范围修改查询问题相关算法及其应用

沈吉濒

杭州学军中学教育集团文渊中学

123

一个经典问题

从一个经典问题引入：

问题

维护一个序列，序列上的每个元素为一个 *2*2* 矩阵，要求做以下操作：
修改操作：给定 l, r, v，将区间 $[l, r]$ 中的矩阵乘上矩阵 v。
查询操作：给定 l, r，查询区间 $[l, r]$ 的矩阵之和。

上述问题是线段树的经典问题，可以使用带懒惰标记的线段树解决。对于线段树上的每个节点，需要维护区间矩阵和，以及对区间中的矩阵乘的一个懒标记。
现在我们考虑抽象数据结构问题：不考虑具体维护什么信息，考虑在维护的信息满足某些特定条件下的通用问题形式。

对于线段树能维护的问题，抽象后的问题是：
维护一个序列，可以用某种数据表示序列上一个区间的信息，可以用某种数据表示序列区间的一次修改，需要满足的性质：
1. 序列上两个区间 $[l, mid], [mid+1, r]$ 的信息可以 $O(1)$ 合并出 $[l, r]$ 的信息。
2. 对一个区间进行的相邻两次修改操作可以 $O(1)$ 合并为一次修改操作。
3. 如果知道了一个区间的信息，可以 $O(1)$ 计算出这个区间经过一次修改后的信息。
矩阵的乘法满足结合律 $(A \times B) \times C = A \times (B \times C)$ 与分配律 $A \times (B + C) = A \times B + A \times C$，但不满足交换律。
在上面的问题中，信息的合并是矩阵加法；而结合律、分配律分别使这个问题能满足 2 和 3 的性质。

抽象信息的定义

定义半群为：
给定一个集合 $\mathcal{D}$，以及 $\mathcal{D}$ 上的一个二元运算：$\mathcal{D} * \mathcal{D} \to \mathcal{D}$。
运算满足结合律：$\forall a, b, c \in D, (a * b) * c = a * (b * c)$。

抽象信息的定义

定义半群为：
给定一个集合 $\mathcal{D}$，以及 $\mathcal{D}$ 上的一个二元运算：$\mathcal{D} * \mathcal{D} \to \mathcal{D}$。
运算满足结合律：$\forall a, b, c \in D, (a * b) * c = a * (b * c)$。
定义幺半群为：
给定一个集合 $\mathcal{D}$，以及 $\mathcal{D}$ 上的一个二元运算：$\mathcal{D} * \mathcal{D} \to \mathcal{D}$。
运算满足结合律：$\forall a, b, c \in D, (a * b) * c = a * (b * c)$。
有幺元：$\forall a \in D, a * \epsilon = \epsilon * a = a$。

抽象信息的定义

定义半群为：
给定一个集合 $\mathcal{D}$，以及 $\mathcal{D}$ 上的一个二元运算：$\mathcal{D} * \mathcal{D} \to \mathcal{D}$。
运算满足结合律：$\forall a, b, c \in D, (a * b) * c = a * (b * c)$。
定义幺半群为：
给定一个集合 $\mathcal{D}$，以及 $\mathcal{D}$ 上的一个二元运算：$\mathcal{D} * \mathcal{D} \to \mathcal{D}$。
运算满足结合律：$\forall a, b, c \in D, (a * b) * c = a * (b * c)$。
有幺元：$\forall a \in D, a * \epsilon = \epsilon * a = a$。
定义交换半群为：
给定一个集合 $\mathcal{D}$，以及 $\mathcal{D}$ 上的一个二元运算：$\mathcal{D} * \mathcal{D} \to \mathcal{D}$。
运算满足结合律：$\forall a, b, c \in D, (a * b) * c = a * (b * c)$。
运算满足交换律：$\forall a, b \in D, a * b = b * a$。

前面的例子维护的信息可以看作半群信息。
可以发现维护的是双半群信息，一个表示区间的信息，一个表示修改的懒标记的信息。
不难发现，只要区间的信息与修改的信息均为半群信息，就可以用线段树维护。

