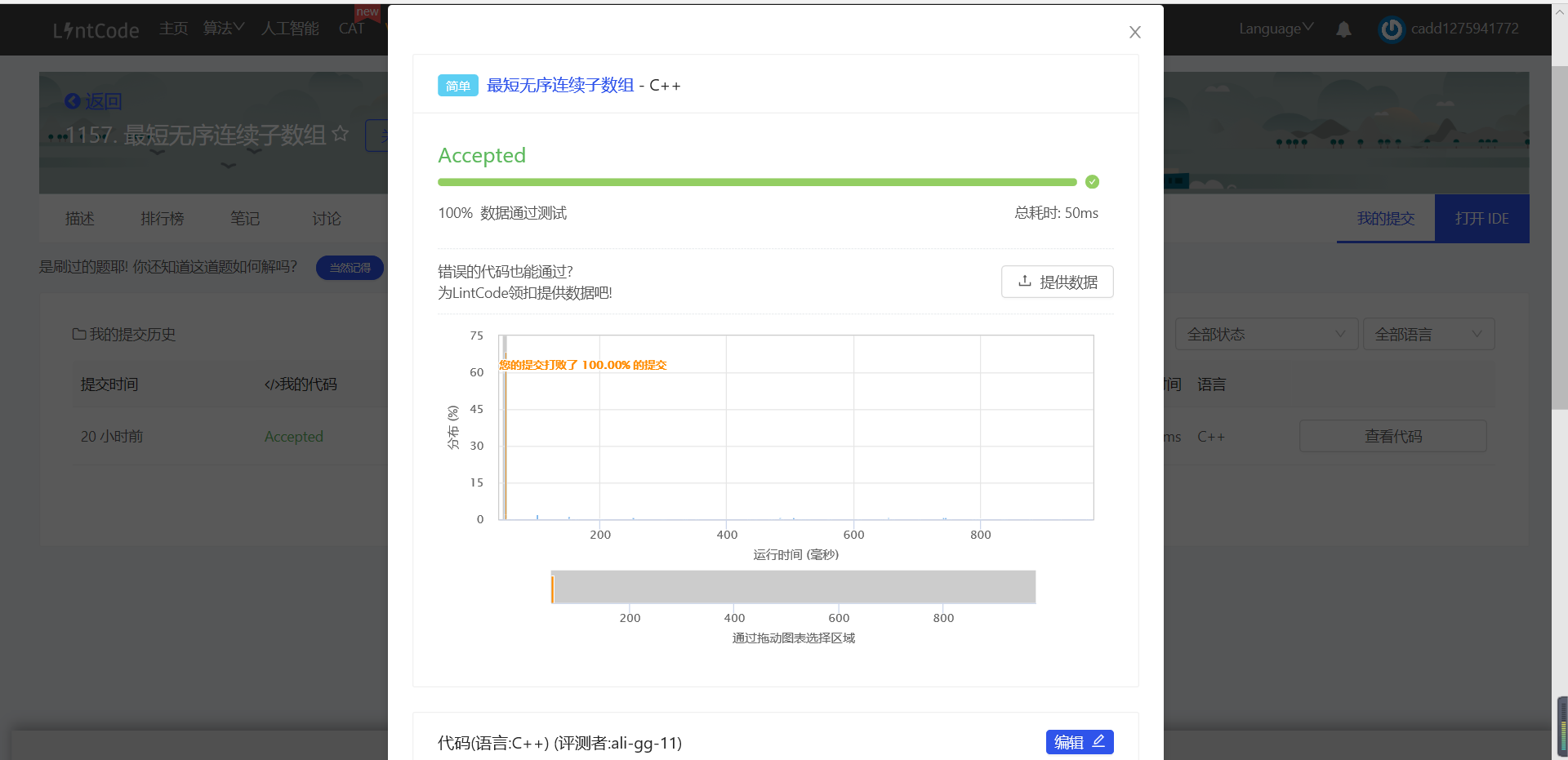
#最短无序连续子数组



思路：

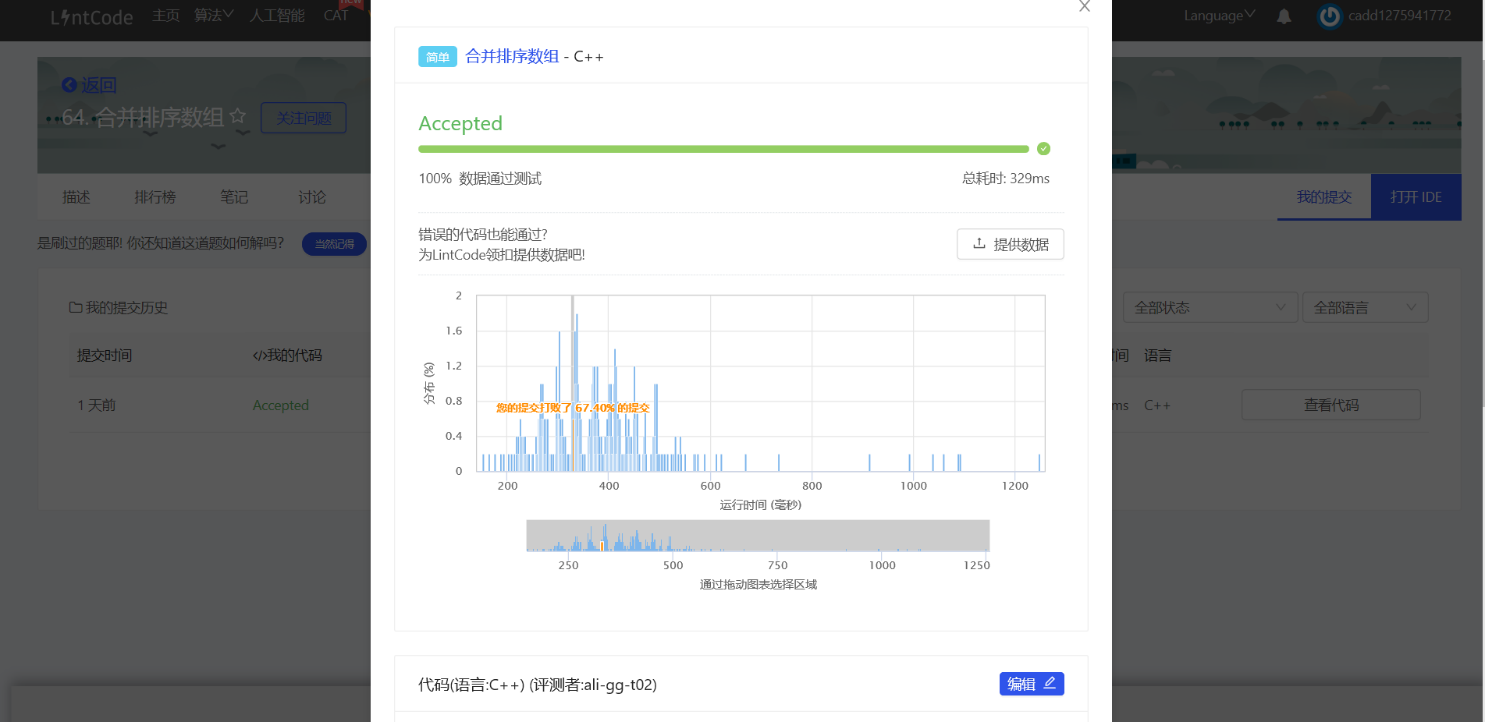
从左往右，找前i个元素的最大值，当之后的元素小于最大值时，该元素就是前i个元素的所求数组的右边界。当i增长到n-1时，也得到了问题所求的子数组的右边界。左边界也同理。

