

暑期算法训练报告

专业班级： 11J192

学 号： 20191000739

姓 名： 朱文灿

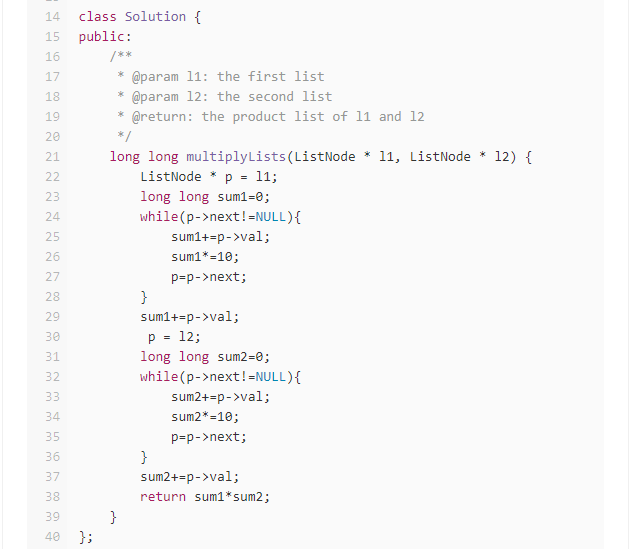
指导老师： 张剑波

**中国地质大学地理与信息工程学院**

**2020年 7 月**

1）链表

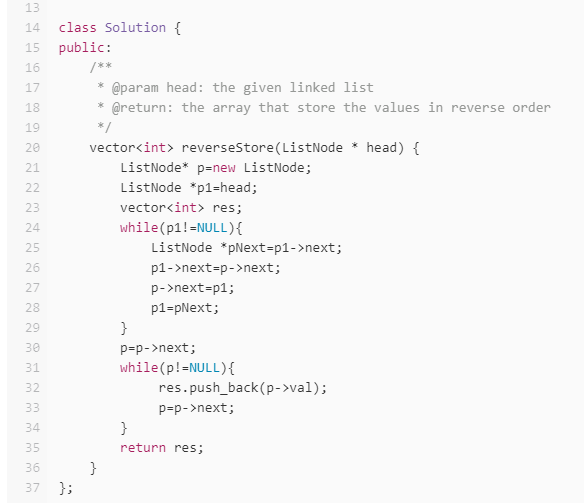
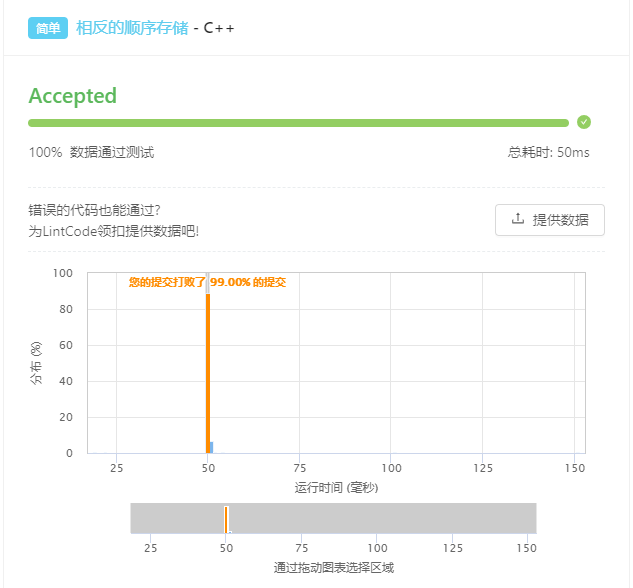
756.两数相乘



指针p指向链表1时，循环语句直至尾节点的前一个节点，循环内先加节点数据再乘10，至循环结束时为\*\*\*0，然后再把尾节点前一个节点的数据加上去，即为所求的数。之后指针p指向链表2，执行相同的操作，最后返回两数相乘的结果。