更广的范围

在问题中，修改和询问的范围不一定是序列的区间，而可能为树简单路径，二维平面矩形，高维正交范围，半平面范围，曲线范围等。

初始有 n 个点，一个范围代表了初始点集的一个子集。

可以用一个信息表示范围内点的信息和，用一个信息表示对范围的一个修改。这两个信息满足的条件是：

1. 两个范围的信息 S, T 可以 $O(1)$ 合并成一个信息 $S + T$。
2. 对一个范围的相邻的两次修改可以 $O(1)$ 合并成一次修改。
3. 如果知道了一个范围的信息，可以 $O(1)$ 计算出这个范围经过一次修改后的信息。

问题的形式

初始给出 n 个点的一个点集，每个点有一个初始权值。
每次修改给一个范围以及一个修改的值，表示把范围内的点的权值与修改权值做一个二元运算。
每次查询给一个范围，求范围内的点的权值求和后得到的值。其中点的权值和修改的权值满足双半群的性质。

问题的形式

形式化的问题如下：
给定交换半群 $(\mathcal{D},+)$，半群 $(\mathcal{M},*)$，二元运算符 $*$：
$\mathcal{D}*\mathcal{M}\to\mathcal{D}$。
其中二元运算符满足：
结合律：$\forall a\in\mathcal{D},b,c\in\mathcal{M},(a*b)*c=a*(b*c)$。
分配律：$\forall a,b\in\mathcal{D},c\in\mathcal{M},(a+b)*c=a*c+b*c$。
给定一个初始集合 I_0，一个有限集合 $I\subseteq I_0$，询问集合 Q。
I 中每个点有初始权值 $d_0(x)\in D$。
第 t 个操作定义为：
1. 范围修改：给出范围 $q_t\in Q$ 和 $f_t\in\mathcal{M}$，如果 $x\in q_t$ 则 $d_t(x)=d_{t-1}(x)*f_t$，否则 $d_t(x)=d_{t-1}(x)$。
2. 范围查询：给出范围 $q_t\in Q$，设 $d_t(x)=d_{t-1}(x)$，答案为 $\sum_{x\in q_t}d_t(x)$。

问题的形式

形式化的问题如下：
给定交换半群 $(\mathcal{D},+)$，半群 $(\mathcal{M},*)$，二元运算符 $*$：
$\mathcal{D}*\mathcal{M}\to\mathcal{D}$。
其中二元运算符满足：
结合律：$\forall a\in\mathcal{D},b,c\in\mathcal{M},(a*b)*c=a*(b*c)$。
分配律：$\forall a,b\in\mathcal{D},c\in\mathcal{M},(a+b)*c=a*c+b*c$。
给定一个初始集合 I_0，一个有限集合 $I\subseteq I_0$，询问集合 Q。
I 中每个点有初始权值 $d_0(x)\in D$。
第 t 个操作定义为：
1. 范围修改：给出范围 $q_t\in Q$ 和 $f_t\in\mathcal{M}$，如果 $x\in q_t$ 则 $d_t(x)=d_{t-1}(x)*f_t$，否则 $d_t(x)=d_{t-1}(x)$。
2. 范围查询：给出范围 $q_t\in Q$，设 $d_t(x)=d_{t-1}(x)$，答案为 $\sum_{x\in q_t}d_t(x)$。
用 $n=|I|$ 表示初始权重的个数，m 表示修改查询的操作数。

离线范围修改查询算法

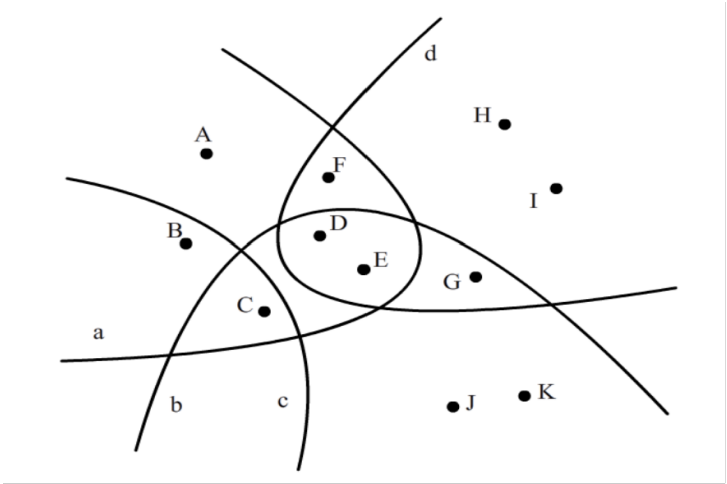
下面将介绍一种离线范围修改查询算法，由清华大学李欣隆与蔡承泽提出[1]。这个算法可以被简单实现，并且常数较小。可以证明这个算法的操作次数复杂度是最优的。

