CF1175题解

A

贪心，显然能除就除，不能除就直接减掉余数，复杂度O(log n)。

B

显然次数就是分开的for循环求和，嵌套的for循环求积。

关键在于怎么处理嵌套的for循环。考虑一个栈。

遇到for就把n压栈，遇到add就给答案加上当前栈顶，遇到end弹出栈顶并乘到答案里。

如果答案溢出直接输出并结束程序即可。

C

题目意思是最小化你找的点跟其他点的距离中第k+1小的那个。

注意到，和某个点距离前k小的区间在序列上一定是连续的k个点。

所以我们可以枚举i到i+k所形成的k+1个点，然后求出区间长度，将我们要选的点放在区间中央即可，这样的话长度为l/2，l为这个区间的长度。所以我们本质就是找到最短的含k+1个点的区间即可，直接枚举就可以了。

题解区一堆二分，有必要吗。。。

D

利用算两次的思想，可以发现答案是k段后缀和，其中一段必然是整个区间。

因此我们在剩下的后缀和中找到k-1个最大的，求和就结束了。

E

这题问你覆盖一段区间需要最少几条线段。

一开始可以想到贪心，即把所有线段按照右端点排序，然后每次找与其有重合且右端点最右的线段。但是贪心是O(nq)的，无法通过本题。

考虑利用倍增来优化这个贪心。

设dp[i][j]表示从i出发，往右延伸2j条线段之后能到达的最远距离。

那么对于一条线段[l,r]上的所有点x，都有dp[x][0]=r。

利用倍增思想可知dp[i][j]=dp[dp[i][j-1]][j-1]。

那么询问就降到了log量级，可以通过本题。

F

有O(n log n)做法和O(n)做法。

O(n log n)做法：

注意到一个区间[l,r]如果是子排列，那么：

[l,r]区间内没有重复数字，且[l,r]区间内最大值为r-l+1。

于是可以分治，每次考虑包含在大区间[L,R]中且经过mid 的所有区间。其中mid 是[L,R]中最大值所在的位置，有多个最大值随意选一个。

那么这些区间的最大值就确定为a[mid]了，只需再满足第一个条件。

预处理每个位置的pre表示这个数字的前一次出现位置。令 maxpre为pre 的前缀max ，那么[l,r]中没有重复的数字就等价于maxpre[r]<l（注意到maxpre[l] 肯定<l ，所以没问题）。

考虑优化复杂度。查看大区间左半边和右半边哪个更短。假设左半边更短，那么枚举每个[L,mid] 中的位置作为左端点，由于区间长度必须等于 a[mid] ，那么右端点也就确定了。此时再看这个区间是否满足maxpre[r]<l 即可。

O(n)做法：

考虑某个 i 以及此时所有 i 之前的位置 j 到 i 的最大值v[j]​=max(a[p],p=j,j+1,…,i)，容易单调栈维护相同 v[j​] 形成的极长连续段 [l[j]​,r[j​]]，则 v[r[j​​]]=a[r[j]]​​，且v[l[j]​−1​]>v[l[j]]​​=⋯=v[r[j​​]]。

此外，我们需要将a[i​] 上一次出现位置 p 及之前的 j 全部弹出，因为这些位置到 i 之间出现了重复数字，不可能产生贡献。因此需支持开头删除，末尾加入删除，将栈改成双端队列。

在此基础上，我们注意到如果一个极长连续段 [l,r] 在 i 处对答案产生贡献，则必然是 l∼i 形成 1∼i−l+1 的排列：若l<p≤r 的 p 对答案产生贡献，则v[p​]=i−p+1，推出 v[l​]<i−l+1，那么 l∼i 之间必然出现重复数字，矛盾。

进一步地，考虑一个极长连续段 [l,r]，则它会在i=l+v[r]​−1 即l+a[r]​−1 处产生 1的贡献，可以直接开数组val[i]​ 表示在 i 处产生贡献的极长连续段数量，则对每个 i 操作结束后将答案加上val[i]​ 即可。

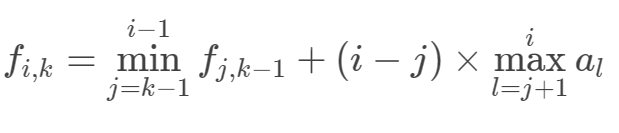
对极长连续段进行修改时，在原来产生贡献的位置将val 减去 1，在新产生贡献的位置将val 加上 1。由于对极长连续段进行修改的次数均摊线性，所以时间复杂度为小常数 O(n)。

G

一个知识点非常综合的好题。

斜率优化+单调栈+可持久化李超树+凸包启发式合并

首先可以有个O(n2k)的DP：令dp[i][k]表示前i个数分成k段的最小值，则

dp[i][k]=min(dp[j][k-1]+(i-j)\*max(a[l],l=j+1,j+2,...,i),j=k-1,k,...,i-1)

是个2D/1D型动态规划，但是式子不满足四边形不等式，因此无法使用决策单调性优化。

考虑其他方式优化。max不好处理，注意到序列可以划分为若干段，其中每一段的max 值都相等。这部分可以用单调栈维护。

在每一段内部，需要求出dp[j][k−1]−j\*max 的最小值。这显然可以用斜率优化。所求即为经过点(j,dp[j][k-1]) 且斜率为max 的直线在 y 轴截距的最小值。凸包上二分即可。

每次向单调栈加入一个值时可能会将若干段合并为一段，因此需要合并凸包。可以将一个凸包的点直接插入另一个凸包。凸包上点的横坐标已经有序，因此可以在O(n) 时间内合并两个大小为 n 的凸包。为了保证时间复杂度，考虑用启发式合并，将小凸包的点插入大凸包。时间复杂度为O(nlogn)。

令k=max，b=dp[j][k−1]−j\*max ，则dp[i][k]​ 即为每一段的直线 y=kx+b 在x=i 时纵坐标的最小值。可以用李超树维护。因为插入的是直线，所以复杂度为 O(nlogn) 。注意到合并段的时候会删除最新加入的几条直线，可以用一个栈记录操作，或者可持久化。

将以上过程重复 k 次即可。时间复杂度O(nklogn) ，空间复杂度O(nlogn) 。