题解

感谢 @bdzzj 对本场比赛的贡献。

114514

首先我们需要得到一个得出 trans 的策略。由于要求字典序最小，所以可以从前往后贪心，每次增加当前的数直到与前面的任何一个数都不同即可。

第一档是给爆搜的。时间复杂度看似是 $O(10^n)$，实际上是 $O(10^{\underline{n}})$ 的。

考虑计数。我们反过来进行这个贪心的过程，那么 b_i 这个位置合法的条件相当于是，对于所有整数 $x \in [b_i, a_i)$，都存在一个 $1 \le j < i$，满足 $a_j = x$。那么显然每个位置是独立的，可以乘起来。

这个就相当于维护一个桶，每次求 a_i 前面的极长连续 1 的长度，然后将 a_i 加入桶。

暴力往前跳可以通过 第二（第三）档，线段树上二分之类的维护可以通过第四档。

正解其实是开一个并查集，每次将 a_i 合并到 $a_i - 1$ 上面。这样子的时间复杂度并不是 $O(n \log n)$，而是 $O(n)$ 的。

具体分析（来自 @bdzzj）考虑均摊，因为每个 a_i 互不相同，所以每个点实际上只会经过一次就被路径压缩掉了。

还有一个离线做法是记每个位置改为 1 的时间戳（没改过 1 的就设置为 ∞），然后相当于是找前面第一个在它后面被改为 1 的位置。

单调栈即可 $O(n)$ 离线。

出题人没有强制在线并对每个前缀求出答案，你应该感谢良心出题人。

沉默乐团

以下设 $\max r_i$ 为 m。

不难发现，题目中的限制等价于不存在一对真前缀和真后缀的和相等。

首先我们可以发现一个性质：如果这对真前缀和真后缀相交，那么一定存在一对不相交的真前缀和真后缀的和也相等。

证明是容易的，假设它们分别为 $[1, i], [j, n]$（$i \ge j$），那么 $[1, j-1], [i+1, n]$ 这一对，两边和都减去了 $[i, j]$ 的和，所以两边也相等且不相交。

那么我们只需要考虑不相交的真前缀和真后缀。考虑设真前缀的和分别为 $[pf_1, pf_2, pf_3, \dots, pf_{n-1}]$，真后缀的和分别为 $[sf_2, sf_3, sf_4, \dots, sf_n]$，显然 pf 严格单调递增，sf 严格单调递减。考虑将 pf 与翻转的 sf 归并排序得到的新序列，那么我们只需要判断新序列相邻两项是否相等。

考虑将判断过程写到 dp 里。那么我们只需要记录当前的两个指针 i, j 与 pf_i, sf_j 即可，每次钦定两边更小的那边转移，当 i, j 相交时把答案转移到 ans 中，但是这样时间复杂度巨大，是 $O(n^4 m^2)$ 以上的。

考虑优化。我们发现完整记录 pf_i, sf_j 是没啥用的，真正有用的信息只有，小的那边要超过大的那边需要加的值，而这个值不可能超过 m。所以我们只需要记录它即可。

设 $f_{i,j,k}$ 表示当前已经加入了 pf_{i-1}, sf_{j+1}，左边大 / 右边大，大的那边比小的那边大 k 的方案数。那么有转移：

$$f_{2,n,0,k}=[k\in[l_1,r_1]]$$
$$f_{1,n-1,1,k}=[k\in[l_n,r_n]]$$
$$f_{i,j,0,k}\to f_{i+1,j,0,k+x}\ (x\in[l_i,r_i],k+x\le m)$$
$$f_{i,j,0,k}\to f_{i,j-1,1,x-k}\ (x\in[l_j,r_j],x-k\ge 1)$$
$$f_{i,j,1,k}\to f_{i,j-1,1,k+x}\ (x\in[l_j,r_j],k+x\le m)$$
$$f_{i,j,1,k}\to f_{i+1,j,0,x-k}\ (x\in[l_i,r_i],x-k\ge 1)$$
$$(r_i-k-\max(l_i-k,1)+1)(f_{i,i,0,k}+f_{i,i,1,k})\to ans$$

时间复杂度为 $O(n^2m^2)$，使用差分 / 前缀和即可优化至 $O(n^2m)$。

深黯「军团」

第一第二档是暴力，第三档是给可能存在的做法的。

接下来讲正解。

去掉若干个完整的循环节（众所周知，所有 $1\sim n$ 的排列的逆序对之和为 $n!\frac{n(n-1)}{4}$）之后，原题目就转化为 $O(1)$ 个这样的问题：

> 对于一个 $1\sim n$ 的排列 p，求出所有字典序 $\le p$ 的排列 q 的 $\operatorname{Inv}(q)$ 之和。

看到排列字典序，考虑康托展开。

康托展开是一个关于求排列排名的算法，结论是：

$$rk=\sum_{i=1}^{n} sf_i\times(n-i)!$$

其中，sf_i 是在 i 后面且比 i 小的数的个数，即 $sf_i=\sum_{j=i+1}^{n}[a_j<a_i]$（显然 $0\le sf_i\le n-i$）。

具体证明考虑枚举 LCP，然后选一个前面没出现过的数，后面可以随便乱排。

实际上我们发现 rk 本质上是一个不等进制数（下面我们称它为排列进制数），右往左数第 i 位（编号从 0 开始）的取值是 $0\sim i-1$，权值是 $(i-1)!$，根据定义，它的实际值就是所对应的排列排名，所对应的排列的逆序对个数就是它的数位之和。

