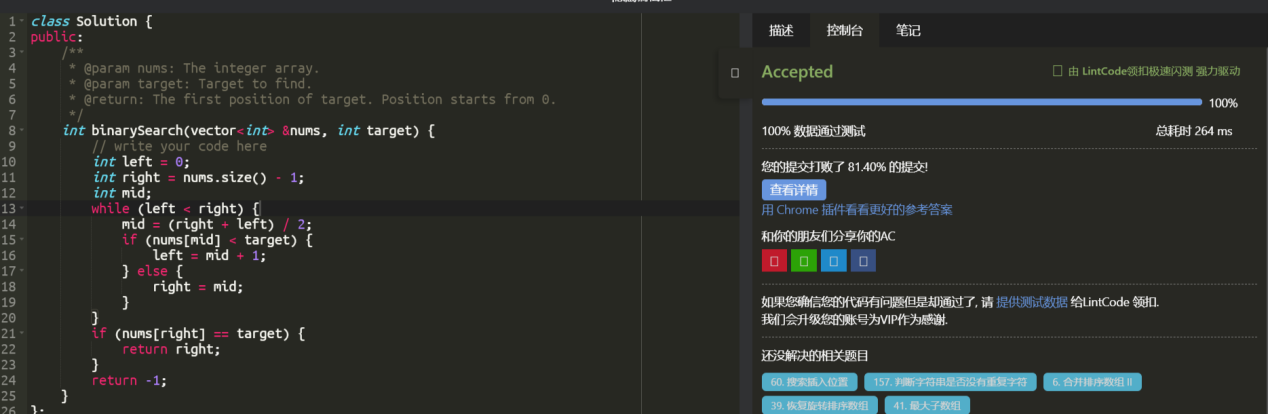
2.遍历数组，因为需要第一个下标小于第二个下标，第二个数直接从第一个数后面开始寻找

3.遍历判断是否等于target，一旦符合直接break

4.定义一个容器存放下标

5.二分查找

给定一个排序的整数数组（升序）和一个要查找的整数target，用O(logn)的时间查找到target第一次出现的下标（从0开始），如果target不存在于数组中，返回-1。



思路：

1.首先无论数组是奇是偶，直接除以2得到的都是二分法需要比较的数据下标

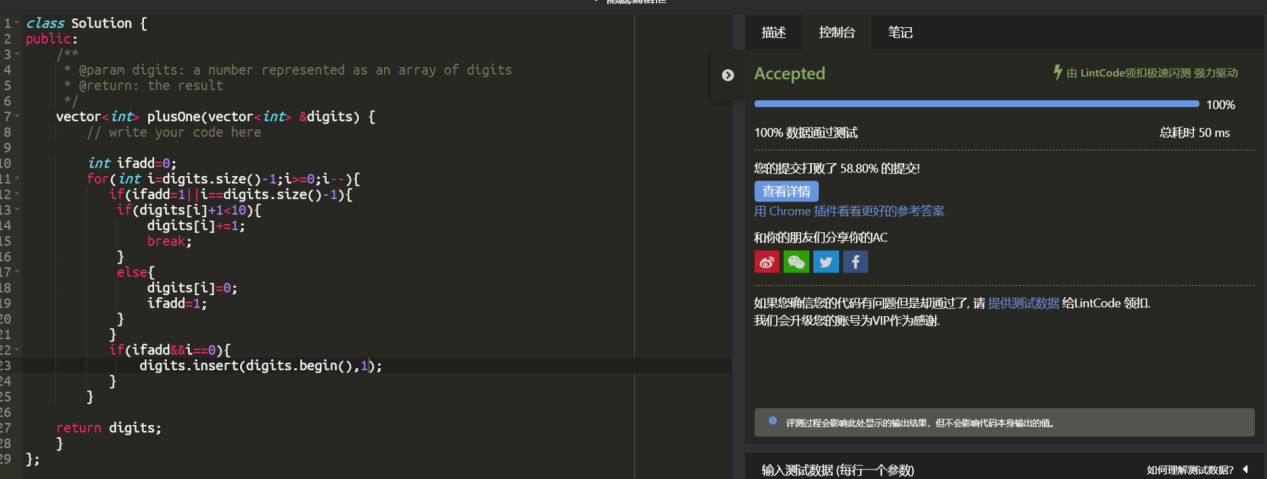
2.循环，每一次用中间的值和target比较。如果小于target，左边界缩小到当前中间下标右一个；如果大于则相反

