Simple Queries(**DISTNUM**)
解题报告

CODECHEF
August Challenge 2015

杨乐（中山纪念中学）

1 题目描述

给定一个含 N 个正整数的数组 A。现有关于它的 Q 个询问，询问有以下五种类型：

- 1 l r：令 S 为由下标范围从 l 到 r 的不同的元素构成的有序集合。你需要求出
$$\left(\sum_{1\le i<j<k\le |S|} S_i S_j S_k\right)(\mathbf{mod}\ 10^9+7)$$
- 2 x y：将下标为 x 的元素赋值为 y
- 3 x：将下标为 x 的元素从数组中删除
- 4 z y：在下标为 z 的元素之后插入元素 y，若 z 等于 0，则在数组最前端插入
- 5 l r：输出下标在 l 到 r 范围内的不同元素个数

数组下标从 1 开始。数据保证数组总是非空。

2 数据范围

- $1 \le N, Q \le 10^5$
- $1 \le A_i, y \le 10^9 + 6$
- $1 \le x \le |A|, 0 \le z \le |A|, 1 \le l \le r \le |A|$
- 数据集 1(10 分):$Q * N \le 2 * 10^7$
- 数据集 2(5 分): 只含类型 5 的询问
- 数据集 3(10 分): 只含类型 1 的询问
- 数据集 4(15 分): 只含类型 2、类型 5 的询问
- 数据集 5(15 分): 只含类型 1、类型 2、类型 5 的询问，且 $N, Q \le 50000$
- 数据集 6(5 分): 只含类型 2、类型 3、类型 4 的询问
- 数据集 7(10 分):$N, Q \le 50000$
- 数据集 8(30 分): 无特殊限制

3 部分分详解

3.1 数据集 1

考察算法：**模拟**

严格按照题目所描述的操作与询问模拟数组 A 的变化。

时间复杂度：$O(NQ)$

3.2 数据集 2

考察算法：**莫队算法**

由于不存在修改，插入，删除操作，直接使用莫队算法回答类型 5 的询问。

具体地，维护当前区间内每一个数的出现次数，当发生从 0 到 1 或 1 到 0 的变化时相应地变化答案；离线排序，通过移动左右指针来回答所有询问。

时间复杂度：$O(N\sqrt{N})$

考察算法：**二维数据结构/可持久化线段树**

对于每一个数 A_i 找到最大的 p_i 满足 $A_i = A_{p_i}$（若不存在则令 $p_i = 0$）。

若查询区间为 $[l, r]$，那么满足 $l \le i \le r, p_i < l$ 的 i 的个数则是询问的答案：所有 $p_i \ge l$ 的 i 在区间中已重复，不必重复统计。

在二维平面上标记出所有的 (i, p_i) 点，每次询问转化为在矩形 $[l, r][0, l-1]$ 内的点数。解决二维平面上的问题，可以使用二维数据结构 (二维线段树/二维树状数组/树状数组套线段树/二维分块) 或可持久化线段树 (在不存在修改操作的情况下)。

时间复杂度：$O(NLog^2N)$、$O(N\sqrt{N})$、或 $O(NLogN)$。

3.3 数据集 3

考察算法：**容斥原理**

题目中要求相乘的三个数不能够重复，故容斥原理能很好的解决重复问题。

记数集为 $|S|$，所有数的一次方，二次方，三次方之和分别是 S_1, S_2, S_3，所求的 Sum 满足：

$$Sum = \frac{S_1^3 - 3S_1S_2 + 2S_3}{6}$$

容斥原理的核心是：加一个大的 S_1^3，减去其中不符合的 $3S_1S_2$，再补上额外减去的 $2S_3$。

但是简单的维护前缀和是不够的，这样会导致同一个数可能会被重复计算。

解决方法是把这些信息全部放到二维平面上，类比询问 5 回答即可。

3.4 数据集 4

考察算法：**平衡树**

由于存在修改，只能使用二维数据结构实现。

在修改 A_i 时，可能会修改若干个 p_i: 给每一个权值建造一棵平衡树。在修改权值时，在对应权值的平衡树查询前驱与后继，修改 p_i 而后再修改二维数据结构。

时间复杂度：$O(NLog^2N)$ 或 $O(N\sqrt{N})$。

3.5 数据集 5

结合数据集 3 与数据集 4 的算法。

3.6 数据集 6

根本不用输出就能得到这部分的分数。

4 参考解法

4.1 插入与删除操作

考察技巧：**在线转离线**

本题当中存在的插入与删除操作表面上要求使用在线算法解决，而实际上可以预先处理所有操作使问题转变为离线。

中心思想：预先处理所有数的相对顺序（包括已删除的）。

用一棵平衡树维护当前的序列————遇到插入操作直接在对应位置插入；遇到删除操作不直接删除该元素，而是把它标记为已删除。这要求平衡树的每个节点记录该子树内有多少个尚未删除的节点，这样在插入操作时可以保证插入位置的正确性。

把所有数的相对顺序确定后，插入与删除操作就可以转变为修改操作。

同时，还需要把询问操作的区间相应的进行修改。（这一步可以使用线段树简单地做到）

4.2 二维数据结构：二维线段树/树状数组套线段树

在用上述两种方法实现时，显然地树状数组套线段树较为简单。

具体的操作方法不再详述。

4.3 二维数据结构：二维树状数组

二维树状数组的空间复杂度太大 $(O(N^2))$，不适合这道题的解答。

如果使用哈希算法，则寻址的总时间复杂度太高了。

4.4 二维数据结构：分块

将整个平面分为 $\sqrt{N} \times \sqrt{N}$ 块，分别处理块内信息与块外信息。

每次询问一个二维区间时，包含的有若干个整块与若干个不完整的部分。对于整块，必然是一个矩形区域，所以维护一个二维树状数组能够方便地回答区间总和；对于不完整的部分，由于每一行或每一列最多同时存在一个点，直接枚举该部分的点就足够了。

这样的总时间复杂度为 $O(N\sqrt{N})$。