#重复计数



思路：  
用unordered\_map 遍历一遍

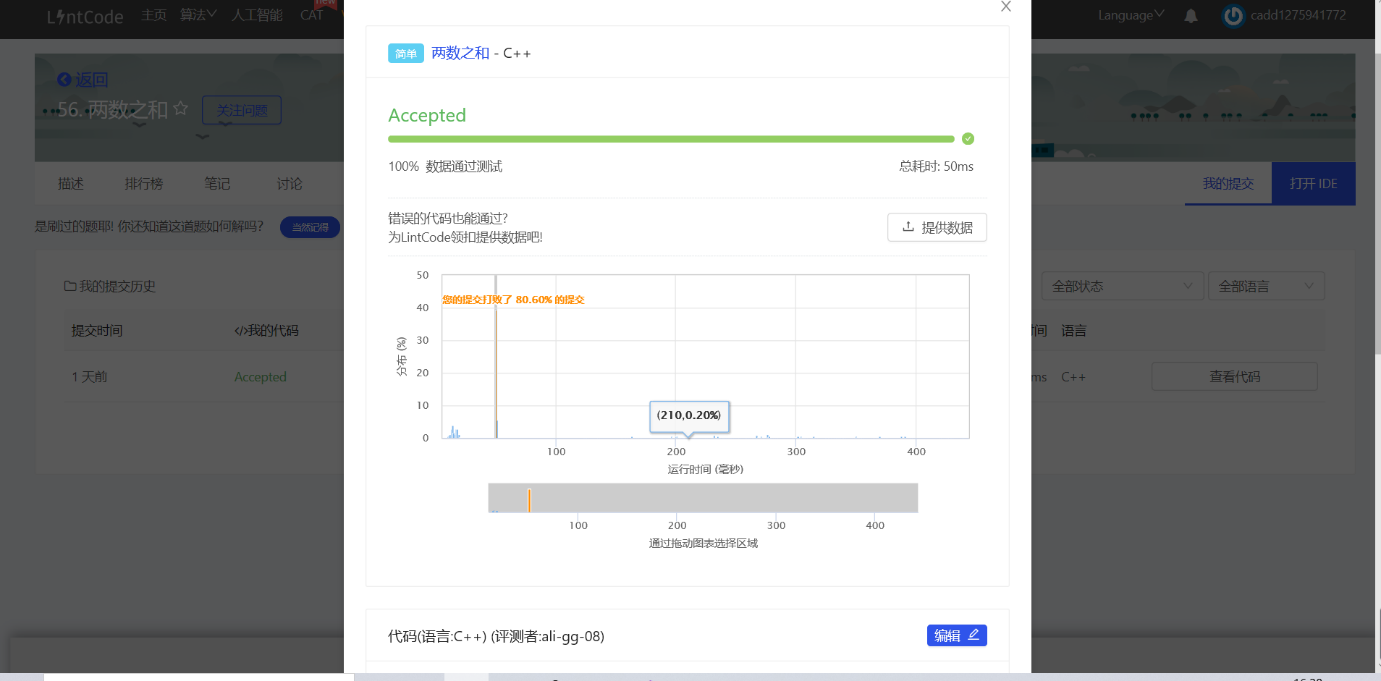
#合并排序数组



思路：

把最终的合并数组当成往A数组里添加B元素，将二者最大值比较，尾插到新数组

#两数之和



思路：

粗暴遍历法

#二分查找



思路：二分法遍历

#加一



思路：  
遍历

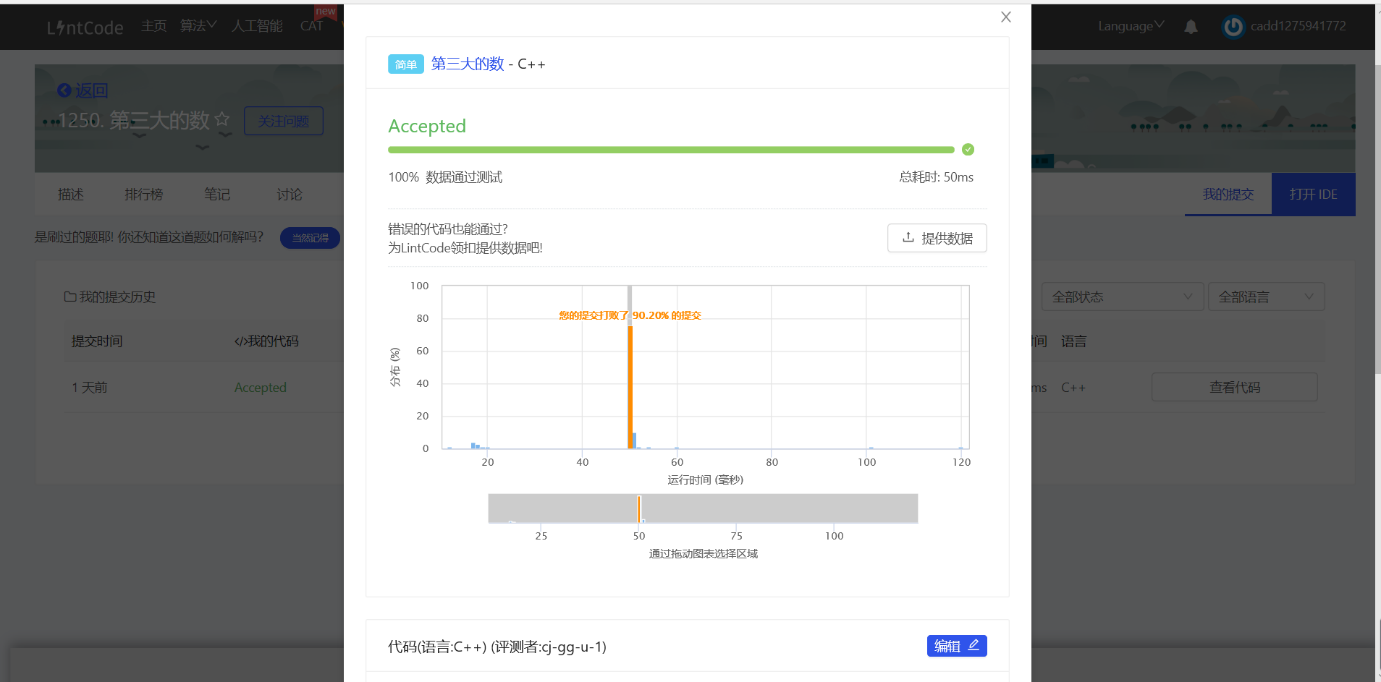
#删除元素



思路：

遍历将非目标元素存储并计数

#第三大的数



思路：  
先排序，然后删除前面的数，保留三个元素，返回第一个元素；若不存在第三大的数，排序完后，直接返回末尾值

#最长上升连续子序列



思路：

当长度为0，返回0；当长度为1，返回1；准备俩个全为1元素的（长度和题目数组长度相同）数组，分别从左往右和从右往左遍历，当满足升序则在1数组加1，得到最后的俩个数组，取数组中的最大值，最后比较返回俩个数组中最大的值