[1] Xinlong Li and Chengze Cai. Offline optimal range query and update algorithm. 2021.

等价关系

定义等价关系：
假设有一个操作序列 $q_1, q_2, ..., q_n$，定义点 x_1 与点 x_2 的等价关系：若 $R(x_1, x_2) = \forall_{i \in [1,n]}, x_1 \in q_i \leftrightarrow x_2 \in q_i$，其中 q_i 为第 i 次操作的范围，则称 x_1 与 x_2 等价。

等价类

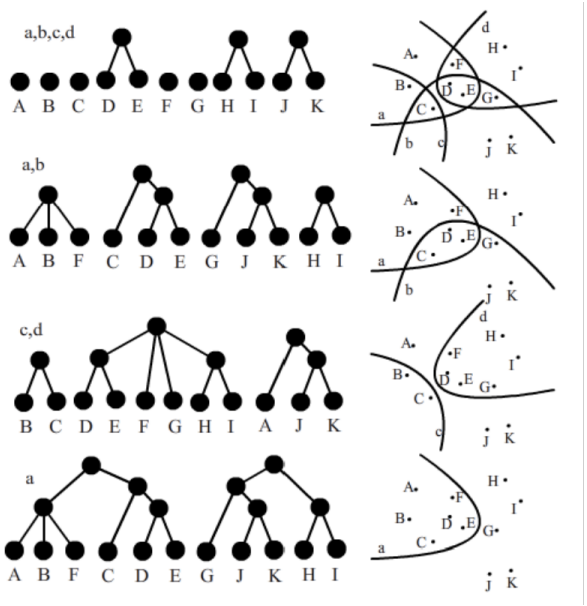


我们将等价的点归于一个等价类中，可以得到若干个等价类。
假设进行了 $q_l, q_{l+1}, \ldots, q_r$ 这些操作，设划分为的等价类集合为 $R_{l,r}$。
如果 $[l', r'] \in [l, r]$，则 $R_{l,r}$ 比 $R_{l',r'}$ 将点集划为更细的部分，称这个关系为 $R_{l,r} \subseteq R_{l',r'}$。

算法介绍

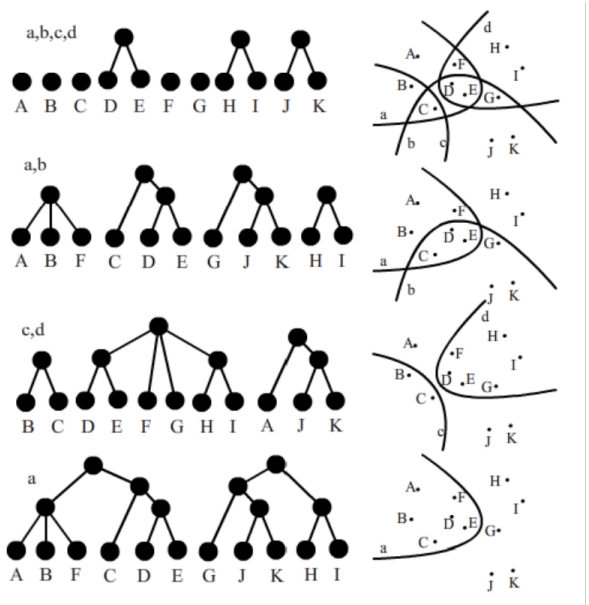
由于修改操作是给定的，我们考虑设计一个在修改操作序列上分治的算法。
在该问题中，考虑维护如下的一个动态有根森林：