3.判断当相等的时候直接返回下标，如果到循环结束没有返回，则返回-1

6.加一

给定一个非负数，表示一个数字数组，在该数的基础上+1，返回一个新的数组。

该数字按照数位高低进行排列，最高位的数在列表的最前面。



思路：

1.提前定义常量表示是否进位

2.根据加法的思想，循环需要从数组最后一位开始。当操作最后一位数时，判断加一之后知否为10，是则判断进位且此位变为0，否则直接加一。这一步判断的条件设置为操作最后一位或进位，如果无需进位则无操作。

3.考虑进位为需要增加数组为数的情况，如果进位一直为1且操作的已经是最后一位，使用insert函数在0位2上增加一个1。

7.最长上升连续子序列

给定一个整数数组（下标从 0 到 n-1， n 表示整个数组的规模），请找出该数组中的最长上升连续子序列。（最长上升连续子序列可以定义为从右到左或从左到右的序列。）



思路：

1.定义ifsame表示与前一个数据的比较关系是否一致。大于为1，小于为2

2.遍历之前，定义一个compare数组用于存放大小关系数据

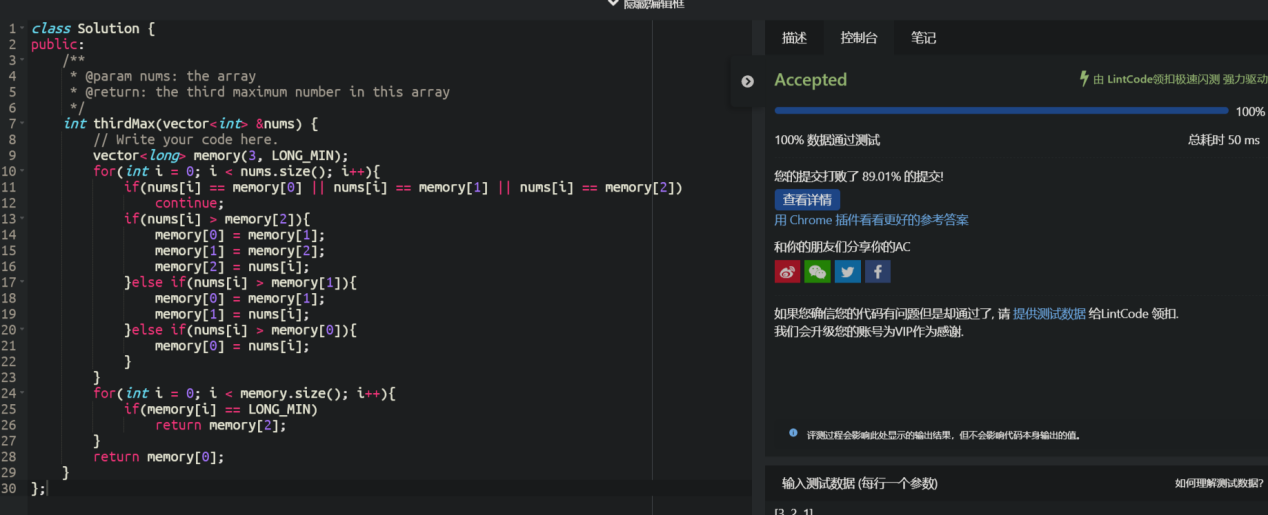
3.从1开始遍历，每一次与前面的比较大小，将ifsame的值放入compare数组