822.相反的顺序存储/35.链表翻转

这两题基本相同，唯一的区别在于一个返回vector，一个返回ListNode，算法都是一样的。

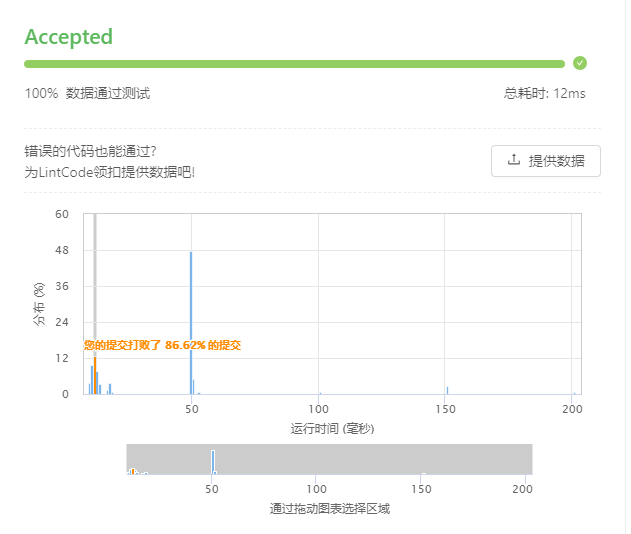


开始创建一个新的ListNode对象，如果new一个ListNode，会有一个无用的头节点，里面val的值为0，如果用malloc则似乎是一个随机值。

之后定位原链表的后一个节点。后一个节点断开，连到新链表第二个节点

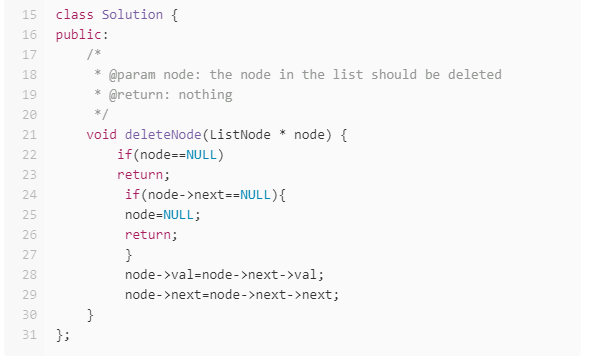
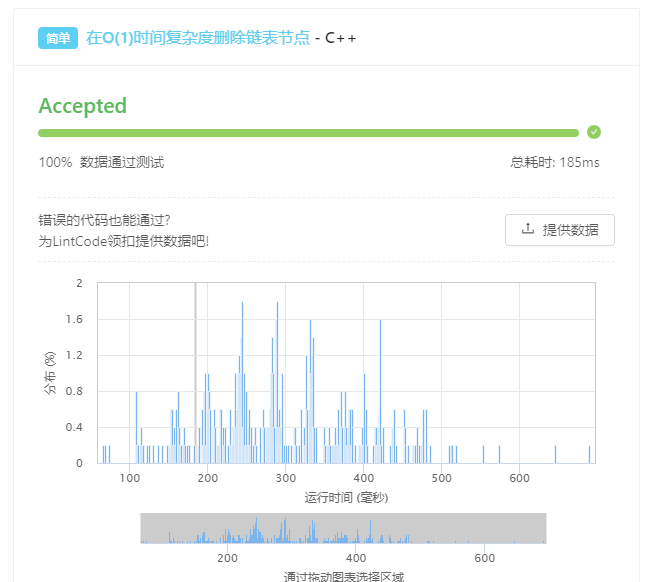
接下来新链表第二个节点变成此时的head，而head轮换到下一个。下一个节点轮番放置到新链表前一个节点之前，变成新的"头“。删除多余的表头后，这样一个由右往左传递的新链表即为翻转后的链表。

1609. 链表的中间结点



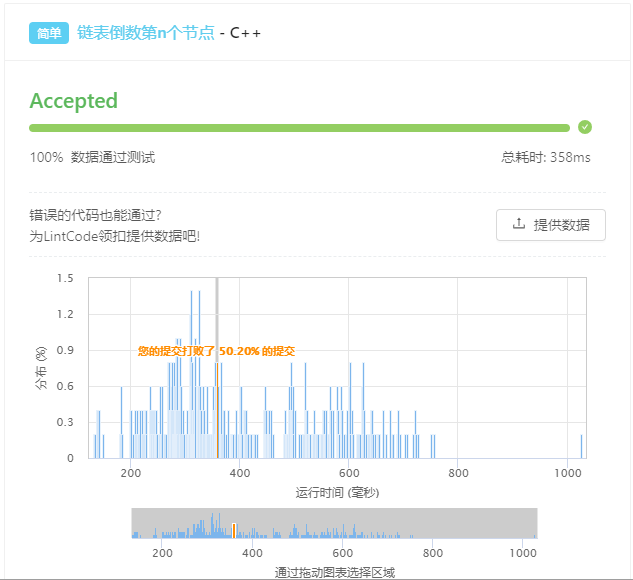
先遍历计算节点数，然后奇数个以5为例，往前推动了2个节点，即为（5-1)/2，偶数个则以6为例，往前推动6/2个节点。以for循环来推动节点，之后返回新链表。

372. 在O(1)时间复杂度删除链表节点



删除节点即为将该节点的数据变为下一个节点的数据，并且指向改成下一个结点的指向，这样其本身的指向被更改，他原来的下一个节点就失去了前面的指向。两个if为特殊情况下的做法，如果链表为空，不需删除；如果链表只有一个值，将这一个值改成NULL。

166. 链表倒数第n个节点



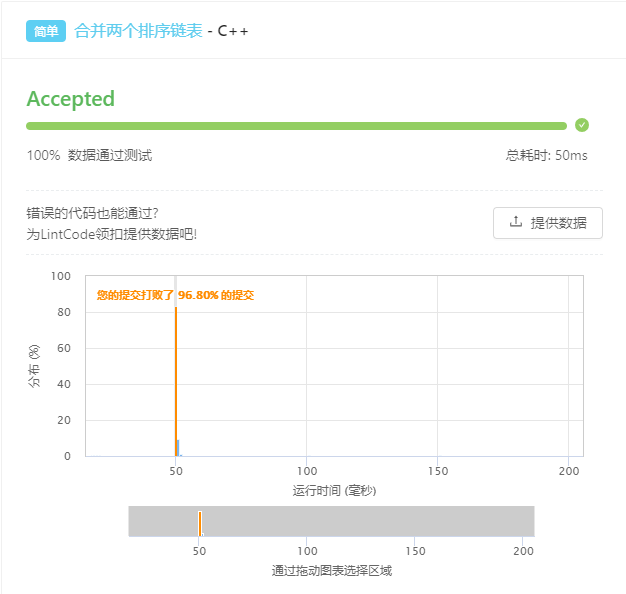
先遍历一遍计算出总节点数量，然后如果是第三个节点，那么就需要移动两次，值等于count-n，用for循环移动一下，得到结果。