- 每个节点都对应一个操作序列的某个等价类；
- 每个叶子节点存储的信息为大小为 1 的等价类的信息，即 n 个初始给定点各自的信息。



假设我们想将当前等价类集合从 R_1 变为 R_2，并满足 $R_1 \subseteq R_2$（R_2 的等价类个数更少）。

若我们对当前有根森林做若干次合并操作，每次合并若干有根树，就可以将状态 R_1 移动到目标状态 R_2。合并若干棵以 $x_1, x_2, ..., x_k$ 为根的有根树的方法是新建一个点 y 作为 $x_1, x_2, ..., x_k$ 的父亲节点。



同样，我们可以通过撤销状态 R_2 的若干次合并操作（即删除根节点），将其改变为 R_1 的状态。称这两种操作为有根森林之间的移动。

为了支持修改查询操作，考虑在有根森林的非叶子节点上维护额外的域。需要记录两个额外的值，一个存储修改的懒惰标记 $u(x)$，一个存储子树的和 $d(x)$（即为等价类中元素的和）。

为了支持修改查询操作，考虑在有根森林的非叶子节点上维护额外的域。需要记录两个额外的值，一个存储修改的懒惰标记 $u(x)$，一个存储子树的和 $d(x)$（即为等价类中元素的和）。
在有根森林加点时，我们会新建一个点 y 作为 $x_1, x_2, ..., x_k$ 的父亲，此时设置 $u(y) = \epsilon_M, d(y) = \sum_{i=1}^{k} d(x_i)$，即可以上传信息。

为了支持修改查询操作，考虑在有根森林的非叶子节点上维护额外的域。需要记录两个额外的值，一个存储修改的懒惰标记 $u(x)$，一个存储子树的和 $d(x)$（即为等价类中元素的和）。
在有根森林加点时，我们会新建一个点 y 作为 $x_1, x_2, ..., x_k$ 的父亲，此时设置 $u(y) = \epsilon_M, d(y) = \sum_{i=1}^{k} d(x_i)$，即可以上传信息。
在有根森林删点时，需要删除点 y 并分裂出 $x_1, x_2, ..., x_k$ 这 k 棵有根树，此时需要下放点 y 的懒惰标记并删除节点 y。

为了支持修改查询操作，考虑在有根森林的非叶子节点上维护额外的域。需要记录两个额外的值，一个存储修改的懒惰标记 $u(x)$，一个存储子树的和 $d(x)$（即为等价类中元素的和）。
在有根森林加点时，我们会新建一个点 y 作为 $x_1, x_2, ..., x_k$ 的父亲，此时设置 $u(y) = \epsilon_M, d(y) = \sum_{i=1}^{k} d(x_i)$，即可以上传信息。
在有根森林删点时，需要删除点 y 并分裂出 $x_1, x_2, ..., x_k$ 这 k 棵有根树，此时需要下放点 y 的懒惰标记并删除节点 y。
如果有一棵以 x 为根的树，其包含的叶子节点恰好是一次修改或查询中包含的点，则 $d(x)$ 即为本次询问的答案，修改则可以对 x 打上标记 $(d(x), u(x)) \to (d(x) * U, u(x) * U)$。

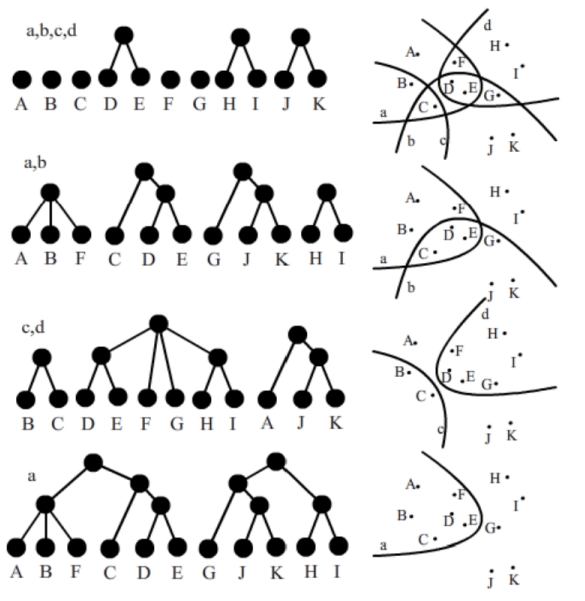
对于原问题，考虑分治。将当前操作序列分成左右两部分。对于两部分操作，由于操作数量更少，拥有的等价类个数也更少，这意味着分治后的两个区间都只需要处理更少的信息。