4.遍历compare数组，前后数据一致则计数加一否则重置为2

5.注意提前定义一个result，每次与res比较，只保存res的最大值

8.第三大的数

给定一个非空数组，返回此数组中第三大的数。如果不存在，则返回数组中最大的数。要求算法时间复杂度必须是O(n)。



思路：

1.用三个变量来存放第一大，第二大，第三大的元素的变量，分别为one, two, three；

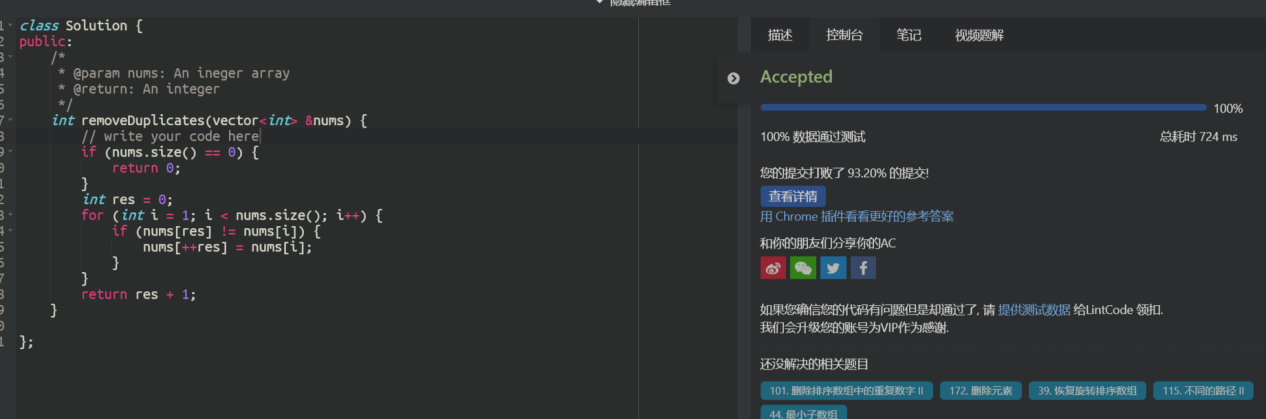
2.遍历数组，若该元素比one大则往后顺移一个元素，比two大则往往后顺移一个元素，比three大则赋值个three；

3.最后得到第三大的元素，若没有则返回第一大的元素。

4.注意在定义memory初始化时，使用int中不存在的数据初始化，在最后判断是否存在第三大的数时直接判断是否还是long类型的最大值。

9.删除排序数组中的重复数字

给定一个排序数组，在原数组中“删除”重复出现的数字，使得每个元素只出现一次，并且返回“新”数组的长度。不要使用额外的数组空间，必须在不使用额外空间的条件下原地完成。



