离线分治算法

李淳风

长郡中学

2024 年 7 月 8 日

介绍

大多数数据结构问题都可以抽象为“维护一系列结构，并对一系列操作依次做出相应”的形式。这些操作一般分为：统计数据信息的“查询”型操作，和更新数据状态的“修改”操作。其中，执行各项操作的“顺序”是一个重点，我们把它称为“时间轴”。
根据“查询”是否需要立即响应，可以把解决上述问题的算法分为“在线”和“离线”两类。在线算法需要在每次“查询”时立即返回结果，离线算法需要先知晓整个操作序列，最后再批量回答所有“查询”的结果。
根据两种操作在“时间轴”上分布的不同，也可以把上述问题分为“动态”和“静态”两类。静态问题只包含“查询”操作，或者一切“查询”操作都在一切“修改”操作之后。
CDQ 分治提供了一种把动态问题划分为若干个静态子问题，并使用离线算法进行求解的框架。

基于时间的分治算法

我们可以这样想：对于操作序列中的每个“查询”，计算结果等价于计算“初始数据”以及“之前的所有修改”对该查询的影响。实际上，我们可以把“初始数据”也看成若干次修改，这样可以方便我们思考。
设有 M 项操作，$\forall l, r \in [1, M]$，定义 $solve(l, r)$：$\forall k \in [l, r]$，若第 k 项操作是“查询”，则计算第 $l \sim k-1$ 项操作中的“修改”对它造成的影响。

- 设 $mid = (l + r) >> 1$，递归计算 $solve(l, mid)$。
- 递归计算 $solve(mid + 1, r)$。
- 计算第 $l \sim mid$ 项操作中的所有“修改”对第 $mid + 1 \sim r$ 项操作中所有“查询”造成的影响。

设第 k 项操作是“查询”。若 $k \leq mid$，那么 $solve(l, mid)$ 解决了第 $l \sim k-1$ 项中的“修改”对它的影响；若 $k > mid$，那么在 $[mid + 1, k - 1]$ 中的“修改”对它的影响在 $solve(mid + 1, r)$ 中处理了，在 $[l, mid]$ 中的“修改”对它的影响在第三步处理了。
这样我们就保证处理了第 $l \sim k-1$ 项中的“修改”对第 k 项的影响。

基于时间的分治算法

$solve(l,r)$ 的计算方法显然是一个分治的过程。$solve(1,M)$ 即为我们的初始问题；$l=r$ 只包含一项操作，不需要进行任何计算，是我们的递归边界。
那么，$solve(l,r)$ 的关键就在于第三部分的计算——左半部分的“修改”对右半部分的“查询”的影响。这一部分是一个静态问题，且原本的动态问题的答案由 $O(M)$ 个这样静态问题的答案组成。
如果我们能够在仅与 $r-l$ 的规模相关（与整个数据集的规模无关）的时间复杂度内解决 $solve(l,r)$ 的第三部分，那么整个算法的复杂度就是 $O(M\log M)$。而实际运用中我们往往需要在分治的过程中使用树状数组、线段树等数据结构，此时的复杂度为 $O(M\log^2 M)$。
这种离线分治算法是基于“时间”顺序对操作序列进行分治的，因此我们称它为基于时间的分治算法，又称为 CDQ 分治算法。

例题

天使玩偶

Ayu 在七年前把一个天使玩偶埋在了地下，现在她却忘了具体的位置了。所以 Ayu 决定凭借一点模糊的记忆来寻找它。
我们把 Ayu 生活的小镇看作一个二维平面直角坐标系，而 Ayu 会不定时地记起可能在某个点 (x, y) 埋下了天使玩偶。或者 Ayu 会询问你，加入她在 (x, y)，那么她离最近的天使玩偶可能埋下地地方有多远。
因为 Ayu 只会沿着平行坐标轴地方向来移动，所以我们定义两个点之间的距离为曼哈顿距离：$dis(A, B) = |A_x - B_x| + |A_y - B_y|$。
输入的第一行包含两个整数 n 和 m，表示在刚开始时，Ayu 已经知道有 n 个点可能埋着天使玩偶，接下来 Ayu 要进行 m 次操作。
接下来 n 行，每行两个整数 x_i, y_i，表示初始 n 个点的坐标。
再接下来 m 行，每行三个整数 t, x, y。
如果 $t = 1$，则表示 Ayu 又回忆起了一个可能埋着玩偶的点 (x, y)。
如果 $t = 2$，则表示 Ayu 询问如果她在坐标 (x, y)，那么在已经回忆出的点里，离她最近的那个点有多远。
数据范围为 $n, m \leq 5 \times 10^5$，坐标范围为 $0 \sim 10^6$。

