分块

李淳风

长郡中学

2024 年 6 月 27 日

引入

在前面两节中，我们探讨了树状数组与线段树两种数据结构。树状数组基于二进制划分和倍增思想，线段树基于分治思想。它们之所以能够高效地在一个序列上执行指令并统计信息，就是因为它们把序列中的元素聚合成若干“段”，花费额外的代价对这些“段”的信息进行维护，从而使得每个区间的信息可以快速由若干已知的“段”结合而成。

当然，树状数组和线段树也有缺点，那就是在维护较为复杂的信息（尤其是不满足区间可加、可减性的信息）时显得吃力，代码实现也非常繁琐。在本节中，我们将介绍分块算法。分块的基本思想是通过适当的划分，预处理一部分信息并存储下来，用空间换取时间，达到平衡。事实上，分块算法往往更加“朴素”，效率往往比不上树状数组和线段树。但它更加通用，而且容易实现。

例题

A Simple Problem with Integers

给定长度为 $n(n \leq 10^5)$ 的数列 a，然后输入 $m(m \leq 10^5)$ 行操作指令。
第一类指令形如 “C l r d”，表示把数列中第 $l \sim r$ 个数都加 d。
第二类指令形如 “Q l r”，表示询问数列中第 $l \sim r$ 个数的和。

我们已经使用树状数组和线段树在 $O((n+m)\log n)$ 的时间内解决过该问题，现在我们用分块来再次求解。
把数列 a 分成若干个长度不超过 $\lfloor\sqrt{n}\rfloor$ 的块，其中第 i 段的左端点为 $(i-1)\lfloor\sqrt{n}\rfloor+1$，右端点为 $\min(i\lfloor\sqrt{n}\rfloor, n)$，$a_i$ 所处的段的编号为 $\lfloor\frac{i-1}{\lfloor\sqrt{n}\rfloor}\rfloor+1$。

A Simple Problem with Integers

我们预处理出数组 sum，其中 sum_i 表示第 i 段的区间和，并用 add_i 来表示第 i 段的“增加标记”，就可以来处理这道题目了。

对于区间加法操作"C l r d"：

- 若 l 与 r 都在第 i 段内，则直接暴力处理，把 $a_l, a_{l+1}, \cdots, a_r$ 都加上 d，同时给 sum_i 加上 $(r-l+1)d$。
- 否则，设 l 处于第 p 段，r 处于第 q 段，对于中间的部分，也就是第 i($i \in [p,q]$) 段，我们更新 add_i；对于开头结尾的第 p、第 q 段同样暴力处理。

对于区间查询“Q l r”：

- 若 l 与 r 都在第 i 段内，则 $a_l + a_{l+1} + \cdots + a_r + (r-l+1) * add_i$ 就是答案，暴力枚举即可。
- 否则，设 l 处于第 p 段，r 处于第 q 段，初始化 $ans = 0$。对于 $i \in [p,q]$，把 ans 加上 $sum_i + add_i * len_i$，其中 len_i 表示第 i 段的长度。对于开头、结尾的两段，同样暴力枚举计算。

这种算法对于整段的修改用 add 标记记录，对于不足整段的修改采用朴素做法。因为段数和段长都是 $O(\sqrt{n})$ 级别的，所以整个算法的时间复杂度是 $O((n+m)\sqrt{n})$。大部分常见的分块思想都是这样，即“大段维护、局部朴素”。

例题

蒲公英

在乡下的小路旁边种着许多蒲公英，我们把所有的蒲公英看成一个长度为 n 的序列 $a_1, a_2, \cdots, a_n$。其中 a_i 为一个正整数，表示第 i 棵蒲公英的种类编号。

接下来有 m 次询问，每次询问一个区间 $[x, y]$，你需要回答 $a_x, a_{x+1}, \cdots, a_y$ 中出现次数最多的是哪种蒲公英。如果有若干种蒲公英出现次数相同，则输出种类编号最小的那个。

