题目大意

给定一个长度为 n 的数组 A，其中 $a_i \in [1,m]$，令 $f(l,r,i)$ 表示 $j \in [l,r]$ 中所有满足 $a_j \leq i$ 的 a_j 的最大值，如果不存在这样的 j，则 $f(l,r,i)$ 为 0。

求 $\sum_{l=1}^{n}\sum_{r=l}^{n}\sum_{i=1}^{m} f(l,r,i)$。

数据范围

对于所有测试点，$1 \leq n \leq 1 \times 10^6, 1 \leq m \leq 1 \times 10^9$。

Subtask 1 (5pts):$n \leq 500$

Subtask 2 (5pts):$n \leq 4000$，依赖 Subtask 1。

Subtask 3 (5pts):$m \leq 2$。

Subtask 4 (20pts):$m \leq 50$，依赖 Subtask 3。

Subtask 5 (10pts): 保证 a_i 在 $[1,m]$ 中随机生成，$n \leq 5 \times 10^5$。

Subtask 6 (20pts):$n \leq 10^5$，依赖 Subtask 1,2。

Subtask 7 (35pts):无限制，依赖 $\text{Subtask 1,2,3,4,5,6}$。

解题过程

做法 1

离散化后，按题意暴力依次枚举 i,l,r，在枚举 r 的时候维护 $f(l,r,i)$ 的值。

时间复杂度 $O(n^3)$。

可以通过 Subtask 1，期望得分 5 。

做法 2

考虑沿用做法 1，考虑在枚举 l,r 的过程中，同时计算所有 i 的答案。

假设当前 $[l,r]$ 中的数构成集合 S，每个数 $x \in S$ 能贡献到的 i 是 $[x, next_x)$，其中 $next_x$ 表示 S 中最小的比 x 大的数。

如果在确定 l 的情况下，从小往大枚举 r，此时需要用数据结构去维护集合的加入，复杂度 $O(n^2 \log n)$。

考虑倒着枚举 r，那么我们只用实现删除，可以用双向链表维护，复杂度 $O(n^2)$。

可以通过 Subtask 1,2，期望得分 10 。

做法 3

考虑 $m \leq 2$，此时 a_i 只有两种取值，

可以分类讨论，

假如当前段 $[l,r]$ 只有 1 或 2，那么对答案的贡献为 2。

否则对答案的贡献为 3。

考虑对这样的段计数，可以通过每个极长 1 和 2 的段算出对应的数量，然后可以算出答案。

复杂度 $O(n)$。

可以通过 $\text{Subtask } 3$，期望得分 5 。

加上算法 2，期望得分 15 。

做法 4

算法 3 启发我们从值域上下手。

此时分类讨论的状态太多，我们不好维护。

考虑这样一个问题，我们在计算 $f(l,r,x)$ 的时候，把 $>x$ 的数看作 0，那么问题变成了简单的区间 $\max$。

令初始序列全为 0，然后从小往大加入这些数，假设此时已经加入了所有 $a_i\le x$ 的数，然后我们统一计算所有区间的 $f(l,r,x)$ 的答案，这个问题可以维护一个单调栈或者笛卡尔树解决。

复杂度 $O(nm)$。

可以通过 $\text{Subtask } 1,2,3,4$，期望得分 35 。

做法 5

考虑沿用做法 4，用类似 Treap 的方法去维护这样一个笛卡尔树，相当于单点修改 Treap 的 fix 值，可以使用旋转 Treap，或者 FHQ Treap 的 merge 和 split 维护。