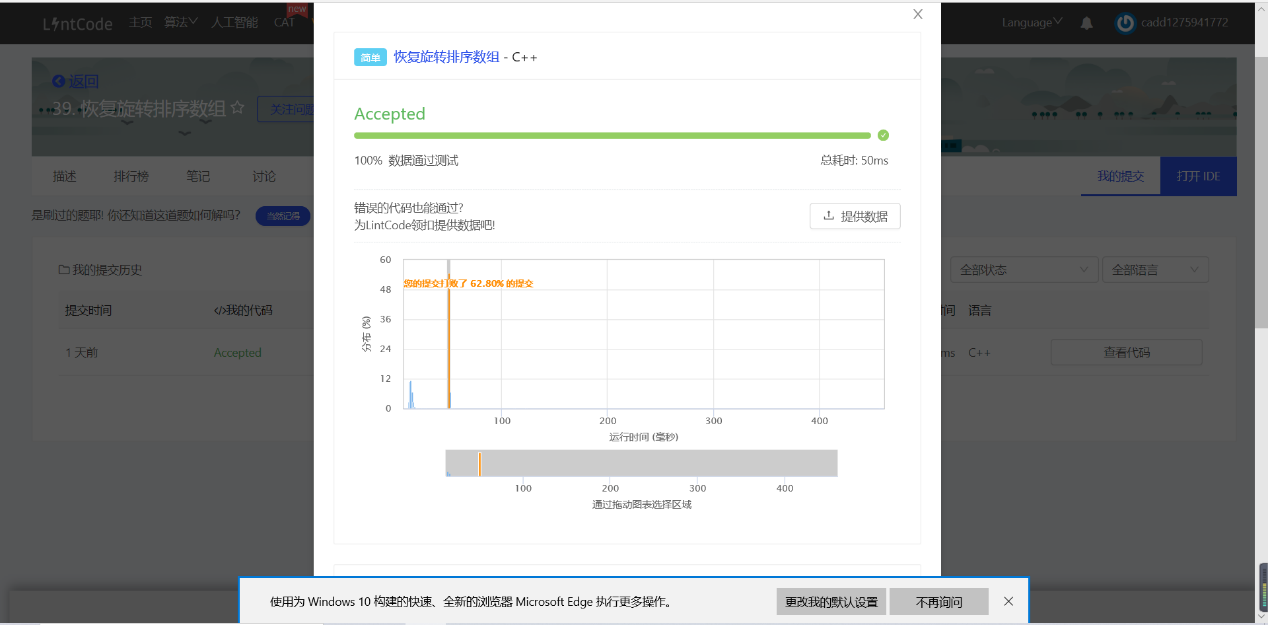
#删除排序数组中的重复数字



思路：

遍历，将重复元素删掉，最后返回剩下的数组长度值

#恢复旋转排序数组



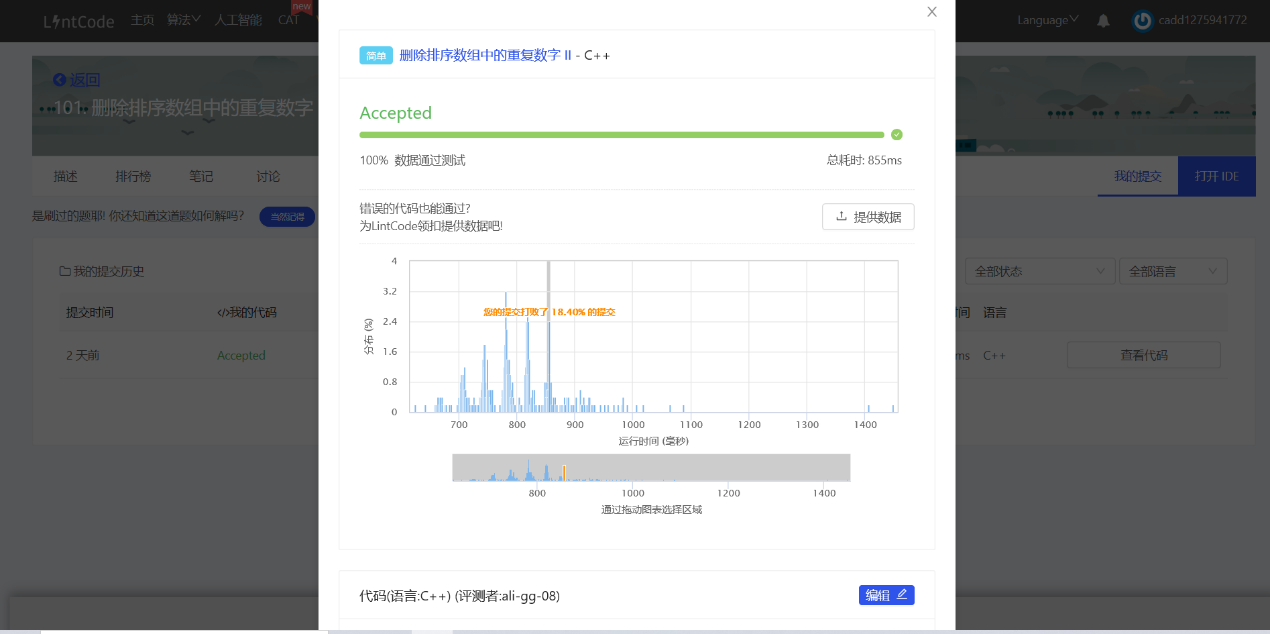
思路：  
三步翻转法：以{4,5,6,7,1,2,3}为例

1.  先找到1的位置.然后翻转{4,5,6,7}得到{7,6,5,4}

2. 翻转{1,2,3}得到{3,2,1}

3. 此时数组为：{7,6,5,4,3,2,1}, 将其翻转即得{1,2,3,4,5,6,7}

#删除排序数组中的重复数字ll



思路：  
在原数组中，若不是相同元素或相同元素只有2个则直接拷贝，若相同元素超过2个，则按原顺序将超出的位置省去，后面跟上不相同元素，最终返回数组长度

#不同的路径ll



思路：

动态规划

定义dp[i][j]为到达(i,j）的不同路径数。

如果(i,j) 处有障碍，则dp[i][j] = 0,即没有路径

如果第0行 j 列的路径，则左移一个，即 dp[0][j] =dp[0][j-1]

如果第 i 行0 列的路径，则上移一个，即 dp[i][0] =dp[i-1][0]

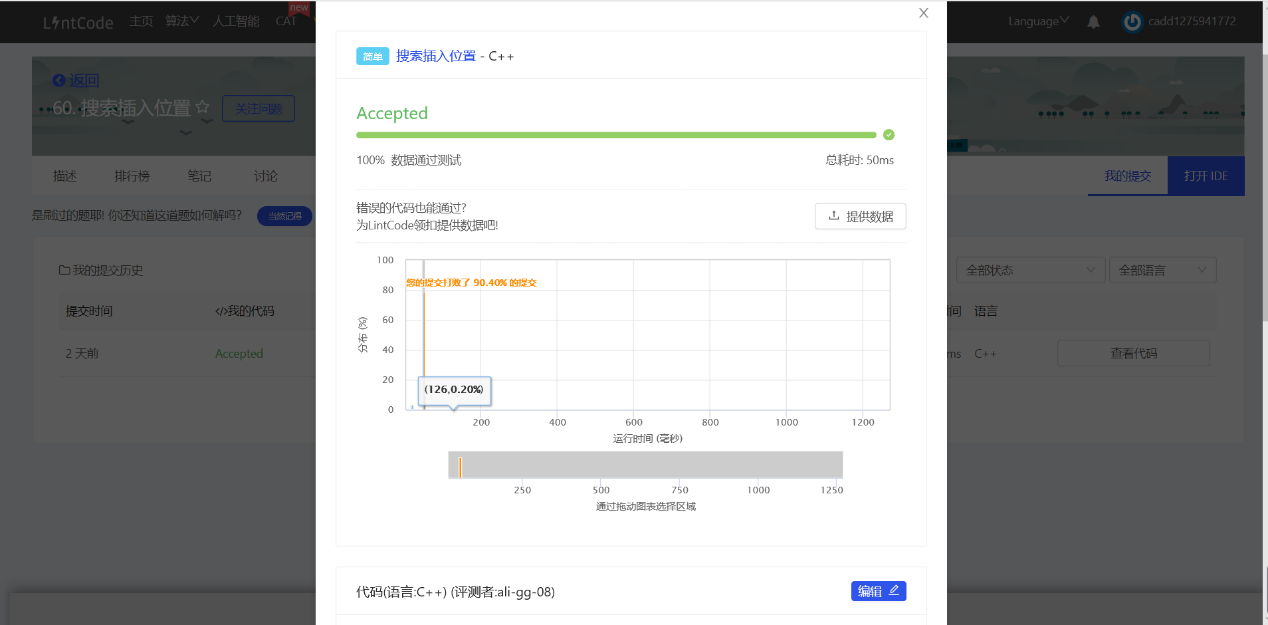
否则，dp[i][j] = dp[i-1][j] + dp[i][j-1]