该题目强制在线。$1 \le n \le 4 \times 10^4, m \le 5 \times 10^5, a_i \le 10^9$。

例题

蒲公英

在乡下的小路旁边种着许多蒲公英，我们把所有的蒲公英看成一个长度为 n 的序列 $a_1, a_2, \cdots, a_n$。其中 a_i 为一个正整数，表示第 i 棵蒲公英的种类编号。
接下来有 m 次询问，每次询问一个区间 $[x, y]$，你需要回答 $a_x, a_{x+1}, \cdots, a_y$ 中出现次数最多的是哪种蒲公英。如果有若干种蒲公英出现次数相同，则输出种类编号最小的那个。
该题目强制在线。$1 \le n \le 4 \times 10^4, m \le 5 \times 10^5, a_i \le 10^9$。

本题是经典的区间众数问题。因为众数不具有“区间可加性”（不能通过区间 $[x, y]$ 和 $[y+1, z]$ 的众数直接得到 $[x, z]$ 的众数），所以用树状数组或线段树维护就非常困难。下面我们介绍两种常见的分块做法。
首先，如果把序列 a 分成 T 块，设 l 处于第 p 块，r 处于第 q 块，我们同样可以把区间 $[l, r]$ 分为三部分进行处理：开头不足一整段的 $[l, L)$；第 $p+1 \sim q-1$ 块构成的区间 $[L, R]$；结尾不足一整段的 $(R, r]$。
显然，a 序列在区间 $[l, r]$ 中的众数要么是区间 $[L, R]$ 的众数，要么是在 $[l, L), (R, r]$ 中出现过的数。我们需要想办法进行维护。

蒲公英

首先有个想法就是预处理出在所有可能的以“段边界”为端点的区间 $[L,R]$ 中，每个数出现的次数，以及区间的众数。由于序列 a 被分为了 T 块，所以这样的区间共有 T^2 个。并且保存每个数出现的次数需要长度为 $O(n)$ 的数组，记为 $cnt_{L,R}$。对于每个询问，我们可以先找到 $cnt_{L,R}$，再直接扫描 $[l,L),(R,r]$ 两段区间，在 $cnt_{L,R}$ 的基础上进行修改并统计答案。得到答案之后再把 $cnt_{L,R}$ 复原。
这样做的复杂度是 $O(nT^2+mn/T)$，空间为 $O(nT^2)$。如果设 n,m 数量级相同的话，通过方程 $nT^2=mn/T$，解得 $T\approx\sqrt[3]{n}$，整个算法的复杂度在 $O(n^{\frac{5}{3}})$ 这个级别。

蒲公英

之前的算法实在是太过暴力，没有很好地运用到众数的性质。由于区间 $[l,r]$ 中的众数要么是区间 $[L,R]$ 的众数，要么是在 $[l,L),(R,r]$ 中出现过的数，所以对于所有可能的以“段边界”为端点的区间 $[L,R]$，我们并不需要保存一个长度为 $O(n)$ 的 cnt 数组，只需要记录众数即可。之后扫描 $[l,L),(R,r]$ 中的数，判断它能否成为众数，这需要我们快速求出一个数在 $[l,r]$ 区间中出现的次数。

因此我们可以对每个数建立一个 $vector$，用于保存它在序列 a 中所有出现的位置。这样一来我们只需要在 $vector$ 中进行两次二分，就能知道一个数在 $[l,r]$ 中的出现次数了。

例如对于 $a=\{1,4,2,3,2,4,3,2,1,4\}$，数值 1 出现的位置为 $\{1,9\}$，数值 2 出现的位置为 $\{3,5,8\}$，数值 3 为 $\{4,7\}$，数值 4 为 $\{2,6,10\}$。若要求数值 2 在区间 $[2,8]$ 中的出现次数，就先在 $\{3,5,8\}$ 中二分找到第一个大于等于 2 的数 3，最后一个小于等于 8 的数 8，分别在 $\{3,5,8\}$ 中的第 1、第 3 个位置，所以数值 2 在区间 $[2,8]$ 中的出现次数就是 $3-1+1=3$ 次。

这个算法的时间复杂度为 $O(nT+mn/T*\log n)$，空间为 $O(T^2)$。同样方法解得 $T=\sqrt{n\log n}$，此时整个算法的复杂度在 $O(n\sqrt{n\log n})$ 级别。