492. 队列维护



enqueue函数相当于链表的尾添加，在末端增加新节点。dequeue就是得出链表头的值，链表往后移一位。多个enqueue执行，会使lists成为一个有多个值的链表，liste则是只有一个值。开始的时候，lists=liste=p这一句写的是lists=p=liste，这样的话lists内只有一个值，分析了一下应该是由于lists和liste是全局变量，连等又是由右往左计算，lists=p的话指针没有指到正确位置。

165. 合并两个排序链表



设置了一个p3为合并后的链表，p4为链表p3的下个节点指向处，在循环里面判断目前l1和l2两个链表哪个值更小，选它接在后面。最后如果一个链表为空，而另一个不为空，则将不为空的接在p3之后。

451. 两两交换链表中的节点



一开始想的是1指向3，2指向1，之后因为要再换位，所以1指向4，4再指向3。由于本身一次是换两个，后面3和4应该在第二次循环内换。

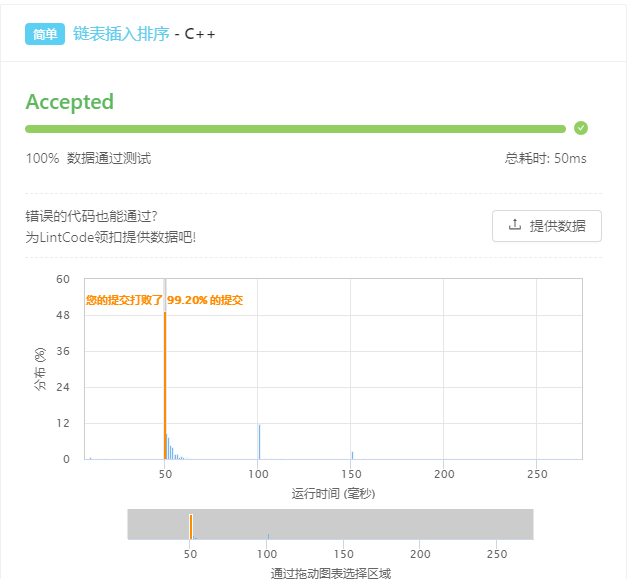
所以第二次改成1指向3，2指向1。这里，因为一开始p1直接等于的是head，之后第一次出现12换位的链表时，因为头节点被替换掉，导致head里面少了一个地址，与换位后的链表衔接不上。（比如head一开始为1234，第一次循环p1=2134, temp=134, 如果temp=p1, p和head依然是134不变。但如果p一开始为01234，p1就为02134，temp=0134，temp=p1,next地址被换了，p也就同时变了。）所以要设置一个无用的链表头，防止原链表头被换掉。

112. 删除排序链表中的重复元素



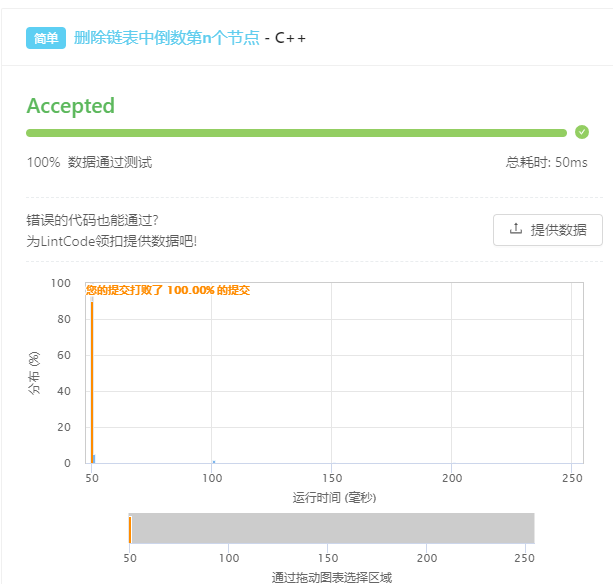
因为是排序列表，只需关注前后相邻节点的值。两个特殊情况：1.当前已到达链表末端，返回。2.原本有不止2个重复值，切换后依然存在重复，这时p指针就不需要动了。

173. 链表插入排序



设置两个活动指针，一个每次从头开始遍历排序好的链表，一个逐次遍历后面的无序链表。如果有序链表中有相较更大的值，无序链表中的值被放置在这个值前再接回去。如果一个更大的值都没有，那么p->next为null，p本身被接到有序链表最后处。最后返回去掉头节点的排序链表。

174. 删除链表中倒数第n个节点

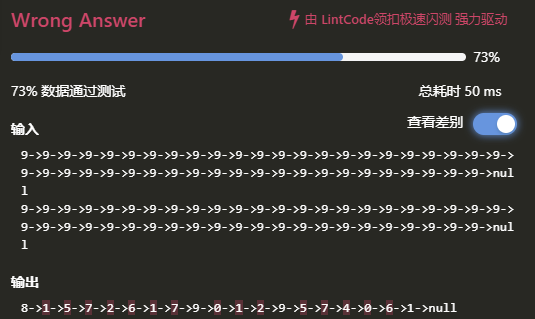
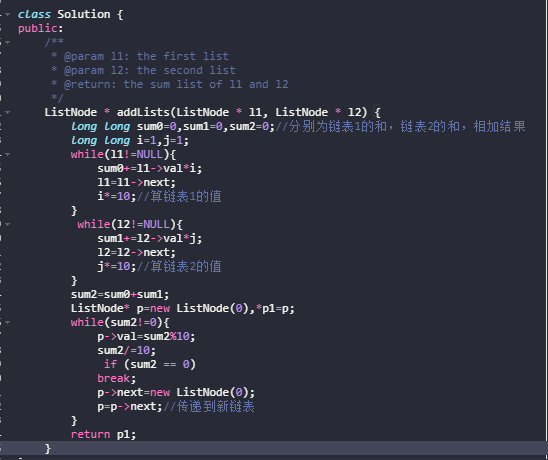