#最佳利用率



思路：遍历A中每个数，在B中做二分检索，找最接近的数，返回索引值

#搜索插入位置



思路：  
二分查找锁定其大致俩位差位置，然后和附近俩位比较，锁定准确位置，超出则直接返回数组长度值即为索引号

#中位数



思路：

使用优先队列priority\_queue将所有数排序，再依次取出第N/2个数。

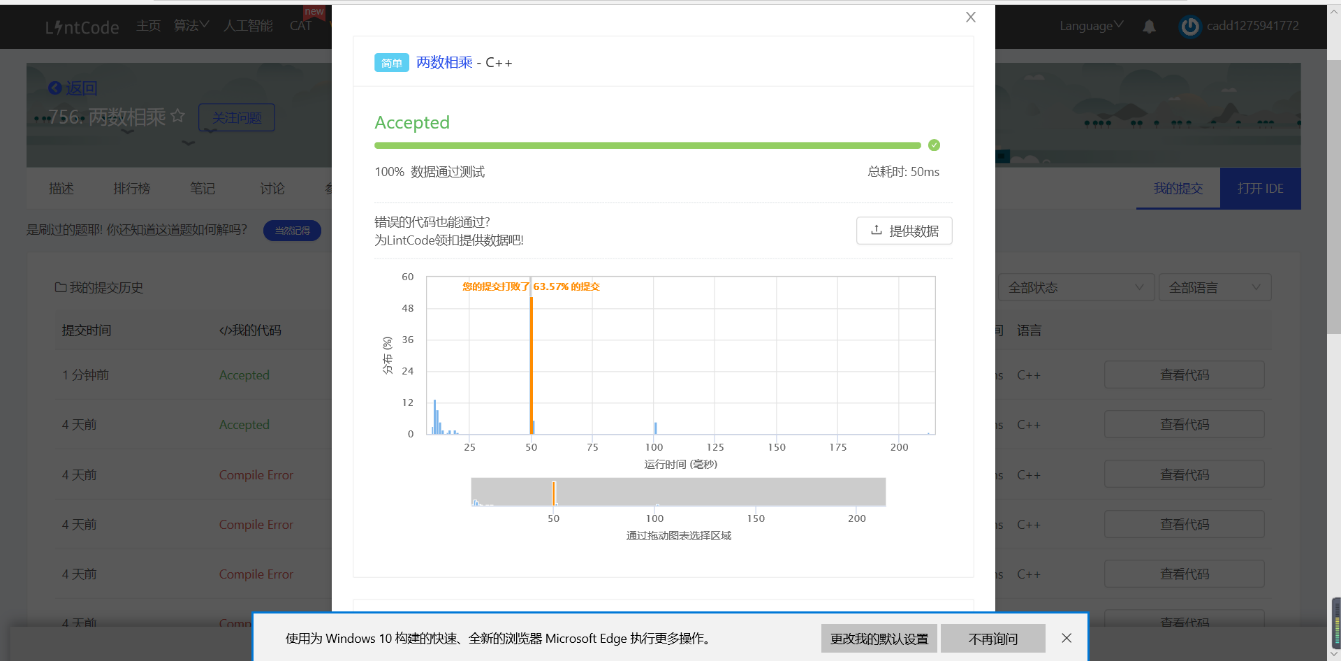
#合并区间



思路：

首先将数组中的区间元素按照左边界大小由小到大排序，然后根据此区间的左边界和上一区间右边界比较大小，来判断是否合并区间，若无重叠直接pushback最后把所得区间输出