天使玩偶

先来解决问题的简化版——假设没有 $t=1$ 的操作。这时，题目变为：在平面上有 n 个点 (x_i, y_i)，执行 m 次查询，每次询问与给定坐标 (x, y) 最近的点有多远。根据题意，此时答案为：

$$\min_{1\le i\le n}\{|x-x_i|+|y-y_i|\}$$

一般看到绝对值符号，都要想着通过分类讨论来去掉绝对值。我们不妨把原来的询问分成四个，分别询问在 (x, y) 左下、左上、右上、右下方向上距离最近的点有多远。四个结果取最小值即为答案。
以左下方向为例，此时式子变为：

$$\min_{1\le i\le n}\{x-x_i+y-y_i\}=(x+y)-\max_{1\le i\le n}\{x_i+y_i\},(x_i\le x, y_i\le y)$$

这时我们就可以把 n 个点的坐标和 m 个询问的坐标一起按照 x 坐标从小到大来排序，保证 $x_i\le x$。然后依次扫描，若扫描到一个点 (x_i, y_i)，则在树状数组/线段树上把 y_i 位置上的值与 x_i+y_i 取 $\max$；若扫描到一个询问 (x, y)，则在树状数组/线段树上查询 $[0, y]$ 上的最大值，并计算答案。

天使玩偶

现在带上了 $t=1$ 的操作，我们又应该如何处理呢？相当于在上个问题的基础上，随时可能在平面上加入一个点，变成了一个动态问题。
这样一来，我们的限制就不再仅仅是横、纵坐标有限制，对时间轴上的时间也有了限制。因此我们可以对整个操作序列应用 CDQ 分治，把动态问题再转化为静态问题。对于 $solve(l,r)$ 的第三部分，我们只需要把第 $l \sim mid$ 项操作中中新增的点依次加入一个初始为空的平面，然后对第 $mid+1 \sim r$ 项操作中的每个询问操作，找到平面上距离最近的点，更新答案。这个静态问题正是我们之前解决的问题，可以使用排序加数据结构来解决。
需要注意的是，为了保证时间复杂度仅与 $r-l$ 相关，清空树状数组时不能全部清空，只能在计算完 $solve(l,r)$ 之后，对树状数组/线段树的操作进行撤销，只清空用到的树状数组/线段树节点。整个算法的时间复杂度为 $O((n+m)\log^2(n+m))$。

三维偏序

三维偏序

给定一个有 n 个点的序列，每个点都有 a_i, b_i, c_i 三个属性。请问，这个序列中有多少个点对 (i, j)，满足 $a_i < a_j, b_i < b_j, c_i < c_j$ 且 $i \neq j$。保证 a, b, c 都是 $1 \sim n$ 的排列，$n \leq 10^5$。

三维偏序

三维偏序

给定一个有 n 个点的序列，每个点都有 a_i, b_i, c_i 三个属性。请问，这个序列中有多少个点对 (i, j)，满足 $a_i < a_j, b_i < b_j, c_i < c_j$ 且 $i \neq j$。
保证 a, b, c 都是 $1 \sim n$ 的排列，$n \leq 10^5$。

CDQ 分治是三维偏序的经典问题。
首先将序列按照 a 排好序，之后开始实现函数 $solve(l, r)$。先递归解决子问题 $solve(l, mid)$ 和 $solve(mid + 1, r)$，我们接下来要考虑的就是对于 $l \leq i \leq mid, mid < j \leq r$ 的点对 (i, j)，有多少满足条件。
由于我们先按照 a 排好序，那么肯定有 $a_i < a_j$，这个限制条件就可以不用考虑了。对于剩下的两个条件，不难发现其实已经变成了一个二维偏序问题。我们把 $[l, mid]$ 和 $[mid + 1, r]$ 中的点分别按照 b 排序，然后枚举所有的 j，把所有 $b_i < b_j$ 的点 i 全部插入到树状数组中，每次查询有多少个点的 c 值小于 c_j 即可。

