****软件工程算法

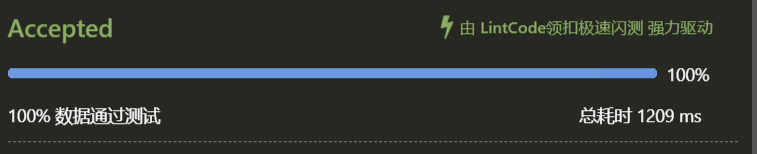
练习设计报告

专业班级： 11J193

学 号： 20191000415

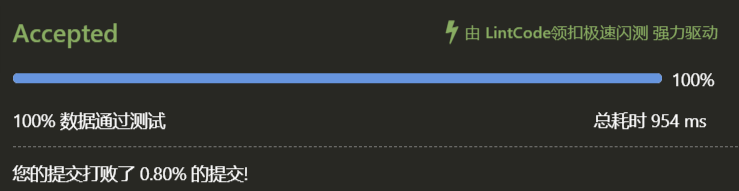
姓 名： 柴生华

1. 数组(按照通过率由低到高)

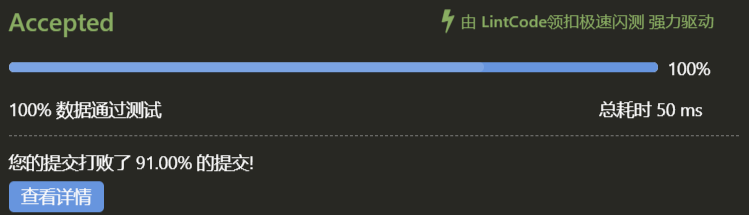
208.最近的城市：

思路：为了尽可能节约时间，所以遍历次数要少。第一次遍历找到目标城市，第二次遍历去寻找最近的城市：因为要求的是欧拉距离，且当x或y与目标城市相同时，才视为存在最短距离，所以第二次遍历的时候分两种情况，x相同和y相同，然后计算y或x的差值，将更短的距离存储，并记录此时的城市名字,遍历结束后便可以找到最近的城市。

1. 合并区间

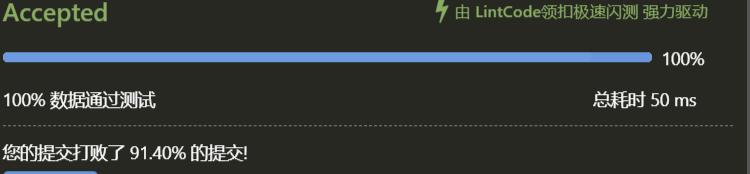
思路：先将所给区间排序，排序规则将区间左值较小的排在前面，弱左值相同，则右值较小的排在前面。排序完成后，一次判断相邻的两区间，若前一区间的右值小于后一区间的左值，则不合并；若大于，则合并两区间，将前一区间的右值更改为为两区间右值的较大值，并且删除后一区间，若发生了区间合并，则循环先不向下进行。遍历完所有相邻区间后区间合并完毕。(不足：因为区间排序我才用的冒泡排序的方法，所以效率很低，时间过慢)