设置两个指针，前指针指到第n个节点位置时，后指针开始动，循环到前指针后第二个节点为空，此时后指针落在要删除节点的前一个，然后删除。

如果要删除第一个节点，在第一个循环后，前指针会到达最后一个节点上，前指针的后两个为null，所以直接返回head的第二个节点来删除。

167. 链表求和

这一题，一开始就是用最单纯的思路来写，算出两个链表的总和值再一相加，然后再把相加的值一个个解出来，放到新链表。后来简单的测试用例也都能过，直到遇见了一个有四十几个节点的链表，int类型作为和放不下了，改成long long也只能支持到2^63-1的大小，因为达到了这个上限，所以一开始的做法进行不下去了。



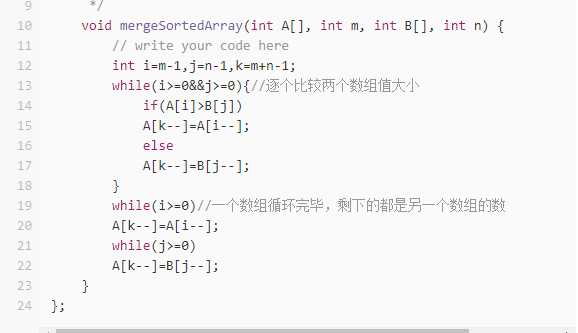
下面开始用新的做法重写。



所以这实际上类似于高精度的加法，需要有一个carry进位来保存，这里用sum来充当这个作用。链表是算一位填一位，而不是之前算出个总的再一位位放进去。

2）数组

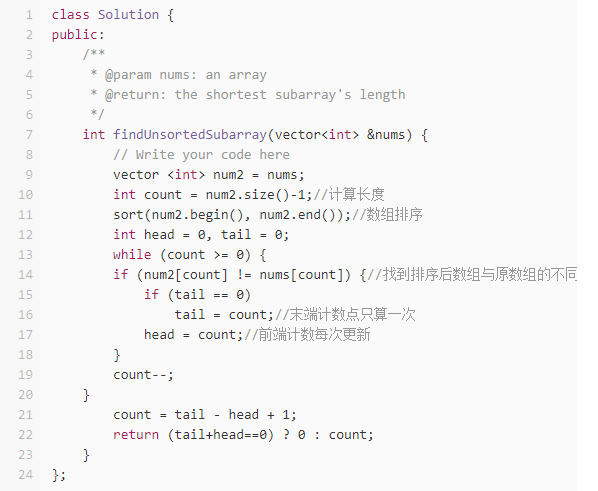
64. 合并排序数组



一个个数组内的数来对比，对比完被放入的那一个数组往前移。当一个数组遍历完的时候，它的计数值（i或者j等于-1），剩下的一个数组一个个往前填。

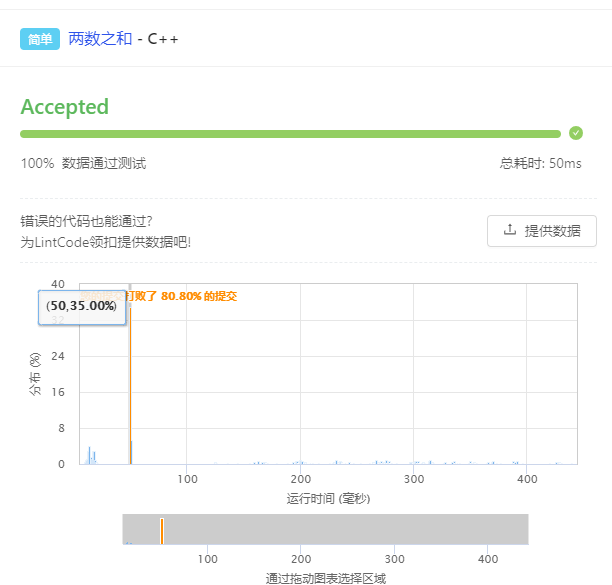
这里第一次是从前往后填，在由数组A变动到数组B之后，数组A内会出现一个原本没有的数，而从后往前填却没有这种情况。

1157. 最短无序连续子数组



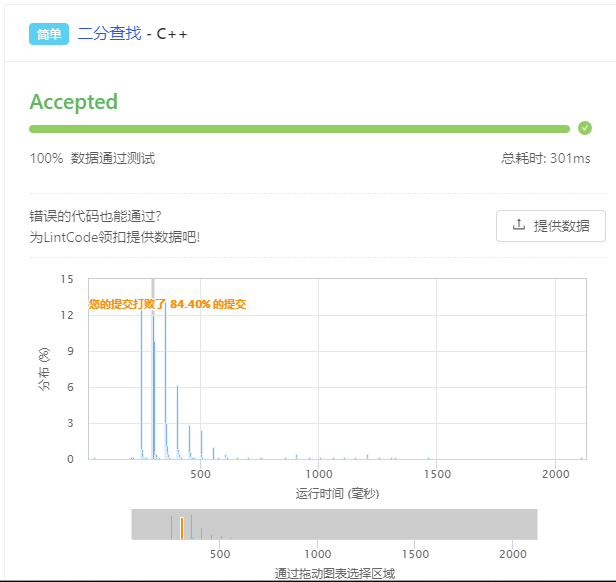
如果存在一个无序连续子数组，那么它的头尾都与升序的数组不同。所以先排序出一个新的数组，然后对比出最前面和最后面两个点，一减就得到结果了。