#两数相乘



思路：

遍历出链表中的数字组成一个值，然后再相乘

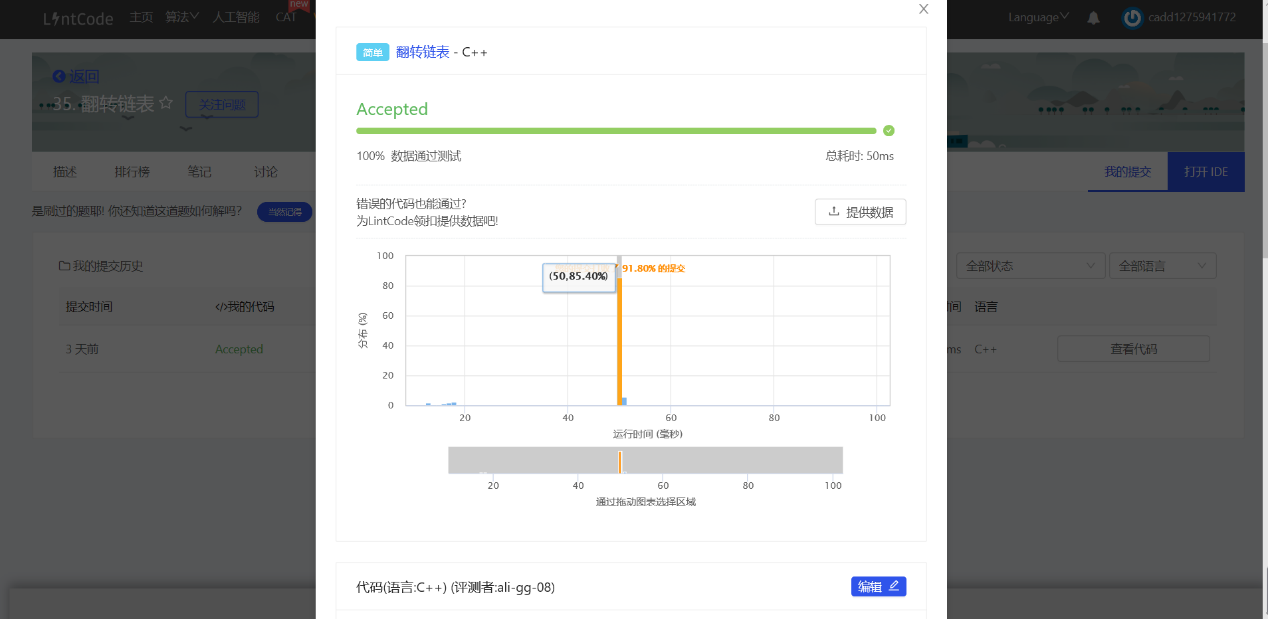
#链表的中间结点



思路：

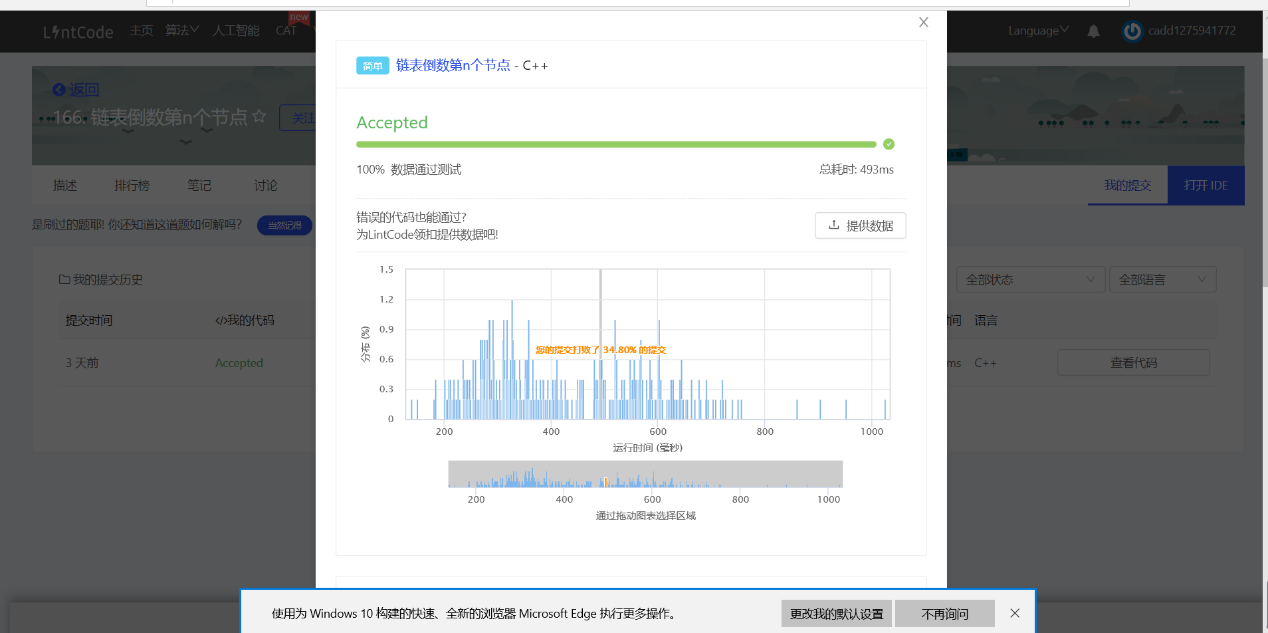
对链表进行遍历，同时将遍历到的元素依次放入数组 A 中（倒序）。如果我们遍历到了 N 个元素，那么链表以及数组的长度也为 N，对应的中间节点即为 A[N/2]。