1. 中位数



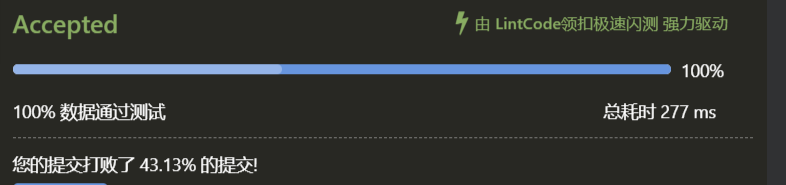
此题比较简单，先用sort经行排序，然后/2找到中位数即可。

1. 搜索插入位置

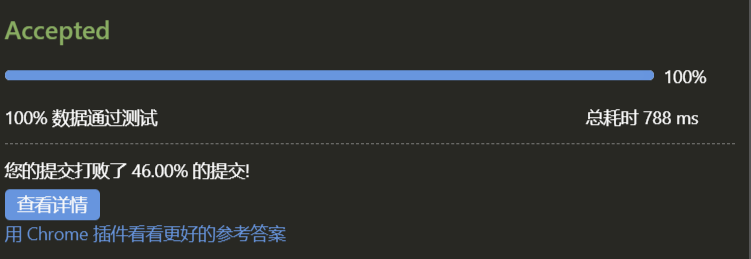


一个循环即可搞定，时间复杂度为O(n)，因为数组已经排好序，所以直接搜索，出现相同的话直接返回位置，若n<target<n+1则n为插入位置。

1797 .最佳利用率

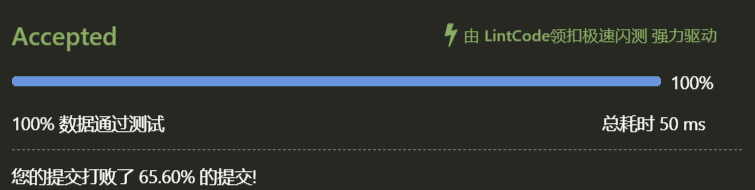


两个数组，从前向后遍历，找到满足和的最大值，直接暴力搜索就可以了。

101.删除排序数组中的重复数字Ⅱ 

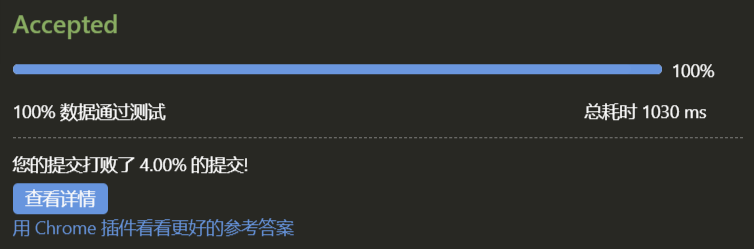
本题采用双指针法，因为是排序数组，只需判断相邻两数是否相同即可。指针i和指针j一起向右移动，若相邻两数在两次以内相同或不同，则每次移动都将i的值赋值给j，若连续两次以上相同，则说明此数出现超过三次，此时j停止移动，i向后移动，当i的值和j的值不同时，将i的值赋值给j，并且再次向后移动，知道i遍历完全部，时间复杂度为O(n)。

1. 恢复旋转排序数组



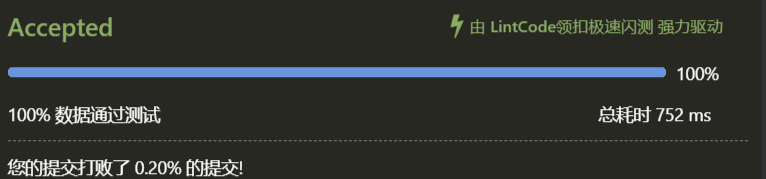
有两种方法，第一种直接用sort进行排序，但是与池体出题意愿不符；第二中方法，判断n和n+1的大小，若n<n+1则将首位添加到数组末端并且删除首位，然后在进行判断，直到n>n+1最后一次将首位添加到数组末端并且删除首位，完成恢复。

1. 删除排序数组中的重复数字



依然采用双指针法，判断相邻两数字是否相同，利用i，j指针.n，n+1的值若相同，则j指针不动，i指针向后移动，直到i，j的值不同，再把j的值赋值给i,完成删除重复数字.

1. 最长上升子序列



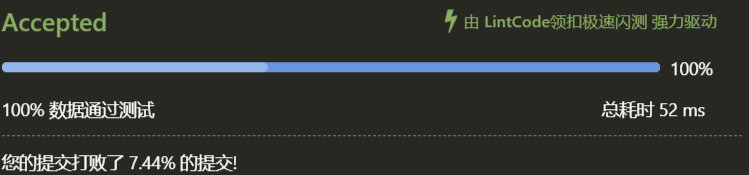
该题意思为最长的递增或递减数列，分两种情况，第一种去计算递增序列的最长长度，第二种去计算递减序列的长度，然后取最长。

1. 删除元素



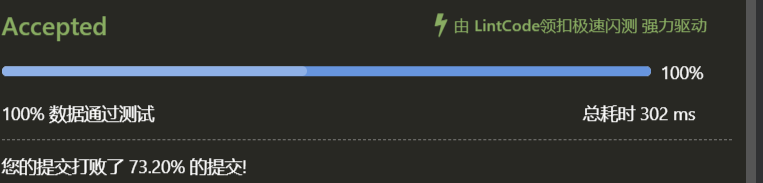
删除指定元素，此题比较简单，遍历找到目标元素，然后用erase（）函数删除即可。

1250.第三大的数



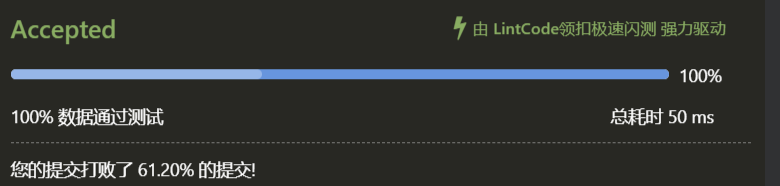
先借助先借助sort进行排序，然后找到第三大的数，需要注意的是，第三大的数不是倒数第三个数，数据存在重复，所以要从前往后找到不重复的第三个数字。