56. 两数之和



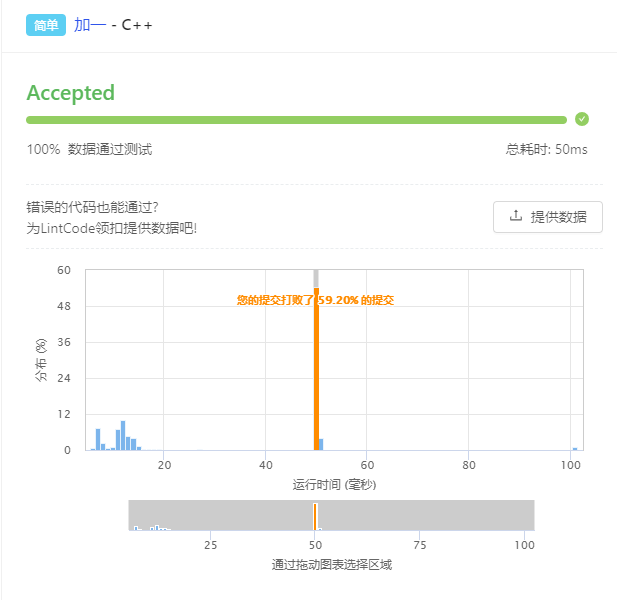
连续两次循环找到对应的下标，缺点在于时间复杂度为O(n^2),如果使用哈希可以使得时间复杂度为O(n),还需学习。

14. 二分查找



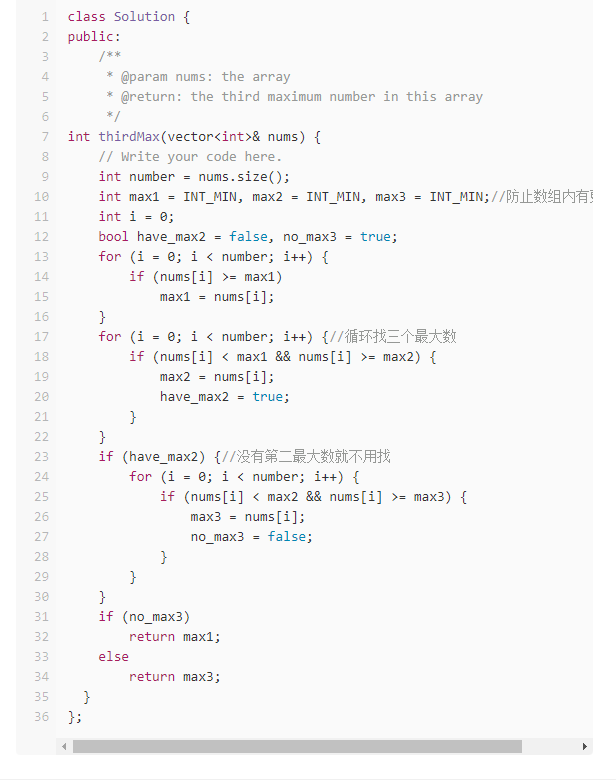
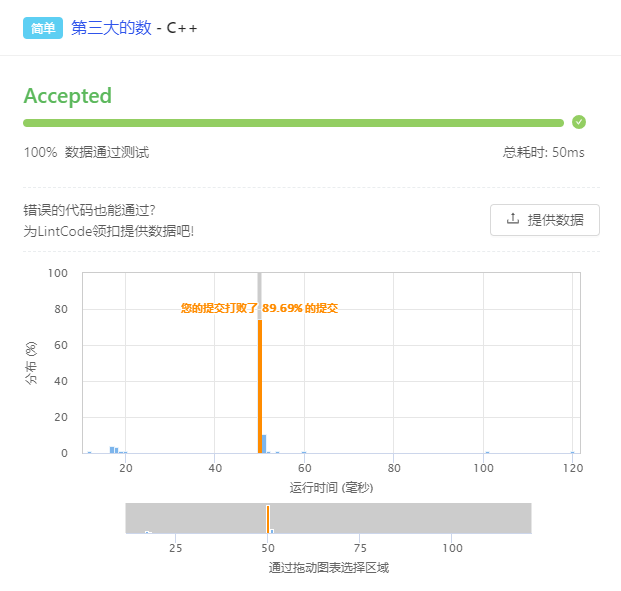
左右两边下标加起来\2获取中点下标，如果没找到的话取中点两边的下标，如果h\_number比mid+1小，就会死循环；t\_number同理。另外如果不用二分查找，逐个对比的话，会因为运行时间太长不给通过。

407. 加一



最低位不是9操作最简单，否则得最低位置0，然后考虑前一位。如果全是9的话，carry到最后会是1，那么最高位变1，后面再多个0。如果只有1个9，因为访问不到digits[-1]，所以多列一种情况。