对于原问题，考虑分治。将当前操作序列分成左右两部分。对于两部分操作，由于操作数量更少，拥有的等价类个数也更少，这意味着分治后的两个区间都只需要处理更少的信息。
当我们需要进行 $q_l, ..., q_r$ 的分治时，设 $mid = \lfloor \frac{l+r}{2} \rfloor$，先将等价类集合从 $R_{l,r}$ 移动到 $q_l, ..., q_{mid}$ 的状态 $R_{l,mid}$，这会在有根森林上加若干个点。处理完 $q_l, ..., q_{mid}$ 后，将状态回退到 $R_{l,r}$，然后移动到 $R_{mid+1,r}$，对 $q_{mid+1}, ..., q_r$ 执行分治算法，再将状态回退到 $R_{l,r}$。

对于原问题，考虑分治。将当前操作序列分成左右两部分。对于两部分操作，由于操作数量更少，拥有的等价类个数也更少，这意味着分治后的两个区间都只需要处理更少的信息。

当我们需要进行 $q_l, ..., q_r$ 的分治时，设 $mid = \lfloor \frac{l+r}{2} \rfloor$，先将等价类集合从 $R_{l,r}$ 移动到 $q_l, ..., q_{mid}$ 的状态 $R_{l,mid}$，这会在有根森林上加若干个点。处理完 $q_l, ..., q_{mid}$ 后，将状态回退到 $R_{l,r}$，然后移动到 $R_{mid+1,r}$，对 $q_{mid+1}, ..., q_r$ 执行分治算法，再将状态回退到 $R_{l,r}$。

当递归到 $l = r$ 时只有一个操作，也就只有 2 个等价类。修改就在需要操作的等价类对应的有根树的根上打上标记，查询则提取该根节点 d 的信息即可。



图中演示了等价关系为 $R_{1,4}, R_{1,2}, R_{3,4}, R_{1,1}$ 时有根森林的结构。

我们从 $R_{1,4}$ 移动到 $R_{1,2}$ 时，将等价类 $\{A\}, \{B\}, \{F\}$ 合并为 $\{A, B, F\}$；$\{C\}, \{D, E\}$ 合并为 $\{C, D, E\}$；$\{G\}, \{J, K\}$ 合并为 $\{G, J, K\}$。从 $R_{1,2}$ 移动回 $R_{1,4}$ 时，撤销上述合并。
递归到 $R_{1,1}$ 时，需要修改的树即为 a 树，代表了等价类，设 x 为 $\{A, B, C, D, E, F\}$ 即 a 树的根，我们打上修改标记并查询 x 的值。

算法的复杂度

定义离散函数 F，$F(x)$ 表示 x 个操作最多将点集分成多少个等价类。
不难发现 $F(x)$ 是单调递增的，且 x 个操作最多将集合 I 分成 $\min(F(x), |I|)$ 个等价类。
定义离散函数 G，$G(n)$ 满足 $F(G(n)) \leq n, F(G(n)+1) > n$。
$G(n)$ 表示在给定范围下，至少需要多少个操作能把 n 个点分成最细的等价关系，即 n 个等价类。

假设有 n 个点与 m 次操作，我们将 m 个操作分成每 $G(n)$ 个操作一组，在每组中使用上述分治算法，在代数结构上的运算次数为
$T(n,m) = O(n) + \frac{m}{G(n)} T_1(G(n)), T_1(n) = 2T_1(n/2) + O(F(n))$。

