浅谈分块的各类应用

丁晓漫

NFLS

May 21, 2021

Contents

1. 询问分块
2. 序列分块
3. 莫队
   - 普通莫队
   - 回滚莫队
   - 二次离线莫队
4. 根号分治
   - 按点的度数分治
   - 按字符串长度分治
   - 按数字出现次数分治
5. 分治结合分块
6. 总结
   - 致谢

分块思想

处理问题时，将维护的对象按某种标准分成若干块，并对整块和散点用不同的高效方法维护。

例题[1]

给定一个 n 个点 m 条边的无向图，每条边有边权。需要支持 q 次以下操作：

- 修改一条边的边权
- 给定起点 u 和权值 v，询问从点 u 开始走，只能经过边权 $\geq v$ 的边，能到达多少个不同的点

[1] [APIO2019] 桥梁

例题[1]

给定一个 n 个点 m 条边的无向图，每条边有边权。需要支持 q 次以下操作：

- 修改一条边的边权
- 给定起点 u 和权值 v，询问从点 u 开始走，只能经过边权 $\geq v$ 的边，能到达多少个不同的点

  $n \leq 5 \times 10^4, m, q \leq 10^5$

[1] [APIO2019] 桥梁

分析性质

连通性考虑使用并查集维护。

分析性质

中国计算机学会 China Computer Federation

连通性考虑使用并查集维护。

如果不存在修改边权的操作，可以离线所有询问并把询问按 v 从大到小排序，把边也按边权从大到小排序。回答每个询问时并查集里加入原图中边权 $\geq v$ 的边，找到 u 所在连通块的大小即可。

分析性质

连通性考虑使用并查集维护。

如果不存在修改边权的操作，可以离线所有询问并把询问按 v 从大到小排序，把边也按边权从大到小排序。回答每个询问时并查集里加入原图中边权 $\geq v$ 的边，找到 u 所在连通块的大小即可。

如果修改边权的操作较少，记 S 为被修改过的边构成的集合，E 为所有边的集合。同样离线所有询问并按 v 从大到小排序，对于 $E \setminus S$ 中的边直接加入并查集。回答询问时遍历所有 S 中的边，如果在这个询问时这条边的边权 $\geq v$ 就加入并查集，即可回答询问。回答之后需要撤销最后加入的 S 中的边，需要可撤销并查集。

询问分块算法

发现我们会处理修改较少的情况，那么可以考虑询问分块，也就是对于操作序列分块。

询问分块算法

发现我们会处理修改较少的情况，那么可以考虑询问分块，也就是对于操作序列分块。

取正整数 B，并将操作 B 个分为一块。现在对于某一块，考虑回答这一块里的所有询问。

询问分块算法

发现我们会处理修改较少的情况，那么可以考虑询问分块，也就是对于操作序列分块。

取正整数 B，并将操作 B 个分为一块。现在对于某一块，考虑回答这一块里的所有询问。

对于这一块里的所有询问来说，在这一块之前和之后的修改涉及到的边都可以视作不被修改，那么我们关心的被修改的集合 $|S| = O(B)$ 。只需要对于这一块里的询问执行修改操作较少的算法即可。

询问分块算法

发现我们会处理修改较少的情况，那么可以考虑询问分块，也就是对于操作序列分块。

取正整数 B，并将操作 B 个分为一块。现在对于某一块，考虑回答这一块里的所有询问。

对于这一块里的所有询问来说，在这一块之前和之后的修改涉及到的边都可以视作不被修改，那么我们关心的被修改的集合 $|S| = O(B)$ 。只需要对于这一块里的询问执行修改操作较少的算法即可。

对于每一块来说，需要将不被修改的边权排序并用并查集维护，复杂度为 $O(\frac{q}{B} m \log_2 m)$；对于每一次询问，需要遍历集合 S 并用可撤销并查集维护，复杂度为 $O(qB \log_2 n)$。把 $\log_2 n$ 和 $\log_2 m$ 视作同阶，分析可得 B 取 $\sqrt{m}$ 时达到理论最优复杂度为 $O(q\sqrt{m} \log_2 n)$。

思考

中国计算机学会 China Computer Federation

上述例题中，将边权的排序改为不变的边权和变化的边权的归并排序，对于 S 中的加边看作在 $O(B)$ 个点上的新图，每次 dfs 找能到达的连通块，可以把复杂度降为 $O(q\sqrt{m}\alpha(n))$。