思路：

1.先考虑错误数据的情况

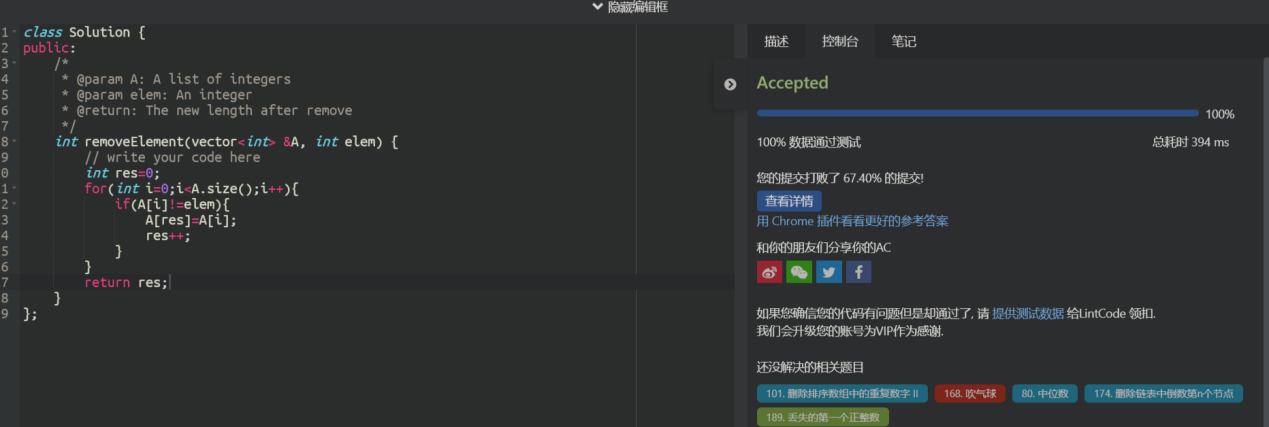
2.定义一个res用于记录修改后数组的长度

3.从1位置开始遍历，因为第一个数据可认为一定不删除。res最初为0，如果判断是不同的数据res加一，这样每次只用判断下标res和当前遍历下标对应数据是否不同。如果不同就计入，如果相同无动作既跳过。

10.删除元素

给定一个数组和一个值，在原地删除与值相同的数字，返回新数组的长度。

元素的顺序可以改变，并且对新的数组不会有影响



思路：

1.定义一个res用于记录修改后数组的长度

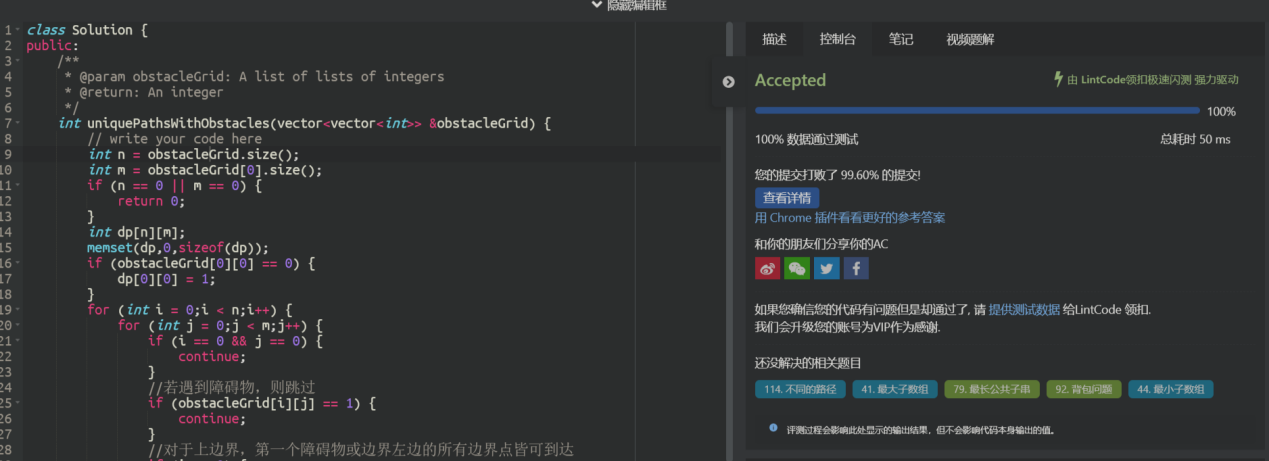
2.res最初为0，如果判断是不同的数据res加一，这样每次只用判断下标res和当前遍历下标对应数据是否不同。如果不同就计入，如果相同就跳过即可达到删除的效果。