假设有 n 个点与 m 次操作，我们将 m 个操作分成每 $G(n)$ 个操作一组，在每组中使用上述分治算法，在代数结构上的运算次数为
$T(n,m)=O(n)+\frac{m}{G(n)}T_1(G(n)),T_1(n)=2T_1(n/2)+O(F(n))$。
在序列上的问题中，有 $F(n)=O(n)$，此时
$T(n,m)=O(n+m\log\min(n,m))$。

假设有 n 个点与 m 次操作，我们将 m 个操作分成每 $G(n)$ 个操作一组，在每组中使用上述分治算法，在代数结构上的运算次数为
$T(n,m)=O(n)+\frac{m}{G(n)}T_1(G(n)),T_1(n)=2T_1(n/2)+O(F(n))$。
在序列上的问题中，有 $F(n)=O(n)$，此时
$T(n,m)=O(n+m\log\min(n,m))$。
对于修改查询范围为半平面，多边形，圆，椭圆，双曲线等情况，有 $F(n)=O(n^2)$，此时 $T(n,m)=O(n+m\sqrt{n})$。

假设有 n 个点与 m 次操作，我们将 m 个操作分成每 $G(n)$ 个操作一组，在每组中使用上述分治算法，在代数结构上的运算次数为
$T(n,m) = O(n) + \frac{m}{G(n)} T_1(G(n)), T_1(n) = 2T_1(n/2) + O(F(n))$。
在序列上的问题中，有 $F(n) = O(n)$，此时
$T(n,m) = O(n + m \log \min(n,m))$。
对于修改查询范围为半平面，多边形，圆，椭圆，双曲线等情况，有 $F(n) = O(n^2)$，此时 $T(n,m) = O(n + m\sqrt{n})$。
对于修改查询范围为 d 维的空间范围时，其中 $d > 1$，有 $F(n) = O(n^d)$，此时 $T(n,m) = O(n + mn^{1-1/d})$。

假设有 n 个点与 m 次操作，我们将 m 个操作分成每 $G(n)$ 个操作一组，在每组中使用上述分治算法，在代数结构上的运算次数为
$T(n,m)=O(n)+\frac{m}{G(n)}T_1(G(n)),T_1(n)=2T_1(n/2)+O(F(n))$。
在序列上的问题中，有 $F(n)=O(n)$，此时
$T(n,m)=O(n+m\log\min(n,m))$。
对于修改查询范围为半平面，多边形，圆，椭圆，双曲线等情况，有 $F(n)=O(n^2)$，此时 $T(n,m)=O(n+m\sqrt{n})$。
对于修改查询范围为 d 维的空间范围时，其中 $d>1$，有 $F(n)=O(n^d)$，此时 $T(n,m)=O(n+mn^{1-1/d})$。
特别的，若修改查询范围为点集的任意子集，有 $F(n)=O(2^n)$，此时 $G(n)=O(\log n)$，可得 $T(n,m)=O(n+m\frac{n}{\log n})$。这意味着修改查询为任意子集可以做到比暴力更优的复杂度。

[CTS2021] 测测你的半平面修改查询[2]

给定交换半群 $(\mathcal{D},+)$，半群 $(\mathcal{M},*)$，二元运算符 $*$：
$\mathcal{D}*\mathcal{M}\to\mathcal{D}$，其中 $*$ 满足结合律与分配律。
给定平面上的 n 个点 (x_i,y_i)，每个点有初始权值 $d_i\in\mathcal{D}$。
需要实现 m 个操作：
第 j 次操作给出 a_j,b_j,c_j 与半群信息 $o_j\in\mathcal{M}$。
对于所有满足 $a_jx_i+b_jy_i<c_j$ 的 i，需要先回答 $\sum_i d_i$，后将所有 d_i 修改为 o_j*d_i。
操作离线。

[2]https://qoj.ac/problem/9020,https://qoj.ac/problem/4817

对操作序列每 B 个分一组进行分治，并应用上面的算法，设 $B = \sqrt{n}$，可以得到操作次数复杂度为 $O(n + m\sqrt{n})$。
接下来我们要处理的问题就是计算几何问题，可以使用平面图点定位的算法。