例题

磁力块

在一片原野上，散落着 n 块磁石，每个磁石的性质可以用一个五元组 (x, y, m, p, r) 描述。其中 x, y 表示其坐标，m 是磁石的质量，p 是磁力，r 是吸引半径。
小 Q 带着一块自己的磁石 L 来到了 (x_0, y_0) 处。他手持磁石 L 并保持原地不动，所有可以被 L 吸引的磁石将会被吸引过来。在每个时刻，他可以选择更换一块自己已经获得的磁石（包括 L）在 (x_0, y_0) 处吸引更多的磁石。
磁石 A 可以吸引磁石 B 的条件是：磁石 A 位于 x_0, y_0，并且磁石 A 和磁石 B 的距离不大于磁石 A 的吸引半径，磁石 B 的质量不大于磁石 A 的磁力。特别地，只有小 Q 手持的磁石才会吸引别的磁石，原野上散落的磁石不会互相吸引。
现在小 Q 想知道，他最多能获得多少块磁石呢？$1 \leq n \leq 2.5 \times 10^5$。

例题

磁力块

在一片原野上，散落着 n 块磁石，每个磁石的性质可以用一个五元组 (x, y, m, p, r) 描述。其中 x, y 表示其坐标，m 是磁石的质量，p 是磁力，r 是吸引半径。
小 Q 带着一块自己的磁石 L 来到了 (x_0, y_0) 处。他手持磁石 L 并保持原地不动，所有可以被 L 吸引的磁石将会被吸引过来。在每个时刻，他可以选择更换一块自己已经获得的磁石（包括 L）在 (x_0, y_0) 处吸引更多的磁石。
磁石 A 可以吸引磁石 B 的条件是：磁石 A 位于 x_0, y_0，并且磁石 A 和磁石 B 的距离不大于磁石 A 的吸引半径，磁石 B 的质量不大于磁石 A 的磁力。特别地，只有小 Q 手持的磁石才会吸引别的磁石，原野上散落的磁石不会互相吸引。
现在小 Q 想知道，他最多能获得多少块磁石呢？$1 \le n \le 2.5 \times 10^5$。

我们先来考虑朴素的暴力如何做。建立一个队列存储小 Q 能拿到的磁石，每次从队首取出一块磁石，找到它能吸引到的所有磁石，并把不在队列中的都加入队列。这样做的话由于每次寻找“能够吸引到的所有磁石”需要 $O(n)$ 的复杂度，因此总复杂度为 $O(n^2)$。

磁力块

本题中，磁石之间的吸引需要满足两个条件，即质量 $\leq$ 磁力、距离 $\leq$ 吸引半径。这个问题用别的数据结构解决会比较复杂，分块算法就是一个很好的选择。
把 n 块磁石按照质量排序，分为 $\sqrt{n}$ 段。然后在每段内部，再按照到 (x_0, y_0) 的距离排序。
我们每次使用队首的磁石（设为 H）去吸引别的磁石时，由于我们先按照质量排序再分段，一定会存在一个 k，满足第 $1 \sim k-1$ 段中所有磁石质量都不大于 H 的质量，且第 $k+1$ 段之后的所有磁石质量都大于 H 的磁力，不可能被吸引。
而由于我们已经再每一段内部按照距离重新排序，所以第 $1 \sim k-1$ 段中与 (x_0, y_0) 的距离小于 H 吸引半径的磁石必定处于每段的开头部分。我们直接在这些段中从左到右扫描，把能吸引的磁石加入队尾，直至第一个不能被吸引的磁石（记为 P）。之后，我们把该段的开头位置移到 P 处。这样被吸引的磁石就不会被重复扫描，扫描过程的均摊复杂度为 $O(1)$，扫描 $k-1$ 段的复杂度为 $O(\sqrt{n})$。
而对于第 k 段，直接暴力扫描该段，把能吸引的磁石取走，并重构这个段。时间复杂度为 $O(\sqrt{n})$。
整个算法的复杂度为 $O(n\sqrt{n})$。

莫队

小 Z 的袜子

小 Z 有 n 只袜子，从 1 到 n 进行编号，每只袜子都有一种颜色。小 Z 无法忍受烦人的找袜子过程，因此他每次从编号 l 到 r 的袜子中随机选出两只来穿。
现在小 Z 想知道，他有多大的概率会抽到两只相同颜色的袜子。当然，小 Z 希望这个概率尽量大，所以他会询问 m 个 (l, r) 以便自己选择。
$n, m \leq 5 * 10^4$。