所以原问题相当于将 p 转化为一个上述的排列进制数 sp，求所有 $0\le sq\le sp$ 的 $\operatorname{pops}(sq)$ 之和，其中 $\operatorname{pops}(x)$ 表示 x 的数位之和。这个做一个简单的数位 DP 即可。而转换进制时只需要跑一遍二维偏序即可求出所有的 sf。

至于如何快速得到 A_{k-1} 的排列进制数，只需要将初始的 a 转化为排列进制数，再将十进制数 k 也转化为排列进制数，直接加即可。

设 $f(x)$ 为反阶乘函数（即 $f(n!)=n$），总时间复杂度为 $O(n\log n+f^2(k))$。

下面是来自验题人 @bdzzj 的另外一个做法的题解：

直接模拟。

容易发现整块的逆序对是好算的，即 $y_1,y_2,\dots,y_{n-m},x_1,x_2,\dots,x_m(x_1<x_2<\dots<x_m)$ 到 $y_1,y_2,\dots,y_{n-m},x_m,x_{m-1},\dots,x_1$ 这之间的所有排列的逆序对和是好算的，所以直接暴力把排列加速模拟跳 `next` 即可。

具体的先预处理前缀的逆序对从后往前把后缀变成整块，再从前往后能跳整块就跳。

设 $f(x)$ 为反阶乘函数，时间复杂度 $O(n\log n+\text{poly}(f(k)))$。

终末螺旋

先考虑不带修的情况怎么做。

我们考虑将颜色 i 的两座塔视作一个区间 $[l_i,r_i]$。则一个区间被激活，可以使所有与这个区间有交的其他区间都被激活。

考虑暴力拓展，则一个初始区间 $[l_i,r_i]$ 经过若干轮拓展后必定会拓展到一个区间 $[L,R]$，我们称其为一个“终止区间”。显然，所有的终止区间要么包含，要么不交。假如我们将所有的终止区间拎出来按包含关系建树，最终会得到一片森林。则森林中的每棵树根内都要选一个点手动激活，而且选择的点位置不能被更小的终止区间包含。

那么答案就很显然了，分别是森林中树的个数，以及每棵树 根的区间长度减去所有儿子的区间长度之和的乘积。现在我们的目标就是求出所有的终止区间。

考虑到，因为终止区间无法拓展，所以每个初始区间与终止区间要么包含，要么不交。所以对于每个终止区间，一个颜色 x 在区间内的出现次数要么为 0 要么为 2。这让我们想起了 XOR-hash：给每个颜色随机赋权，则一个终止区间内的权值异或和为 0，且其不能被划分为多个终止区间。

做个前缀异或，将 $[l,r]$ 内异或和等于 0 转化为 $sa_{l-1}=sa_r$，其中 sa_i 表示前 i 个元素的带权异或和。而如果一个终止区间 $[l,r]$ 满足 $sa_{l-1}=sa_r=0$，则该区间为森林的一个根。

我们考虑将被非根区间包含的点覆盖。考虑记一个 c 数组为一个点被覆盖的次数，那么对于一个前缀异或和 $x(x\neq 0)$，设其在 sa 中出现的位置从小到大排序分别为 $pos_{x,0},pos_{x,1},\dots,pos_{x,k}$，则将 c 区间 $[pos_{x,0}+1,pos_{x,1}],[pos_{x,1}+1,pos_{x,2}],\dots$ 加 1。这也相当于将 c 区间 $[pos_{x,0}+1,pos_{x,k}]$ 加 1。最后，设 0 在 sa 中出现的位置从小到大排序分别为 $pos_{0,0},pos_{0,1},\dots,pos_{0,k}$（显然有 $pos_{0,0}=0,pos_{0,k}=n$），那么答案显然分别为 $k,\prod_{i=1}^{k}\sum_{j=pos_{0,i-1}+1}^{pos_{0,i}}c_j$。

差分维护 c，则单次询问的时间复杂度为 $O(n)$，总时间复杂度为 $O(nq)$。

考虑使用数据结构维护交换过程，交换相邻两个颜色对于异或和的影响相当于单点修改。

假设将 $x=sa_p$ 改为 y，则修改可以分为三种情况：

1. $x\neq 0,y\neq 0$

   此时我们的修改操作与 0 无关，所以 k 不变。我们使用 set 维护每种前缀异或和出现的位置，就相当于在 x 的 set 中删去 p，并在 y 的 set 中加上 p。先撤销掉 x,y 的区间覆盖操作，修改两个集合之后再覆盖回去即可。因为终止区间形成了一片森林的结构，所以 x,y 的区间覆盖操作只会影响一个根区间内部，同样撤销这个根区间的贡献，更改完之后再加入回去。

2. $x=0,y\neq 0$

   此时我们相当于合并了两个根区间，撤销原先两个根区间的贡献，更新 y 的区间覆盖操作再加入新的贡献即可。

3. $x\neq 0,y=0$

   此时我们相当于分裂了两个根区间，撤销原先根区间的贡献，更新 x 的区间覆盖操作再加入新的两个贡献即可。

发现区间覆盖是不能撤销的，我们将区间覆盖改为区间加一，查询时查询区间 0 的个数。因为 c 数组时刻都非负，所以只需要开一棵线段树维护最小值以及最小值个数。模数是质数，所以撤销贡献时直接乘上逆元即可。随机哈希使用 `unsigned long long`，给每种前缀异或和用 `unordered_map` 动态离散化一下即可。总时间复杂度为 $O((n+q)\log n)$，常数可能较大。