对操作序列每 B 个分一组进行分治，并应用上面的算法，设 $B=\sqrt{n}$，可以得到操作次数复杂度为 $O(n+m\sqrt{n})$。
接下来我们要处理的问题就是计算几何问题，可以使用平面图点定位的算法。
具体的，对于 $\sqrt{m}$ 个询问的直线，处理出每两条直线的交点，并将交点按照 x 坐标排序后扫描线。
扫描线过程中，实时维护出所有直线在当前 x 坐标下，按照 y 坐标排序后的顺序，在遇到一个交点后就交换两条直线的顺序。
同时对所有点按 x 坐标排序，在扫描到该点时二分定位出该点属于哪个区域。这样就得到了 $O(m)$ 个区域的初始信息。

对于每条直线，将直线上的交点按 x 坐标增大排序，维护一个链表，每个交点属于两条直线的链表。

对于每条直线，将直线上的交点按 x 坐标增大排序，维护一个链表，每个交点属于两条直线的链表。
当删除该直线时，遍历链表上的每个交点，在该交点所属的另一个链表中删除该点，并合并两边的区域。

对于每条直线，将直线上的交点按 x 坐标增大排序，维护一个链表，每个交点属于两条直线的链表。
当删除该直线时，遍历链表上的每个交点，在该交点所属的另一个链表中删除该点，并合并两边的区域。
这样就完成了删直线并合并区域的过程。撤回删掉的直线时，也同样遍历该直线的链表，依次撤回操作即可。

对于每条直线，将直线上的交点按 x 坐标增大排序，维护一个链表，每个交点属于两条直线的链表。
当删除该直线时，遍历链表上的每个交点，在该交点所属的另一个链表中删除该点，并合并两边的区域。
这样就完成了删直线并合并区域的过程。撤回删掉的直线时，也同样遍历该直线的链表，依次撤回操作即可。
这样做的时间复杂度为 $O(n + m\sqrt{n}\log n)$，瓶颈在对交点排序与扫描线二分。不过该题是交互题，只需要保证操作次数正确即可。

问题的形式

在该问题中，没有修改操作，只有查询操作，查询为半平面的一侧的交换半群信息。
若可以离线，则使用上述的修改查询算法，就可以达到运算次数的下界 $O(n + m\sqrt{n})$。
但在仅有查询，没有修改的情况下，有另一些复杂度可能更低、或可以强制在线的算法。下面将介绍一些针对该问题的算法。