1250. 第三大的数



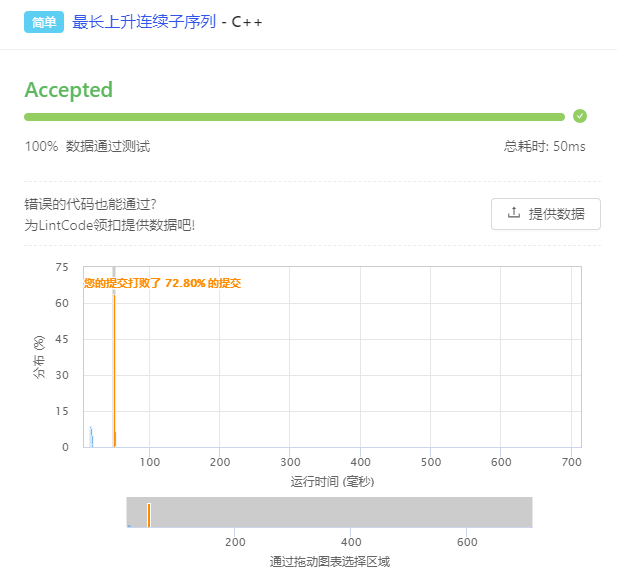
分别找出第一大，第二大，第三大的数再输出。如果不加bool类型的判断，那么如果数组内含有INT\_MIN的值，在最后判断的时候会导致输出max1，所以要加上no\_max3这个数据。

172. 删除元素



把前面的n位替换成应有的输出，实际上数组并没有做到”删除“，面板里看到的数据是输出的数组A的前n位。

397. 最长上升连续子序列



这题关键在于如果有连续中的序列中断，比如说从升序变成降序，那么要重新开始计数。所以在循环最开始加上判断语句求个最大值，简化一点可以写个求最大值函数。另外无数据的时候A[1]无意义，加个特殊情况。

100. 删除排序数组中的重复数字



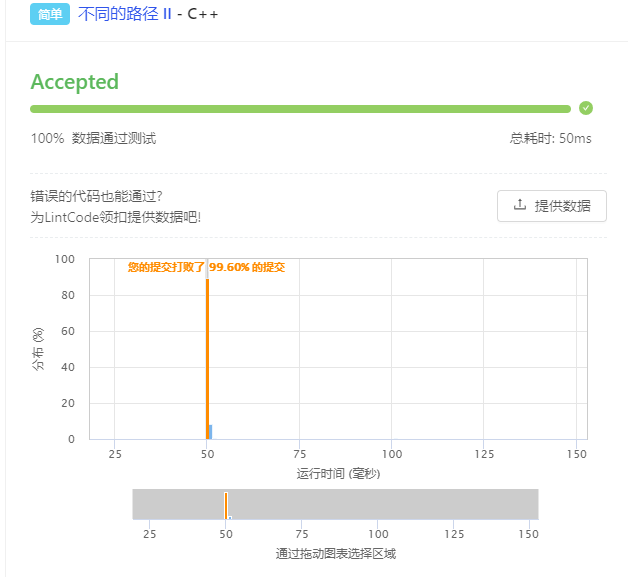
与之前类似，也不是把元素真正从数组里面删除了，两个指针，指到不一样的时候替换前一个指针的后一个元素。

101. 删除排序数组中的重复数字 II



如果一个数出现两次以上，那么输出的两个相同数字是一头一尾的两个数，中间的数因为两边都有与他相同的数所以没有操作。最后到number-1的时候，j+1找不到数组数据，所以在最前面额外写一个if语句。