11.不同的路径 II

"不同的路径" 的跟进问题：

现在考虑网格中有障碍物，那样将会有多少条不同的路径？

网格中的障碍和空位置分别用 1 和 0 来表示。



思路：

1.通过所有到达(i-1,j)这个点的路径往下走一步可到达(i,j), 这路径数总共有dp[i-1][j]条

通过所有到达(i,j-1)这个点的路径往右走一步可到达(i,j), 这路径数总共有dp[i][-1j]条

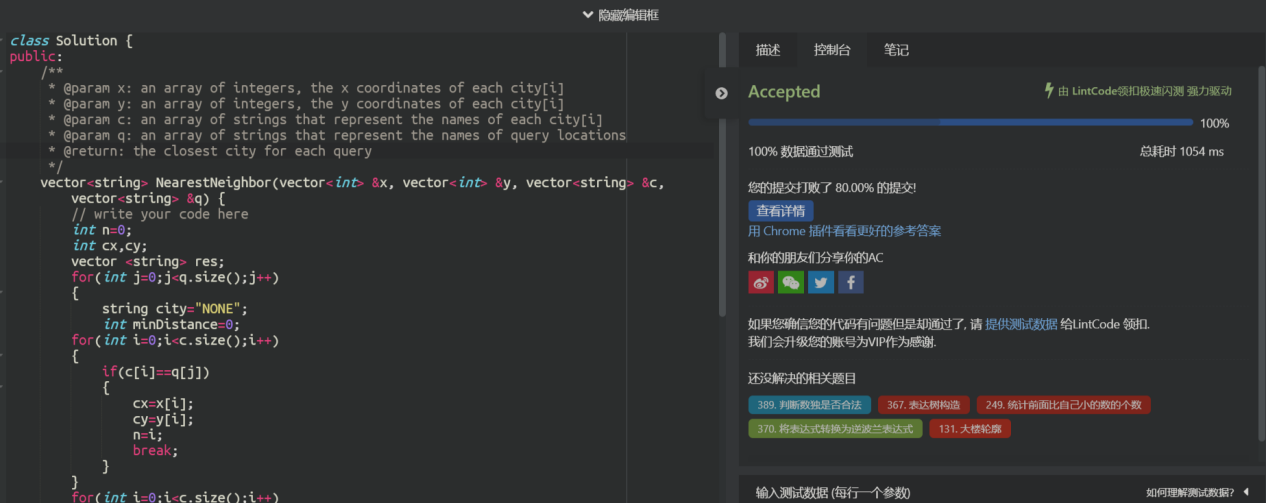
由此可以推出递推式dp[i][j] = dp[i-1][j]+dp[i][j-1]

2.遍历两次，算出每一个位置的路径数，设计遇到边界时特殊情况，以及遇到1是跳过

2.注意错误的数据，直接输出0；

12.最近的城市

在一个二维平面上有许多城市，所有的城市都有着自己的名字c[i]c[i]，以及位置坐标(x[i],y[i])(x[i],y[i])（都为整数）。有qq组询问，每组询问给出一个城市名字，你需要回答离该城市距离最近的，xx相同或者yy相同的城市名称。