旋转扫描线

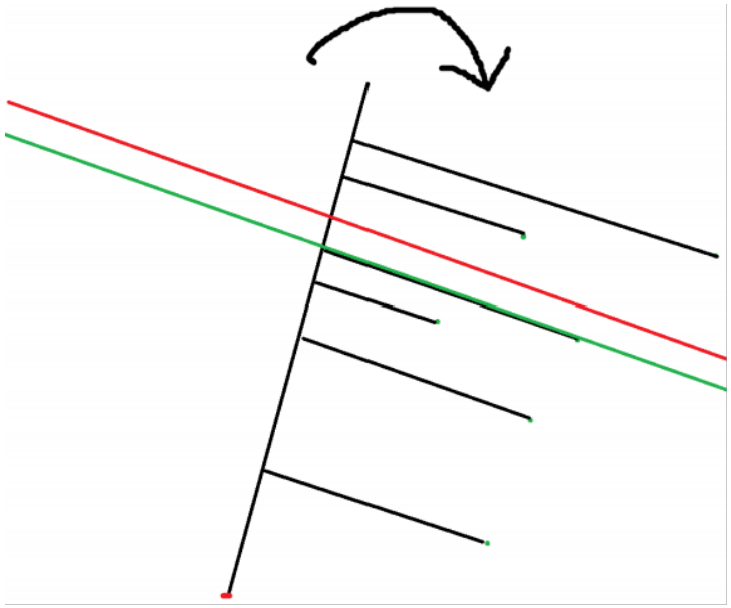
将点集随意划分为大小为 B 的块。

旋转扫描线

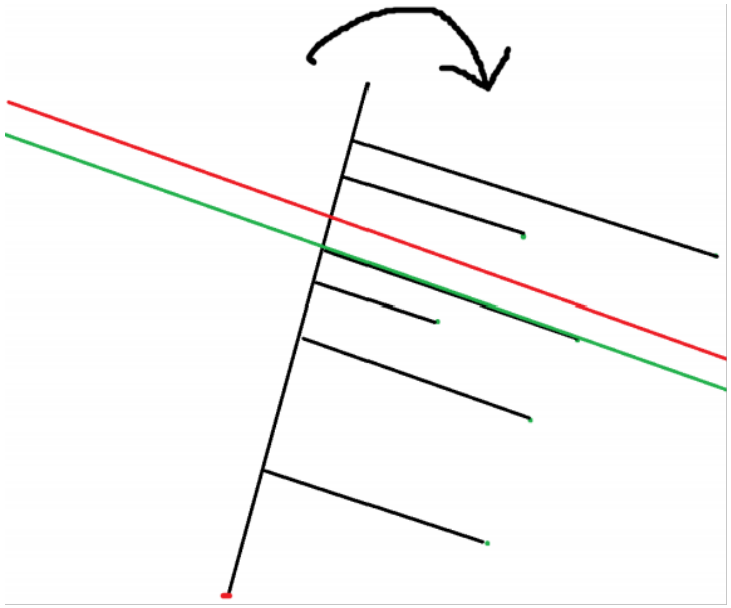
将点集随意划分为大小为 B 的块。
定义半平面的标准型为将询问的半平面先一直向下平移，直到碰到第一个点，之后将询问的半平面一直顺时针方向旋转，直到碰到第一个点。可以证明这样的变换是不会改变这个半平面一侧的点集的。

旋转扫描线

将点集随意划分为大小为 B 的块。
定义半平面的标准型为将询问的半平面先一直向下平移，直到碰到第一个点，之后将询问的半平面一直顺时针方向旋转，直到碰到第一个点。可以证明这样的变换是不会改变这个半平面一侧的点集的。
在这样的变换下，在 B 个点的情况下，只存在 $O(B^2)$ 个不同的半平面的标准型。如果两个半平面变换到了相同的标准型，则这两个平面对应同一个点集，询问的答案相同，即只有 $O(B^2)$ 个需要预处理的答案。



旋转扫描线的过程为：假设当前有一条直线，斜率从 $-\infty$ 逐渐移向 $+\infty$，在过程中维护当前所有点的顺序。



旋转扫描线的过程为：假设当前有一条直线，斜率从 $-\infty$ 逐渐移向 $+\infty$，在过程中维护当前所有点的顺序。
对于一次半平面范围询问，可以在扫描线扫到对应斜率时，在点的序列上二分求出前驱后继。然后相当于要在序列上支持前缀，后缀查询。动态维护当前旋转扫描线的序列，每次交换的元素都是相邻的，可以 $O(1)$ 进行修改。

取 $B=\sqrt{m}$，代数结构上的运算次数为 $O(n\sqrt{m})$，总时间复杂度 $O(n\sqrt{m}\log n)$。时间复杂度的瓶颈在于对斜率的排序以及点定位的二分。

注意这里复杂度与修改查询问题的区别：范围修改查询问题的复杂度下界为 $O(n+m\sqrt{n})$，不会因为操作多而降低复杂度；而仅支持查询的复杂度可以做到 $O(n\sqrt{m})$，会因为操作多而降低复杂度。

此外，还有一些其他的处理范围修改查询问题的方法，如最优点集划分树、平面分块等。由于时间所限，在这里不进行展开，可以阅读我的集训队论文。

致谢

感谢大家的聆听。
感谢中国计算机学会提供学习和交流的平台。
感谢国家集训队教练彭思进、杨耀良的指导。
感谢家人对我的陪伴与支持。
感谢学军中学徐先友老师的关心与指导。
感谢李欣隆学长、蔡承泽学长与我交流讨论、给我启发。
感谢学军中学的同学们为本文验稿。
感谢其他给予我帮助的老师与同学。

Thanks for listening!