115. 不同的路径 II

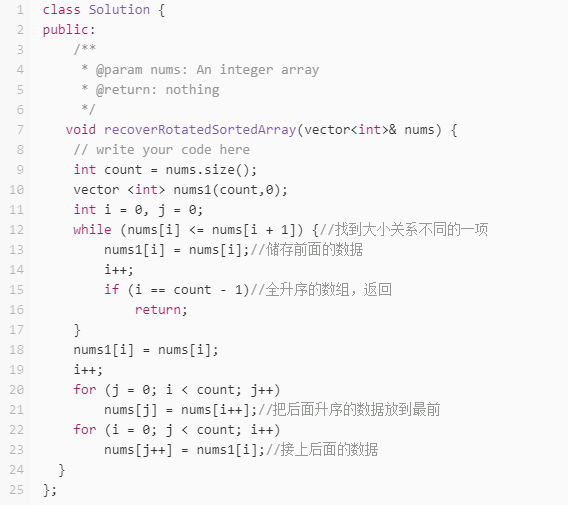


障碍物在边界上：此点以及它后面的点都为0。

障碍物在内部：此点为0。

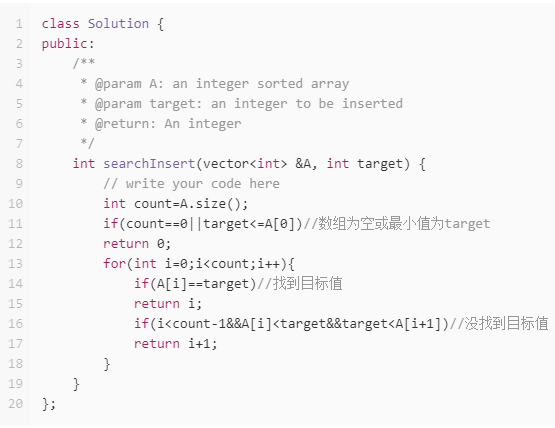
每一步的步数等于其上方及左方的步数之和。

39. 恢复旋转排序数组



在出现>关系的地方断开，保存前面的数据，把断开处的数据接到数组最前面，再把之前保存的数组放在后面。

60. 搜索插入位置

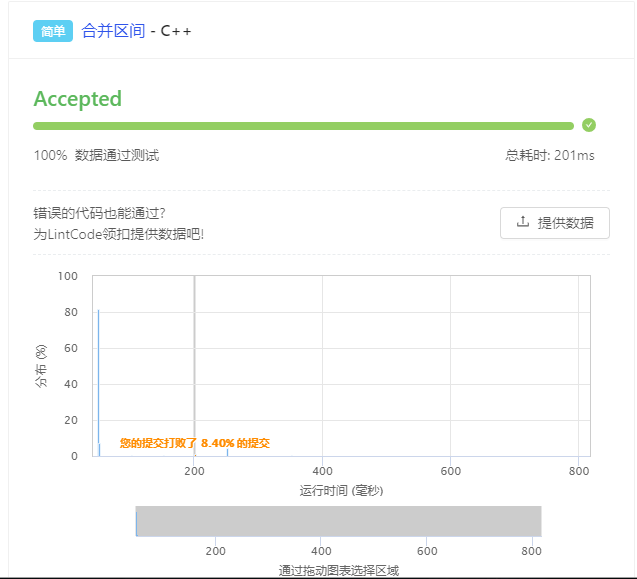


这题按字面意思来写即可，最后循环里面为了防止越界要在到达倒数第二个数据停下。

如果要达到时间复杂度为O(logn),就要用二分查找。之前题目里面的代码可以稍作改动直接拿来用。

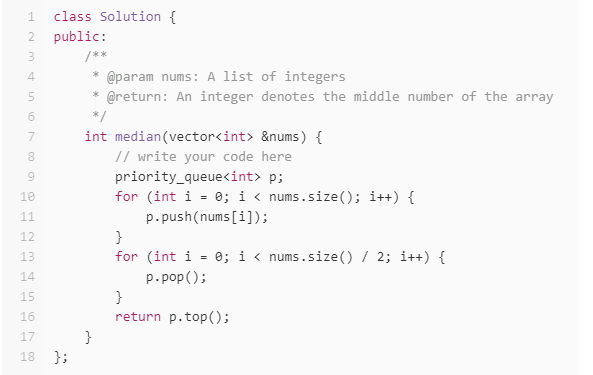
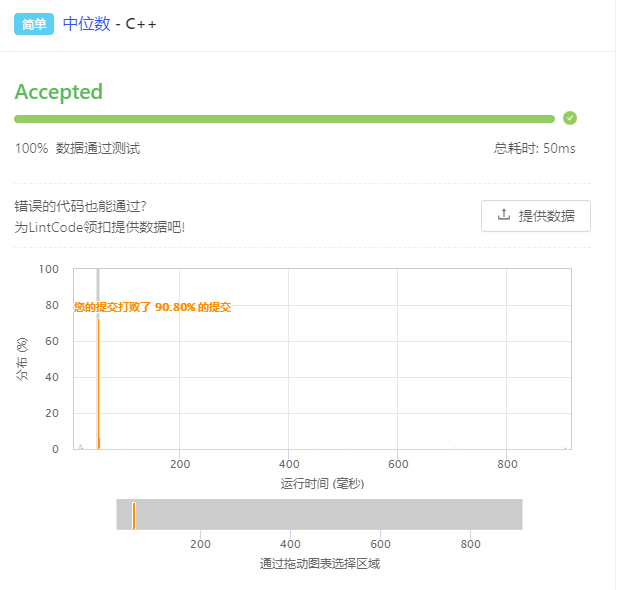