思考

上述例题中，将边权的排序改为不变的边权和变化的边权的归并排序，对于 S 中的加边看作在 $O(B)$ 个点上的新图，每次 dfs 找能到达的连通块，可以把复杂度降为 $O(q\sqrt{m}\alpha(n))$。

如果没有修改的询问容易回答，少数修改的询问也不难回答，可以考虑尝试询问分块。

例题[2]

给定一个长度为 n 的序列 a_i，保证 $a_i \neq 0$。

[2] [2017-2018 ACM-ICPC, NEERC, Northern Subregional Contest] Joker

例题[2]

给定一个长度为 n 的序列 a_i，保证 $a_i \neq 0$。

令这 n 个数中所有正数的和为 P，所有负数的和为 N。对于 $1 \leq i \leq n$，如果 $a_i > 0$, 令 $w_i = \frac{a_i}{P}$，否则 $w_i = \frac{a_i}{|N|}$ 。

[2] [2017-2018 ACM-ICPC, NEERC, Northern Subregional Contest]Joker

例题[2]

给定一个长度为 n 的序列 a_i，保证 $a_i \neq 0$。

令这 n 个数中所有正数的和为 P，所有负数的和为 N。对于 $1 \leq i \leq n$，如果 $a_i > 0$, 令 $w_i = \frac{a_i}{P}$，否则 $w_i = \frac{a_i}{|N|}$ 。

令 $s_i = \sum_{j=1}^{i} w_j$。给定 q 次修改，每次改变一个位置上的 a_i，需要在第一次修改之前和每次修改之后求出 s_i 最大的位置，如果有多个最大值选择下标最小的。

[2] [2017-2018 ACM-ICPC, NEERC, Northern Subregional Contest] Joker

例题[2]

给定一个长度为 n 的序列 a_i，保证 $a_i \neq 0$。

令这 n 个数中所有正数的和为 P，所有负数的和为 N。对于 $1 \leq i \leq n$，如果 $a_i > 0$, 令 $w_i = \frac{a_i}{P}$，否则 $w_i = \frac{a_i}{|N|}$ 。

令 $s_i = \sum_{j=1}^{i} w_j$。给定 q 次修改，每次改变一个位置上的 a_i，需要在第一次修改之前和每次修改之后求出 s_i 最大的位置，如果有多个最大值选择下标最小的。

$n, m \leq 5 \times 10^4$

[2] [2017-2018 ACM-ICPC, NEERC, Northern Subregional Contest] Joker

性质分析

记 $w'_i = w_i \times P \times |N|, s'_i = \sum_{j=1}^{i} w'_j = s_i \times P \times |N|$，不难发现 $s'_i = (\sum_{j=1}^{n} a_j[a_j > 0]) \times |N| - (\sum_{j=1}^{n} -a_j[a_j < 0]) \times P$，是叉积的形式。

性质分析

记 $w'_i = w_i \times P \times |N|, s'_i = \sum_{j=1}^{i} w'_j = s_i \times P \times |N|$，不难发现 $s'_i = (\sum_{j=1}^{n} a_j[a_j > 0]) \times |N| - (\sum_{j=1}^{n} -a_j[a_j < 0]) \times P$，是叉积的形式。

令 p_i 为二维平面上的点。如果 $a_i > 0$，令 $p_i = (a_i, 0)$，否则 $p_i = (0, -a_i)$，且记 $Q = (\sum_{j=1}^{n} a_j[a_j > 0], \sum_{j=1}^{n} -a_j[a_j < 0])$，那么即是要求 $\det(\sum_{j=1}^{i} p_j, Q)$ 的最大值。

性质分析

记 $w'_i = w_i \times P \times |N|, s'_i = \sum_{j=1}^{i} w'_j = s_i \times P \times |N|$，不难发现 $s'_i = (\sum_{j=1}^{n} a_j[a_j > 0]) \times |N| - (\sum_{j=1}^{n} -a_j[a_j < 0]) \times P$，是叉积的形式。

令 p_i 为二维平面上的点。如果 $a_i > 0$，令 $p_i = (a_i, 0)$，否则 $p_i = (0, -a_i)$，且记 $Q = (\sum_{j=1}^{n} a_j[a_j > 0], \sum_{j=1}^{n} -a_j[a_j < 0])$，那么即是要求 $\det(\sum_{j=1}^{i} p_j, Q)$ 的最大值。