#翻转链表



思路：  
让第一个节点指向null，然后将目标指针指向的二个节点，再让第二个节点指向原第一个节点，循环

#链表倒数第n个节点



思路：

首先计数，算出总节点数k，然后倒数第n个节点就是正数第k-n+1个数，返回即可

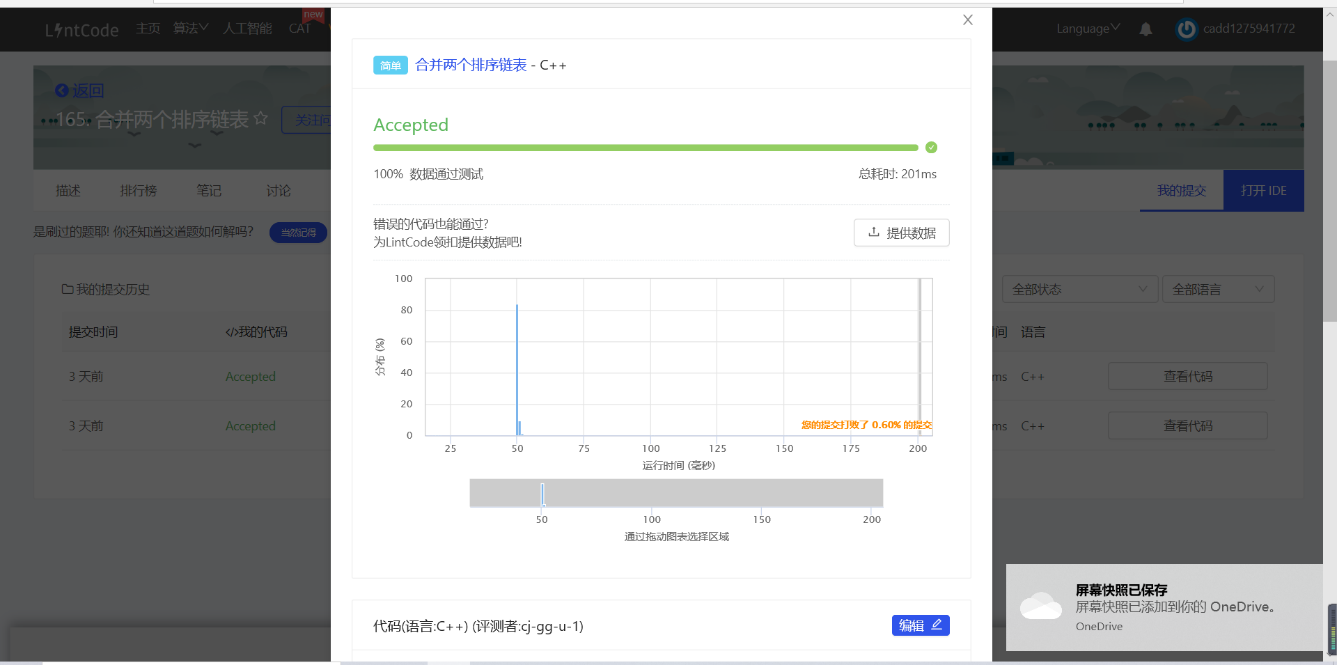
#删除链表中倒数第n个节点



思路：

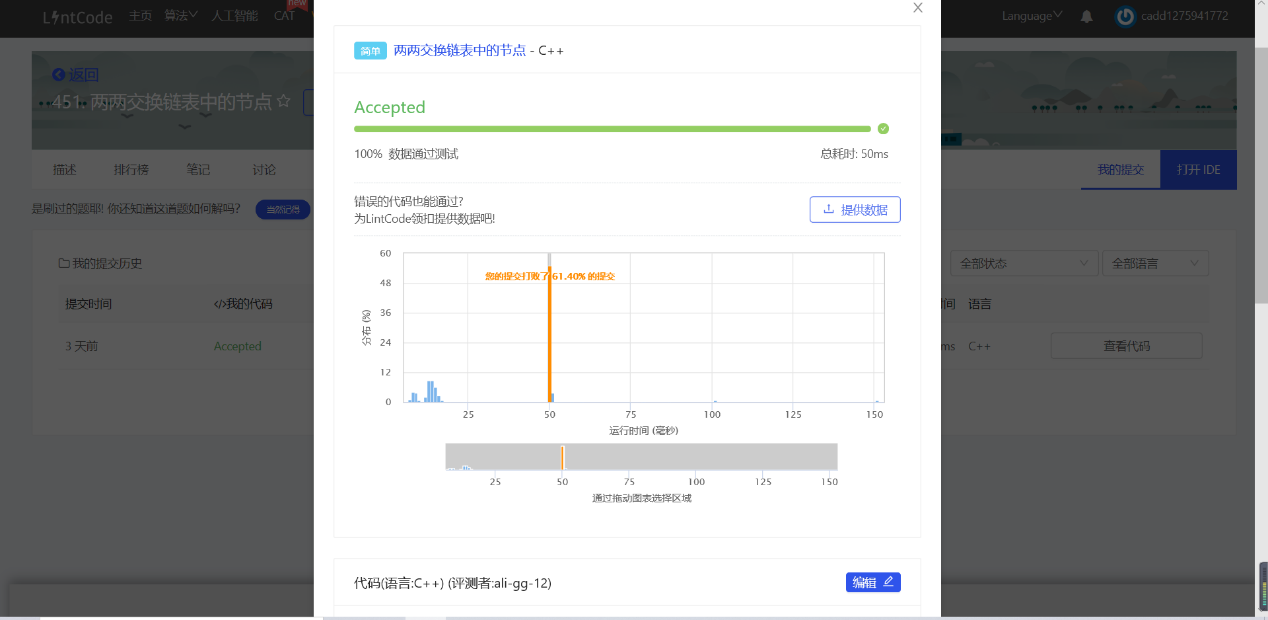
用快慢俩个指针，倒数第n个节点，则让快指针从head开始正着走n下，若结果快指针为null，则删除第一个节点，head指向下一个，此节点则为新链表头节点；否则则说明删除节点不为head，直接返回head就是新链表头节点，慢指针下一位即为要删除节点