莫队

小 Z 的袜子

小 Z 有 n 只袜子，从 1 到 n 进行编号，每只袜子都有一种颜色。小 Z 无法忍受烦人的找袜子过程，因此他每次从编号 l 到 r 的袜子中随机选出两只来穿。
现在小 Z 想知道，他有多大的概率会抽到两只相同颜色的袜子。当然，小 Z 希望这个概率尽量大，所以他会询问 m 个 (l, r) 以便自己选择。
$n, m \leq 5 * 10^4$。

在本题中，我们会介绍分块算法的一种重要形式——对“询问”进行分块。这是一种离线做法，又被称为莫队算法。

莫队

我们先把 $[1,n]$ 这个序列分为 $\sqrt{n}$ 块，然后把所有询问 $[l,r]$ 读入，以左端点 l 所在的块为第一关键字，右端点 r 为第二关键字进行排序。
这样一来，如果相邻两个询问左端点在同一个块内，那么左端点的变化不超过 $\sqrt{n}$；最多只有 $\sqrt{n}$ 对相邻左端点不在同一个块内，变化幅度为 $O(n)$。所以左端点变化的范围之和为 $O(n\sqrt{n})$。右端点也是同理，由于左端点所在的块总数为 $\sqrt{n}$，每一块内右端点都是递增的，所以右端点的变化总量也是 $O(n\sqrt{n})$。
因此，只要我们能够在了解 $[l,r]$ 答案的情况下，$O(1)$ 拓展到 $[l-1,r],[l+1,r],[l,r-1],[l,r+1]$ 这四个相邻的区间，就可以使用莫队算法在 $O(n\sqrt{n})$ 的复杂度内求解。
回到这道题，我们先维护一个数组 num 并暴力处理第一个询问，num_c 表示颜色 c 在当前区间中出现的次数。另外我们再维护一个变量 ans，存储 $\sum_c num_c*(num_c-1)/2$。
之后，当我们从区间 $[l,r]$ 往相邻区间拓展时，在 num 进行加减，同时更新 ans 的值即可。抽到同色袜子的概率就是 ans/C_{r-l+1}^{2}。

对询问分块

而对于区间众数这类问题，由于只能从 $[l, r]$ 拓展到 $[l+1, r], [l, r+1]$，也就是区间只能扩大不能减小。我们如果采用莫队算法的话，就需要稍加改进。
注意到我们排序之后对于左端点的每一块，右端点是单增的，我们就还需要想办法让左端点只往左移动。这其实也挺好办，因为我们记录每个数出现次数的 num 数组是支持左端点右移的，只是 ans 不支持。所以对于每个询问，我们把左端点先放到这一块的最右侧，保存这时的答案，然后再往左移动并计算答案。之后直接把左端点移动回去并把 ans 也回滚，再移动右端点。
如果某类问题的区间只能减小不能扩大，也是采用同样的思路。

对询问分块

而对于区间众数这类问题，由于只能从 $[l,r]$ 拓展到 $[l+1,r],[l,r+1]$，也就是区间只能扩大不能减小。我们如果采用莫队算法的话，就需要稍加改进。
注意到我们排序之后对于左端点的每一块，右端点是单增的，我们就还需要想办法让左端点只往左移动。这其实也挺好办，因为我们记录每个数出现次数的 num 数组是支持左端点右移的，只是 ans 不支持。所以对于每个询问，我们把左端点先放到这一块的最右侧，保存这时的答案，然后再往左移动并计算答案。之后直接把左端点移动回去并把 ans 也回滚，再移动右端点。
如果某类问题的区间只能减小不能扩大，也是采用同样的思路。
对询问分块不仅限于莫队。以区间修改，区间查询为例，如果没有修改，我们可以通过前缀和在 $O(1)$ 的时间内得到区间和。
现在有了修改，我们也可以维护前缀和，然后把 m 个操作分成 $\sqrt{m}$ 块。我们每次进入新的一块，就重新计算一次前缀和数组，然后暴力枚举块内的修改对于当前询问的贡献。这样的复杂度也是 $(n+m)\sqrt{m}$ 级别的。