联想到叉积最大值一定在凸包上，只要维护所有 $\sum_{j=1}^{i} p_j$ 的凸包即可二分找到叉积最大值的位置。

序列分块算法

发现修改时需要 $O(n)$ 重构（由于 p_i 两维都非负，$\sum_{j=1}^{i} p_j$ 的两维都单调不降不需要排序），查询需要 $O(\log_2 n)$，复杂度不太平衡，考虑通过序列分块降低维护的复杂度，适当增加查询的复杂度。

序列分块算法

发现修改时需要 $O(n)$ 重构（由于 p_i 两维都非负，$\sum_{j=1}^{i} p_j$ 的两维都单调不降不需要排序），查询需要 $O(\log_2 n)$，复杂度不太平衡，考虑通过序列分块降低维护的复杂度，适当增加查询的复杂度。

取正整数 B 并把序列每长度 B 分为一块，对于每块维护块内 p_i 前缀和的凸包。每次修改时只需要重构修改位置所在块的凸包，查询时枚举最大位置所在的块，每块在凸包上二分查询。

序列分块算法

发现修改时需要 $O(n)$ 重构（由于 p_i 两维都非负，$\sum_{j=1}^{i} p_j$ 的两维都单调不降不需要排序），查询需要 $O(\log_2 n)$，复杂度不太平衡，考虑通过序列分块降低维护的复杂度，适当增加查询的复杂度。

取正整数 B 并把序列每长度 B 分为一块，对于每块维护块内 p_i 前缀和的凸包。每次修改时只需要重构修改位置所在块的凸包，查询时枚举最大位置所在的块，每块在凸包上二分查询。

每次重构凸包复杂度为 $O(B)$，查询复杂度为 $O(\frac{n}{B}\log_2 B)$。认为 $\log_2 n$ 和 $\log_2 B$ 同阶的话，取 $B=\sqrt{n\log_2 n}$ 可以得到最优复杂度 $O(q\sqrt{n\log_2 n})$。

例题[3]

平面上有 n 个点，第 i 个点为 (x_i, y_i)。

[3] [UNR #4] 己酸集合

例题[3]

平面上有 n 个点，第 i 个点为 (x_i, y_i)。

有 q 次询问，每次询问给定整数 z 和 R，需要回答到点 $(0, z)$ 的欧几里得距离不超过 R 的点数。

[3] [UNR #4] 已酸集合

例题[3]

平面上有 n 个点，第 i 个点为 (x_i, y_i)。

有 q 次询问，每次询问给定整数 z 和 R，需要回答到点 $(0, z)$ 的欧几里得距离不超过 R 的点数。

$n \leq 12000, q \leq 10^6$

[3] [UNR #4] 已酸集合

性质分析

条件即为 $x_i^2+(y_i-z)^2\le R^2$，整理得 $x_i^2+y_i^2-2y_iz\le R^2-z^2$。对于点 (x_i,y_i)，令 $f_i(z)=-2y_iz+x_i^2+y_i^2$。将每个询问看作点 (z,R^2-z^2)，即是要回答某个点下方直线的条数。

性质分析

条件即为 $x_i^2+(y_i-z)^2\le R^2$，整理得 $x_i^2+y_i^2-2y_iz\le R^2-z^2$。对于点 (x_i,y_i)，令 $f_i(z)=-2y_iz+x_i^2+y_i^2$。将每个询问看作点 (z,R^2-z^2)，即是要回答某个点下方直线的条数。

发现对于任意 $f_i(z)$ 和 $f_j(z)$，它们的相对大小关系随着 z 增大至多变化一次，这说明所有 $f_i(z)$ 的相对大小关系随着 z 的增大至多变化 $O(n^2)$ 次。预处理这 $O(n^2)$ 段，每次询问二分得到 z 所在的段，再二分得到这个点下方的直线条数。

性质分析

条件即为 $x_i^2+(y_i-z)^2\le R^2$，整理得 $x_i^2+y_i^2-2y_iz\le R^2-z^2$。对于点 (x_i,y_i)，令 $f_i(z)=-2y_iz+x_i^2+y_i^2$。将每个询问看作点 (z,R^2-z^2)，即是要回答某个点下方直线的条数。