14.二分查找



普通的暴力搜索查找确实简单，但是很浪费时间，不是很可取，需要将排序数组分为两部分，观察中间的值比目标值大还是小，然后再去确定搜索范围，减少了工作量，提高了效率。

407.加一



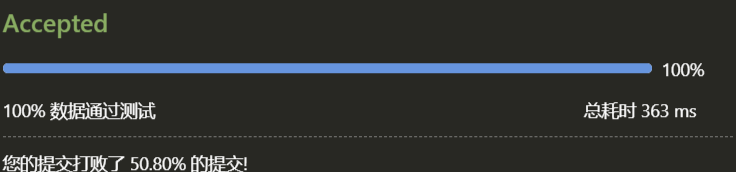
此题的关键是进位，从数组的最后一位算起，要判断好是否进位，同时注意若数组的首位还要进位，则应当在数组的首位插入1。

1. 两数之和



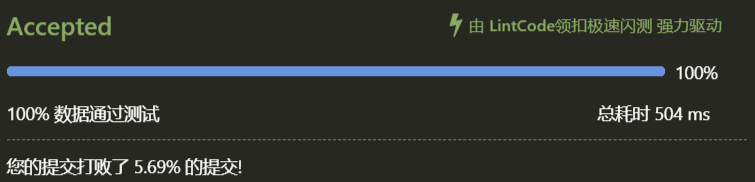
从前向后遍历，先计算目标值与当指针的值的差值k，再去寻找数组中是否存在值和k相等，若存在，则记录角标，并返回。

64.合并排序数组



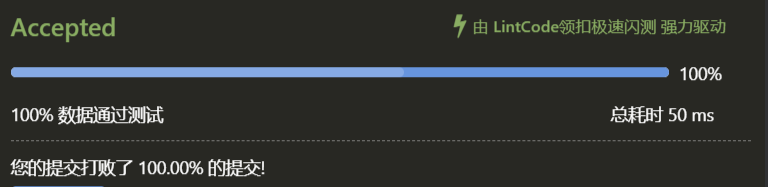
此题需要将两个数组合并，并且排好序，若直接拼接再进行sort会很浪费时间，因为A数组足够长，所以从A数组的m+n-1开始，从A，B两数组中挑选最大的值放在末端，然后依次类推，把B全部放入A数组中。

1. 重复计数



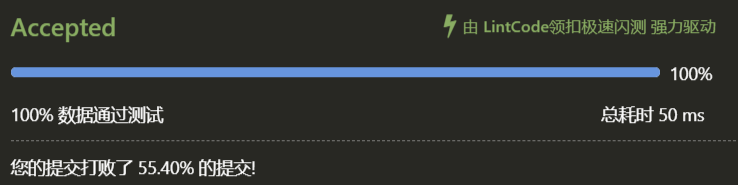
定义一个vector，来保存重复的元素，依然是先排序，然后采用暴力搜索的方式记录数据是否重复。

1157.最短无序数组



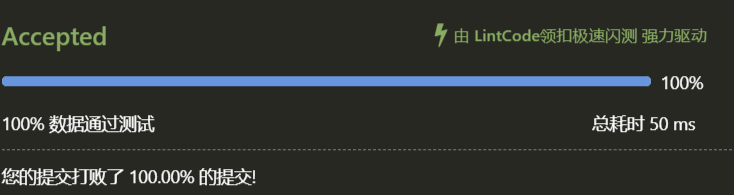
先将原数组用sort进行排序，然后判断排序之后的数组和原数组有哪些地方不一样，找到这不一样的地方即为无序数组。

1. 链表
2. 链表求和



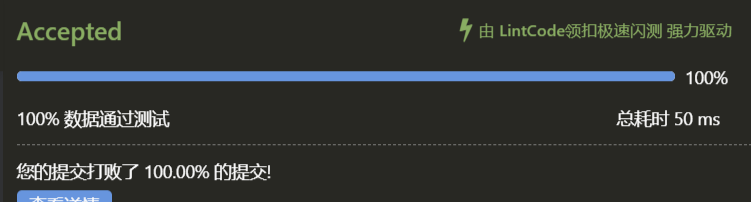
若A链表的长度大于B链表的长度，将A链表中的数据依次和B链表中的数据相加，并赋值给A链表，注意进位，若求和结果长度大于A链表，则取B链表的最后一位接到A链表的末端来存贮进位；若B链表的长度大于A链表的长度，将B链表中的数据依次和A链表中的数据相加，并赋值给B链表的相应位置，注意进位，若求和结果长度大于B链表，则取A链表的最后一位接到B链表的末端来存贮进位。

174.删除链表中倒数第n个节点



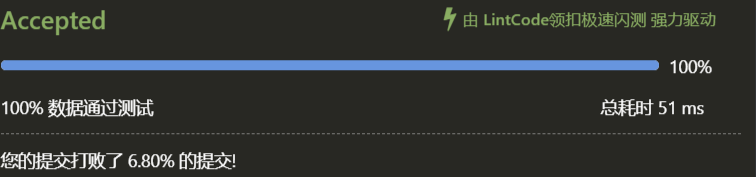
第一次先遍历链表找到链表长度，然后再算出正数的位置，利用head->next=head->next->next来删除目标节点。