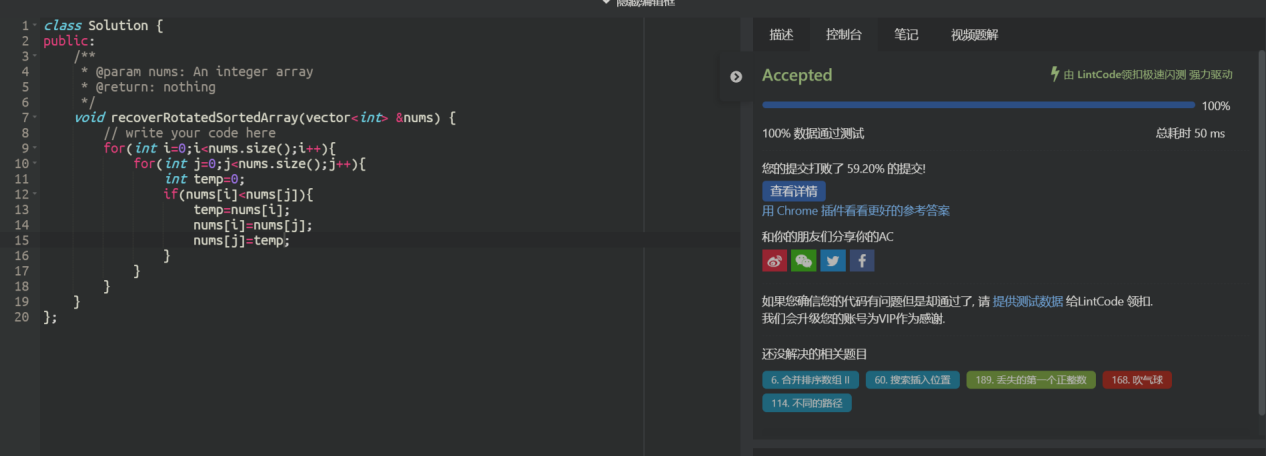
1.定义cx，cy用于记录当前城市的坐标。定义新容器res作为输出结果

2.第一次循环遍历所需城市，第二次循环定位到q中的城市在c中的位置，然后基于c操作

3.遍历c城市列表，如果找到x或y相同的城市，记录距离，如果有其他满足条件的，可以只保留最短距离的下标，如果无满足条件的城市则直接返回none

13.恢复旋转排序数组

给定一个旋转排序数组，在原地恢复其排序。

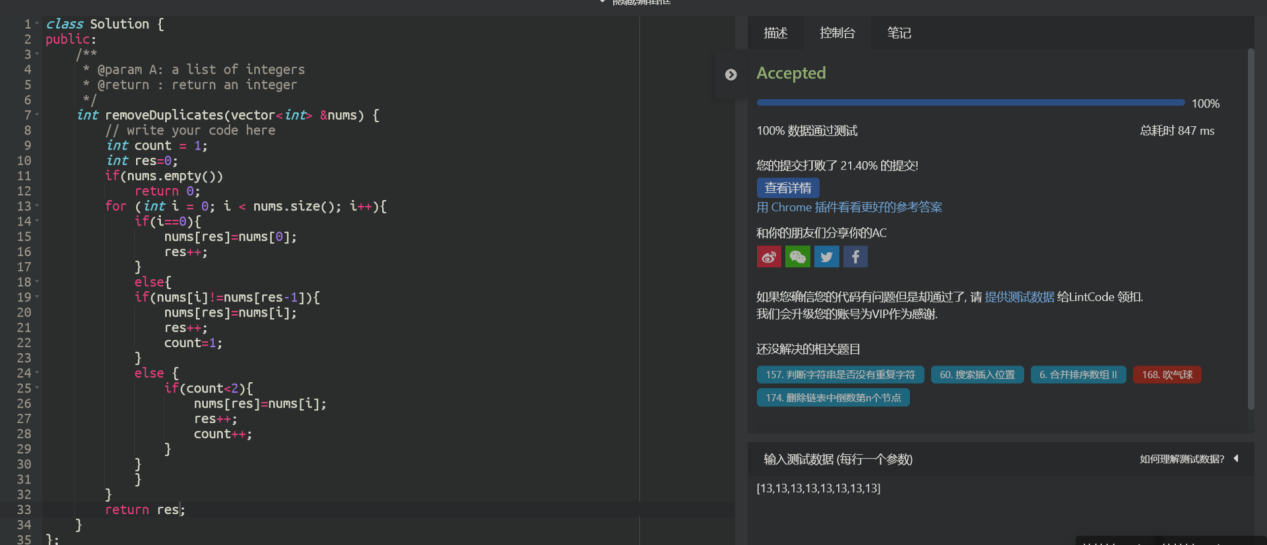


思路：

1.冒泡排序

14.删除排序数组中的重复数字 II

给你一个排序数组，删除其中的重复元素，使得每个数字最多出现两次，返回新的数组的长度。如果一个数字出现超过2次，则这个数字最后保留两个。



思路：

1.设置一个参量count记录同一个数字出现的次数默认为1，定义res为0，res为新数组的下标

2.在遍历中多加入一个判断是否是第一个数据，可以认为第一个数据一定不删除