发现对于任意 $f_i(z)$ 和 $f_j(z)$，它们的相对大小关系随着 z 增大至多变化一次，这说明所有 $f_i(z)$ 的相对大小关系随着 z 的增大至多变化 $O(n^2)$ 次。预处理这 $O(n^2)$ 段，每次询问二分得到 z 所在的段，再二分得到这个点下方的直线条数。

序列分块算法

发现预处理需要 $O(n^2 \log_2 n)$，查询只需要 $O(q \log_2 n)$，考虑用序列分块平衡这两部分的复杂度。

序列分块算法

发现预处理需要 $O(n^2 \log_2 n)$，查询只需要 $O(q \log_2 n)$，考虑用序列分块平衡这两部分的复杂度。

取正整数 B 并把 n 条直线按任意顺序 B 条分为一块。对于每块执行上述算法，预处理部分复杂度为 $O(Bn \log_2 B)$，查询时答案为在每一块里查询的答案之和，复杂度为 $O(\frac{n}{B} q \log_2 B)$。将 $\log_2 B$ 和 $\log_2 n$ 视为同阶，$B = \sqrt{q}$ 时复杂度最优为 $O(n\sqrt{q} \log_2 n)$。

例题[4]

有 n 个 std::queue<int>，编号从 1 到 n。第 i 个序列的容量为 a_i。

[4] [2018 集训队互测 Day 1] 完美的队列

例题[4]

有 n 个 std::queue<int>，编号从 1 到 n。第 i 个序列的容量为 a_i。

初始 n 个序列都是空的。有 m 次操作，每次操作给定 l, r, x，表示对编号在 $[l, r]$ 内的所有队列执行 push(x) 操作。每次操作结束后，检查每一个队列，如果第 i 个队列的 size() 大于 a_i，就对这个队列执行 pop()。

[4] [2018 集训队互测 Day 1] 完美的队列

例题[4]

有 n 个 std::queue<int>，编号从 1 到 n。第 i 个序列的容量为 a_i。

初始 n 个序列都是空的。有 m 次操作，每次操作给定 l, r, x，表示对编号在 $[l, r]$ 内的所有队列执行 push(x) 操作。每次操作结束后，检查每一个队列，如果第 i 个队列的 size() 大于 a_i，就对这个队列执行 pop()。

需要在每次操作后，输出目前还在队列内的权值种数。

[4] [2018 集训队互测 Day 1] 完美的队列

例题[4]

有 n 个 std::queue<int>，编号从 1 到 n。第 i 个序列的容量为 a_i。

初始 n 个序列都是空的。有 m 次操作，每次操作给定 l, r, x，表示对编号在 $[l, r]$ 内的所有队列执行 push(x) 操作。每次操作结束后，检查每一个队列，如果第 i 个队列的 size() 大于 a_i，就对这个队列执行 pop()。

需要在每次操作后，输出目前还在队列内的权值种数。

$n, m \leq 10^5$

[4] [2018 集训队互测 Day 1] 完美的队列

性质分析

对于每次操作，如果能计算出这次操作加入的 x 全部被弹出序列的时刻，那么可以简单扫描线求得答案。

性质分析

对于每次操作，如果能计算出这次操作加入的 x 全部被弹出序列的时刻，那么可以简单扫描线求得答案。

对于操作 i，记这次操作加入的 x 全部被弹出的时刻为 f_i。如果对于操作 $i < j$，操作 i 和操作 j 的区间相同，那么有 $f_i \leq f_j$。如果操作中不同的 $[l, r]$ 比较少，可以对每一对 $[l, r]$ 单独 two pointers 处理。