整体分治

K-th Number

给定一个长度为 n 的整数序列 a，执行 m 次操作，其中第 i 次操作给出三个整数 l_i, r_i, k_i，求 $a_{l_i}, a_{l_i+1}, \cdots, a_{r_i}$ 中第 k_i 小的数是多少。
$n \leq 10^5, m \leq 10^4, |a_i| \leq 10^9$。

整体分治

K-th Number

给定一个长度为 n 的整数序列 a，执行 m 次操作，其中第 i 次操作给出三个整数 l_i, r_i, k_i，求 $a_{l_i}, a_{l_i+1}, \cdots, a_{r_i}$ 中第 k_i 小的数是多少。
$n \le 10^5, m \le 10^4, |a_i| \le 10^9$。

首先，如果只有第 i 次询问的话，我们可以在 a 的值域 $[MINA, MAXA]$ 上二分答案求解。设第一次二分的答案为 mid，在二分的过程当中，扫描在 $[l_i, r_i]$ 中不大于 mid 的数的个数，记为 c_i。
若 $k_i \le c_i$，则说明要找的数在 $[MINA, mid]$ 中；
若 $k_i > c_i$，则说明答案在 $[mid+1, MAXA]$ 中，等价于在 $[mid+1, MAXA]$ 中找第 $k_i - c_i$ 小的数。

整体分治

现在虽然有 m 次询问，但是我们可以发现，对于每次询问我们二分的结构都是一样的，因此我们可以把所有询问一起进行分治，每次分治根据 k_i 和 c_i 的关系把询问分为两类。
而且，我们最初的值域是 $[MINA, MAXA]$，此时所有 a 序列中的数都可能对答案有影响；二分一次之后，值域分别变为了 $[MINA, mid]$ 和 $[mid+1, MAXA]$，那么相应地，我们也可以把 a_i 按照 $\leq mid$ 和 $> mid$ 分为两个子序列 la 和 ra。
这样一来，在值域 $[MINA, mid]$ 中，我们只需要考虑 la 对第一类询问 $(k_i \leq c_i)$ 的影响；在值域 $[mid+1, MAXA]$ 中，我们只需要考虑 ra 对第二类询问 $(k_i > c_i)$ 的影响，这样就变成了两个子问题，可以递归进行求解。

整体分治

综上所述，我们得到一个分治算法。设 $solve(L, R, st, ed)$ 表示当前的值域区间为 $[L, R]$，需要处理操作序列数组 q 中的第 $st \sim ed$ 项（把初值也视为一个修改操作，同时询问也是操作）。

- 设 $mid = (L + R) >> 1$。
- 利用树状数组，扫描 q 中第 $st \sim ed$ 项，若遇到小于 mid 的修改 a_i 的操作，则利用树状数组中维护 i 位置的值加一；若遇到询问操作，则利用树状数组统计 $[l_i, r_i]$ 的区间和 c_i，也就是区间小于 mid 的数的个数。
- 扫描 q，对于每个询问操作，若 $k_i \leq c_i$，则把该询问加入到序列 lq 中；否则令 $k_i -= c_i$，加入到序列 rq 中；对于每个修改操作，若小于 mid 则加入 lq，否则加入 rq。
- 把 lq 和 rq 依次复制回 q 中 $[st, ed]$ 区间，设 lq 长度为 l_1。
- 递归求解 $solve(L, mid, st, st + l_1 - 1), solve(mid + 1, R, st + l_1, ed)$。

递归边界为：当 q 为空时，直接返回；当 $L = R$ 时，直接把 L 作为 q 中每个询问的答案。同时别忘了在递归求解之前通过撤销操作的方式，清空树状数组。
该算法时间复杂度 $O((n + m) \log^2 n)$。

整体分治

这个整体分治的算法碰到修改操作也可以同样处理。
把修改看作一次删除和一次添加，碰到删除小于等于 mid 的数是给 i 位置减一，添加则是加一，可以通过另外添加一个标记来区分。记得在递归儿子的时候，对 lq，rq 保留操作的时间顺序。