112.删除链表中的重复元素



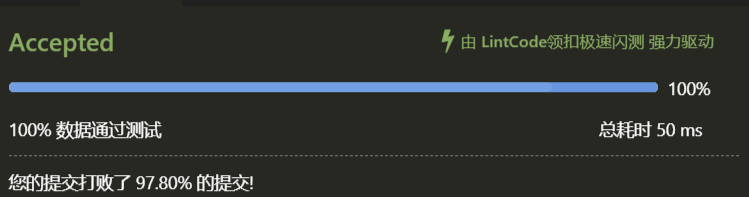
判断head->val=head->next->val是否成立，若成立，则删去一个，删去的方法依旧是head->next=head->next->next。

1. 两两交换链表中的节点



与其交换节点，不如交换值快一些，借助一个中间变量将链表的节点的值进行交换。

1. 合并两个排序链表



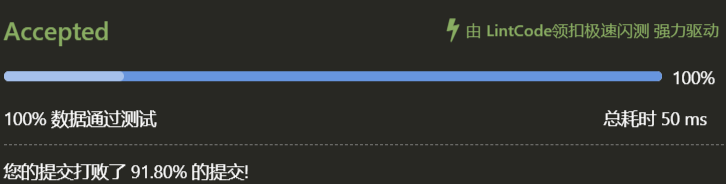
利用两个head指针，依次比较指针所指向节点的大小，将较小的排在前面，比如A链表的头较小，A链表的头排在首位，然后A链表的head移动到下一个节点，然后B的head指针不动，再去比较两个指针所指节点的val大小，依次连接所有节点。

1. 倒数第n个节点



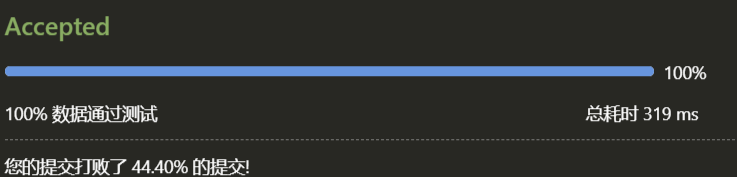
第一次先遍历链表找到链表长度，然后再算出正数的位置，最后返回该位置的节点即可。

35.反转链表



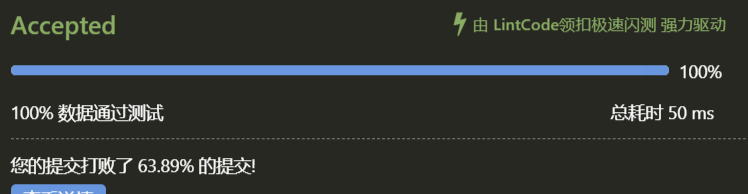
反转链表只需要将后面的节点的next指向前一个节点的地址即可，但是要注意起初的末地址指向的是null，最后要将首位节点指向null。

1. 在O(1)时间复杂度删除链表节点



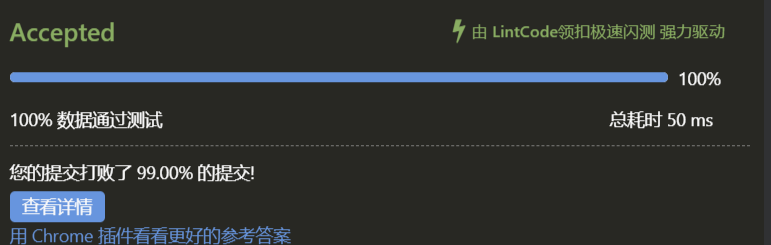
删除节点直接利用node->val=node->next->val;node->next=node->next->next;更改一下此节点的值为下一个节点的值，然后删除下一个节点可以达到相同的效果。

1609.链表的中间节点



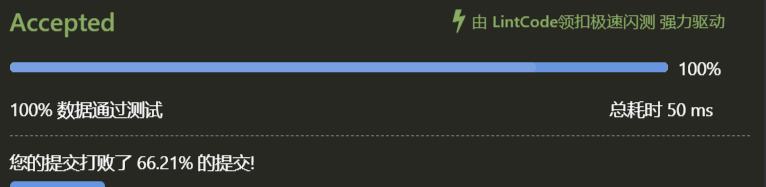
第一次先遍历链表找到链表长度，然后/2找到链表的中间节点，返回中间节点的地址即可。

1. 相反的顺序存储



先将链表中的数据按照顺序读取到vector中，然后再将这个vector中的数据倒序存入另外一个vector中。

756.两数相乘



此题将链表中的数据转化为整数，第一次先遍历链表找到链表长度n，然后每一位乘以10^(n-1)求和，最后把所得数据相乘即可。