因为数据随机，所以此时树高期望是 $O(\log n)$ 的。

复杂度 $O(n\log n)$。

可以通过 $\text{Subtask } 1,2,3,4,5$，期望得分 45。

做法 6

做法 4 的转化很类似于今年 UNR D2T3 大海的深度。

相当于有 n 次单点修改，每次修改后计算一下全局每个区间 $\max$ 的和，然后乘上它能贡献的时间长度贡献到答案。

这里引用原题解的做法：

考虑分治，处理所有跨过分治点的贡献。

分治。处理所有跨过分治点的贡献。只需 $O(\log n)$ 代价即可转化到原问题。

离线，扫描值域。令当前扫描到 x。对于每个 i，维护 p_i 表示第 i 时刻分治中心左边距离中心最近的 $\ge x$ 的数的距离，q_i 表示右边的距离。

x 变小时对 p,q 的修改均为区间取 $\min$，用 segbeats 维护即可。

还需要维护每个时刻的答案 w_i。第 i 时刻 $\max<x$ 的区间数量即为 $p_i\times q_i$，反过来可得 $max\ge x$ 的区间数量为 $(mid-l+1)\times(r-mid)-p_i\times q_i$。因此 x 增大时，需要令 $ans_i\leftarrow ans_i+(mid-l+1)\times(r-mid)-p_i\times q_i$。

每个 i 处维护四维向量 $(p_i,q_i,p_i\times q_i,ans_i)$，用矩阵表示修改即可。时间复杂度为 $O(n\log n)$。

加上外层分治，总时间复杂度为 $O(n\log^2 n)$。

可以通过 $\text{Subtask } 1,2,3,4,5,6$，期望得分 65。

做法 7

依旧考虑分治，计算跨过分治点 mid 的所有贡献。

还是从小到大枚举加入所有数，然后会发现有用的位置只有每个时刻 $[l, mid]$ 的后缀最大值和 $(mid, r]$ 的前缀最大值。

对于每个时刻，

每个 $[l, mid]$ 的后缀最大值位置 j 的贡献为 $b_j \times (x_j - mid) \times (j - y_j)$，其中 x_j 为最大的不超过 r 的数，满足 $\forall k \in (mid, x_j]$ 都有 $b_k \leq b_j$，y_j 为下一个比 j 小的后缀最大值位置。

每个 $(mid, r]$ 的前缀最大值位置 j 的贡献为 $b_j \times (mid - x_j + 1) \times (y_j - j)$，其中 x_j 为最小的不小于 l 的数，满足 $\forall k \in [x_j, mid]$ 都有 $b_k < b_j$，y_j 为下一个比 j 大的前缀最大值位置。

考虑快速维护这个贡献，

在从小到大加入每个数的过程中，当前加入的数一定是目前最大的。

不妨假设它的位置在 $j \in [l, mid]$，

1. 首先，它会弹掉那些在 $[l, mid]$ 中在它左边的后缀最大值，使它们的贡献消失，
2. 其次，它会使在 $(mid, r]$ 的前缀最大值的对应的 x_j 变成 $max(x_j, j)$。
3. 最后，加入 j 对应的贡献。

假设我们已经维护好了当前所有位置的贡献，第一、三部分可以用单调栈去维护暴力删除。

我们定义一个 $i \in [l, mid]$ 的后缀最大值 i 的支配集合为所有 $k \in (mid, r]$ 的前缀最大值 k，满足 $x_k = i + 1$。

容易发现一个 k 最多只会属于一个 i 的支配集合。

知道支配集合后，我们只用维护每个点的支配集合，并计算 $\sum b_j \times (y_j - j)$ 就可以快速维护第二部分的贡献。

如何求出当前位置 j 的支配集合，我们发现正好是第一部分被弹掉的位置的支配集合的并，并且支持 $O(1)$ 合并。

假如我们要同时维护两边的支配集合，在第一部分删除的时候会使得另一边的支配集合改变，可以用并查集维护这个点属于哪个支配集合。

于是我们可以在 $O(n\alpha(n))$ 算出对应分治区间的答案，算上外层分治，总复杂度 $O(n \log n\alpha(n))$。

可以通过所有 $\mathrm{Subtask}$，期望得分 100。

参考资料

[UOJ NOI Round #8 Day2 题解 - 博客 - zhoukangyang的博客]