#合并两个排序链表



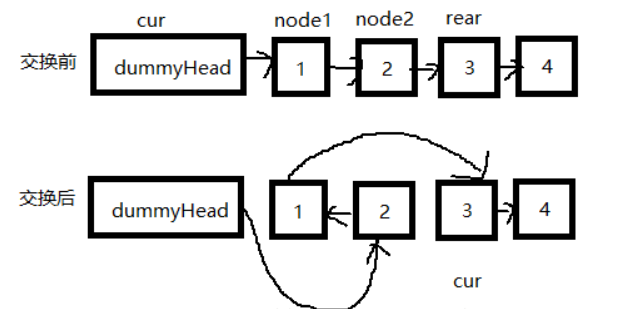
思路：  
若其中一个为空，则直接返回另一个链表；递归法合并

#两两交换链表中的节点

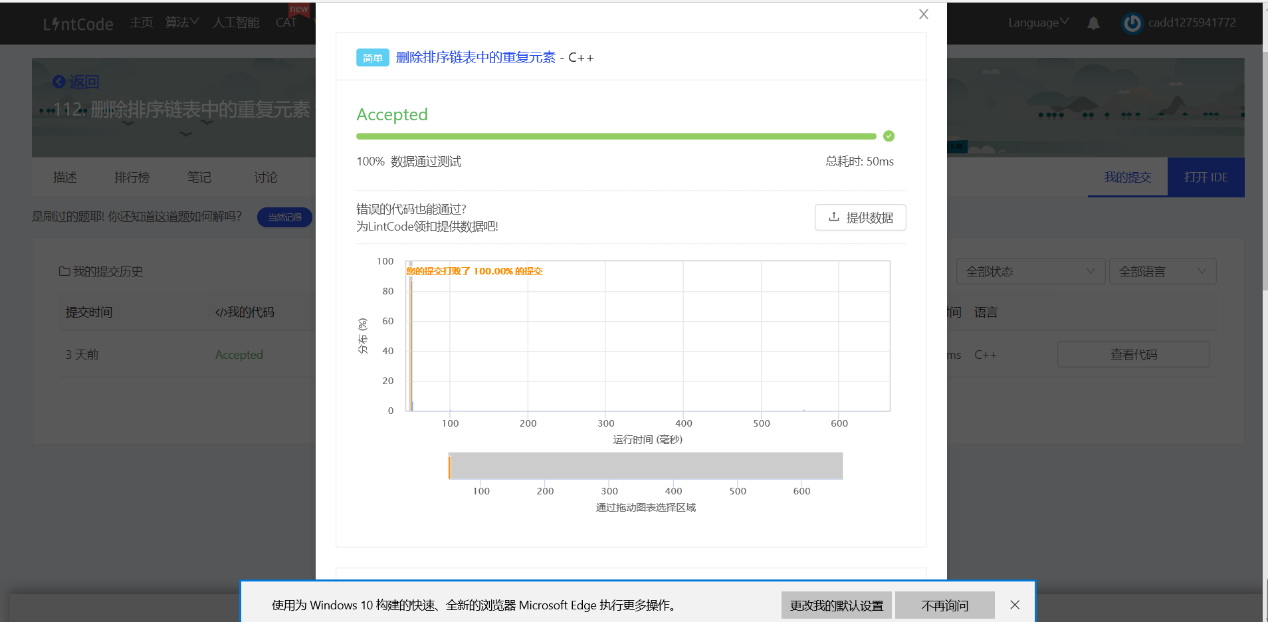


思路：

建立虚拟头结点（为了不对head进行特殊处理），按照如下示意图进行节点交换。



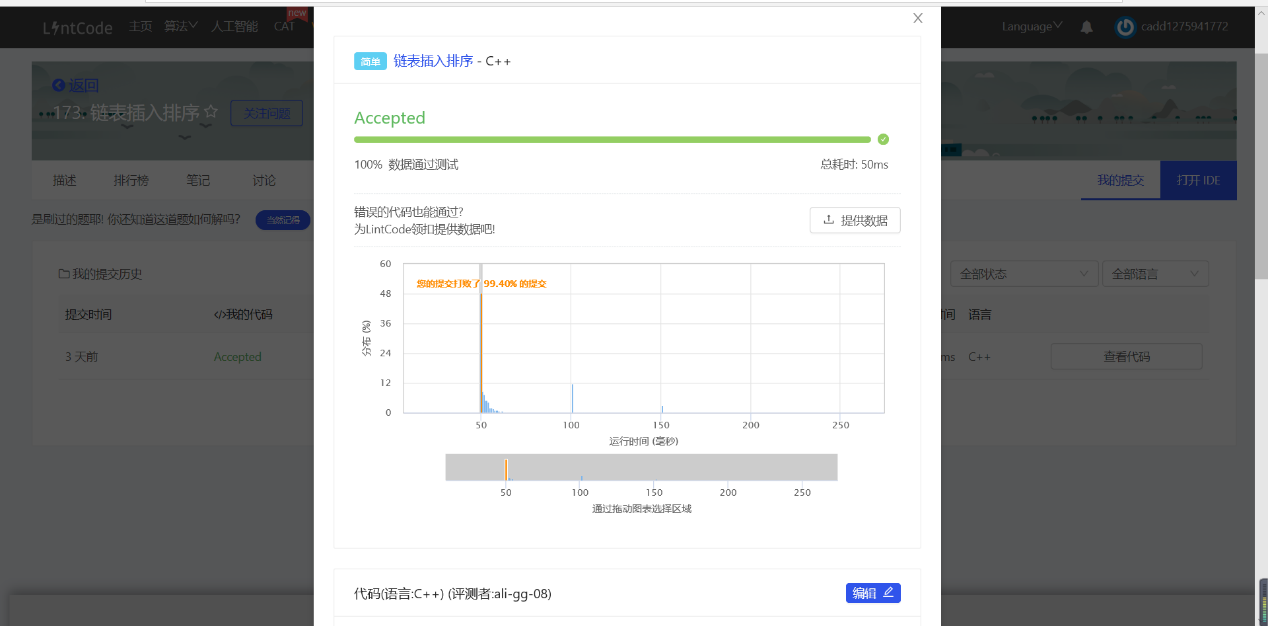
#删除排序链表中的重复元素



思路：  
首先判断链表是否为空，如果所给链表为空直接返回，否则在执行ptr->next这部操作时会因为ptr为NULL而造成执行时错误，遍历首个元素到第倒数第二个元素，因为是逐个向后比较，最后一个元素会被比较到，若相同则直接

ptr->next=ptr->next->next

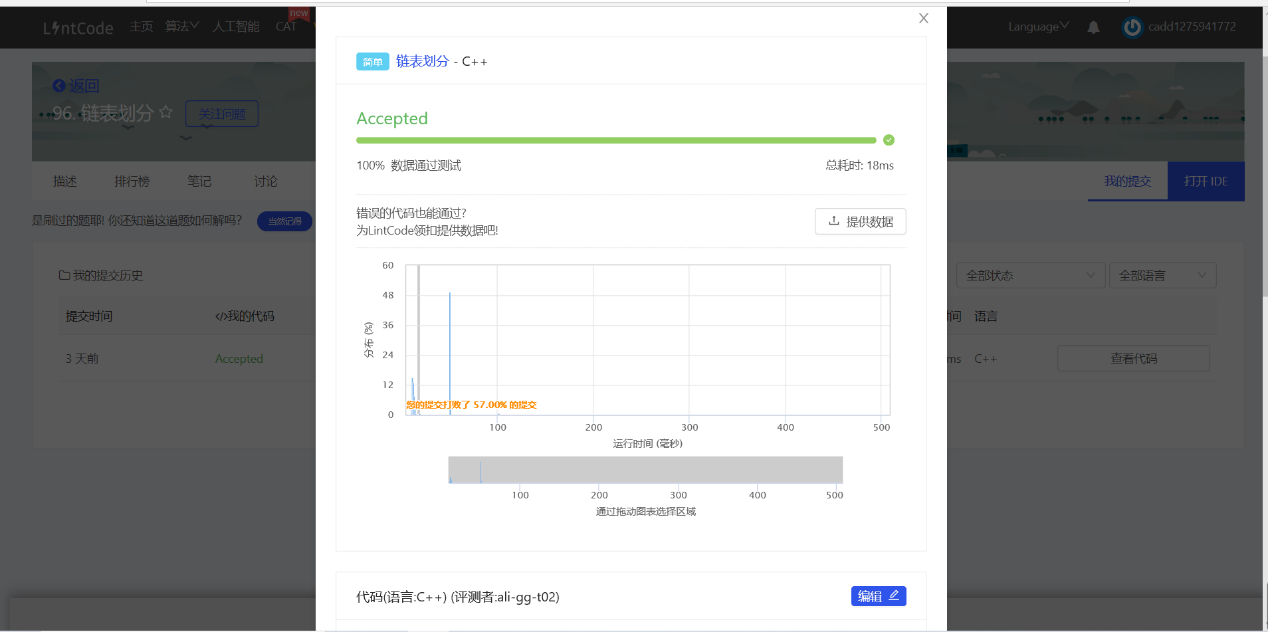
#链表插入元素



思路：创建一个虚拟头结点，让虚拟头结点指向head，

tail指向已排好序部分的尾结点；sort是下一个待排序的结点定义一个pos指针，用于每次从头（dummy处）寻找该插入sort的结点的位置，将pos定位到已排好序的且比sort小的部分中的最大的那个结点，断链比如dummy->4->2->1->3，tail是4，sort是2，则让4->1，把2拎出来。 pos此时在dummy处（因为4大于2），则让 2->4。上一步相当于断掉了dummy->4，现在重新接起来：dummy->2->4->1->3。让sort重新回到下一个待排序的结点处

#链表划分



思路：

创建俩个虚拟链表节点，将原链表比x小的节点放在lesshead，大于等于x的放在morehead，最后将两链表拼接

#链表求和



思路：

当俩个链表节点不为空时，则让俩节点依次相加，和取10的余数作为虚拟节点下一位，依次得到结果

#删除链表中的元素



思路：

若链表为空，则直接返回；将链表从第二个数比较，相同则直接让上一节点指向下下节点；最后判断第一个节点，若是目标值则返回下一节点