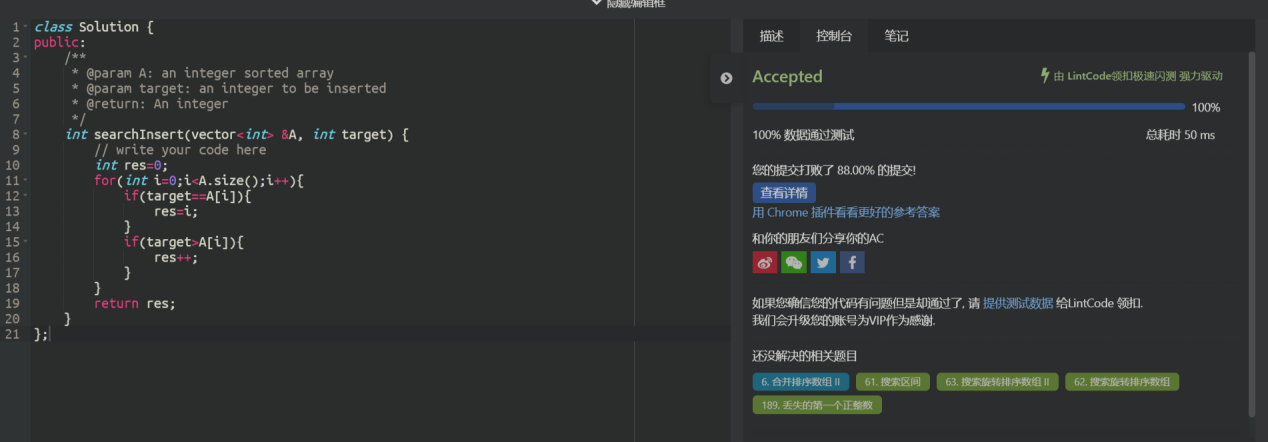
3.从1位开始，每一位数据与前面一位比较，如果不相等则记录到res的位置，res加一；

如果相等则分情况，count为1即为重复第二次的数，也记录到res的位置，res加一，否则无操作跳过

15.搜索插入位置

给定一个排序数组和一个目标值，如果在数组中找到目标值则返回索引。如果没有，返回到它将会被按顺序插入的位置。

你可以假设在数组中无重复元素。



思路：

1.定义res目标数

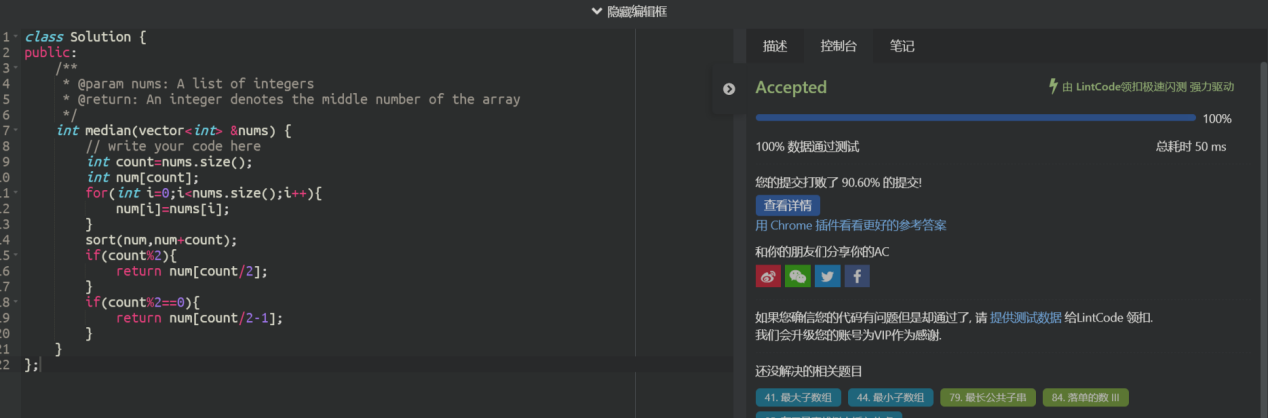
2.遍历数组如果与target相等则直接res为当前下标；如果小于target则res需要增加；

如果大于则无操作跳过，此时下标即为应插入的下标

16.中位数

给定一个未排序的整数数组，找到其中位数。

中位数是排序后数组的中间值，如果数组的个数是偶数个，则返回排序后数组的第N/2个数。



思路:

1.先设置一个相同的正常数组

2.对这个数组sort排序

3.在新数组中找中位数直接输出