156. 合并区间



先按区间左侧升序排序，找到右侧比下一个区间左侧大的，替换数据后用erase把下一个区间删掉，指向不变。如果不满足条件，指向下一个。

80. 中位数



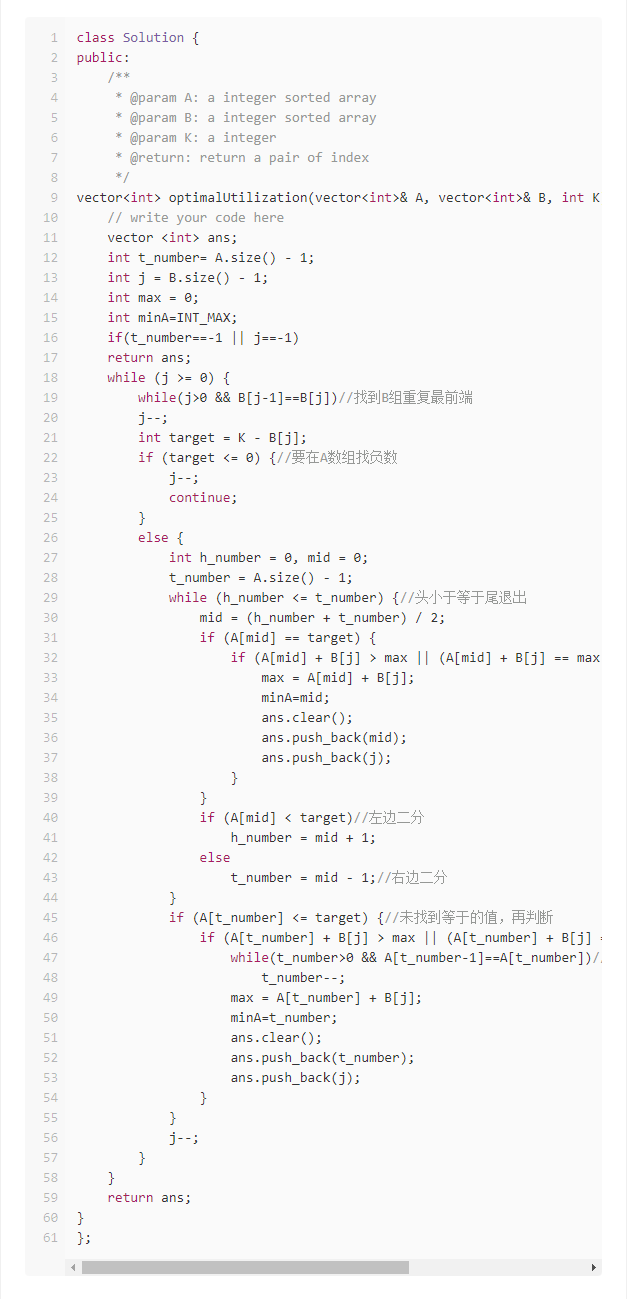
用 priority\_queue把内部元素先排序一遍，然后pop掉一半元素，再把top弹出来。

1794. 重复计数



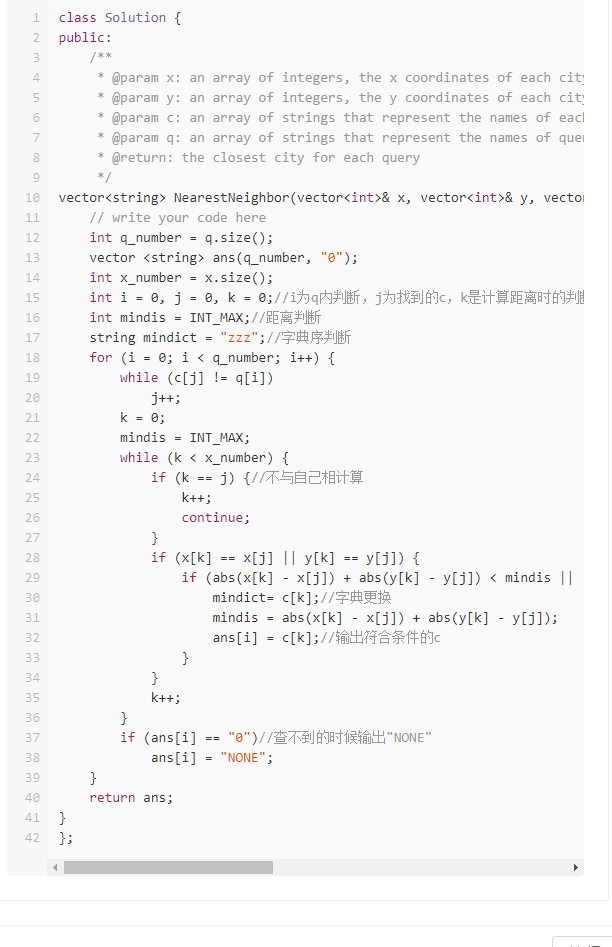
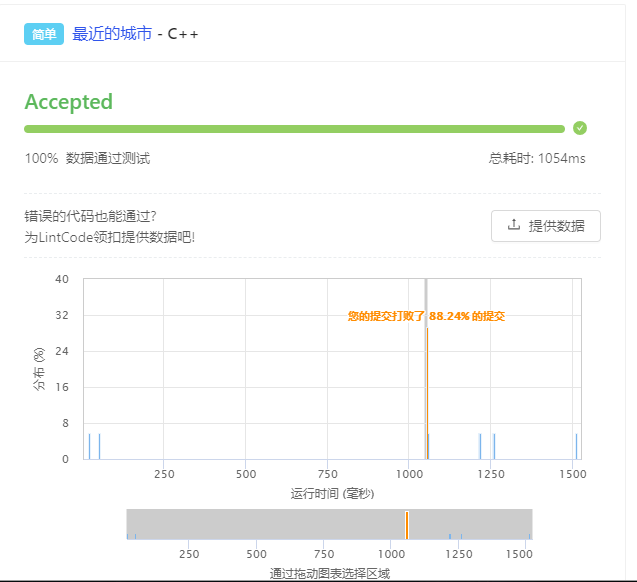
新建一个vector,存nums里面的值，如果有重复的，那会多次指向同一个下标，第一次都是false,第二次就会是true。再用迭代器检测在nums1里面是否已经出现过，没有出现过就加到nums1里面。

1797. 最佳利用率



用输入数减去数组B末位数，再到数组A里面寻找。如果得出的差小于等于0，就不用再继续寻找。如果大于0，用二分查找找目标数，如果找不到，则下标落到最近的满足条件的下标。符合条件后更新最小索引，清除原有数组，加入最新数。

280. 最近的城市



对于每个q内的询问，循环找到c里面的对应数据，然后从第一个开始判断是否有x或者y相等的节点，如果有的话判断距离是否最短。若距离相同判断字典序是否更小。如果所有节点都没有x或y相等，输出NONE。