序列分块算法

发现操作可以 $[l, r]$ 分裂成任意 $[l, m]$ 和 $[m+1, r]$ 满足 $1 \leq m < r$。

序列分块算法

中国计算机学会 China Computer Federation

发现操作可以 $[l,r]$ 分裂成任意 $[l,m]$ 和 $[m+1,r]$ 满足 $1 \le m < r$。

取正整数 B，对于原序列每 B 个元素分一块，每次操作 $[l,r]$ 就被自然分裂成了 $O(B)$ 个散点以及 $O(\frac{n}{B})$ 个整块。

序列分块算法

发现操作可以 $[l,r]$ 分裂成任意 $[l,m]$ 和 $[m+1,r]$ 满足 $1 \le m < r$。

取正整数 B，对于原序列每 B 个元素分一块，每次操作 $[l,r]$ 就被自然分裂成了 $O(B)$ 个散点以及 $O(\frac{n}{B})$ 个整块。

对于这个 $O(\frac{n}{B})$ 个整块，我们称这次操作完整覆盖了这一块；对于这 $O(B)$ 个散点，称这次操作不完整覆盖了这个散点。

序列分块算法

对于每个整块，从前往后枚举所有操作并维护当前队列容量减去当前队列大小的最大值并 two pointers。如果操作完全覆盖这个整块，维护整体加的 tag；如果操作不完全覆盖了这块里的某个散点，则重构整块维护新的最大值。不完全覆盖的情况只会出现 $O(q)$ 次，故这部分总复杂度为 $O(\frac{n}{B}q + Bq)$ 。

序列分块算法

对于每个整块，从前往后枚举所有操作并维护当前队列容量减去当前队列大小的最大值并 two pointers。如果操作完全覆盖这个整块，维护整体加的 tag；如果操作不完全覆盖了这块里的某个散点，则重构整块维护新的最大值。不完全覆盖的情况只会出现 $O(q)$ 次，故这部分总复杂度为 $O(\frac{n}{B}q + Bq)$ 。

对于每个散点，记不完全覆盖这个散点的操作序列为 $p_1, p_2, \ldots, p_k$ 。可以预处理 s_i 表示前 i 次操作完整覆盖这个散点所在整块的操作个数，这样可以只在 p 序列上 two pointers，即找到 p_i 被弹出之前最后一次被不完整覆盖的操作 p_j，那么 f_{p_i} 一定是 p_j 和 p_{j+1} 之间某个完整覆盖整块的操作，可以直接求出。这部分总复杂度为 $O(\frac{n}{B}q + Bq)$。

序列分块算法

对于每个整块，从前往后枚举所有操作并维护当前队列容量减去当前队列大小的最大值并 two pointers。如果操作完全覆盖这个整块，维护整体加的 tag；如果操作不完全覆盖了这块里的某个散点，则重构整块维护新的最大值。不完全覆盖的情况只会出现 $O(q)$ 次，故这部分总复杂度为 $O(\frac{n}{B}q + Bq)$ 。

对于每个散点，记不完全覆盖这个散点的操作序列为 $p_1, p_2, \ldots, p_k$ 。可以预处理 s_i 表示前 i 次操作完整覆盖这个散点所在整块的操作个数，这样可以只在 p 序列上 two pointers，即找到 p_i 被弹出之前最后一次被不完整覆盖的操作 p_j，那么 f_{p_i} 一定是 p_j 和 p_{j+1} 之间某个完整覆盖整块的操作，可以直接求出。这部分总复杂度为 $O(\frac{n}{B}q + Bq)$。

取 $B = \sqrt{n}$，最优复杂度为 $O(\sqrt{n}q)$。

思考

碰到预处理/维护复杂度和询问复杂度不平衡时，可以考虑使用序列分块进行平衡。

普通莫队算法

莫队的本质也是一种序列分块算法。

普通莫队算法

莫队的本质也是一种序列分块算法。

对于长度为 n 的序列上的 q 个区间询问问题，取正整数 B，把序列元素 B 个分为一块。

普通莫队算法

莫队的本质也是一种序列分块算法。

对于长度为 n 的序列上的 q 个区间询问问题，取正整数 B，把序列元素 B 个分为一块。

对于询问的区间，按左端点所在块的编号为第一关键字，右端点为第二关键字从小到大排序。按顺序求每个区间的答案，每次直接从上一个区间暴力扩展移动到这个区间，即从 $[l, r]$ 转移到 $[l, r-1]$，$[l, r+1]$，$[l-1, r]$ 或 $[l+1, r]$。上述操作称为一次扩展。

普通莫队算法

莫队的本质也是一种序列分块算法。

对于长度为 n 的序列上的 q 个区间询问问题，取正整数 B，把序列元素 B 个分为一块。

对于询问的区间，按左端点所在块的编号为第一关键字，右端点为第二关键字从小到大排序。按顺序求每个区间的答案，每次直接从上一个区间暴力扩展移动到这个区间，即从 $[l,r]$ 转移到 $[l,r-1]$，$[l,r+1]$，$[l-1,r]$ 或 $[l+1,r]$。上述操作称为一次扩展。

对于左端点在同一块内的相邻询问，右端点递增，一共至多扩展 $O(n)$ 次，左端点每次询问至多扩展 $O(B)$ 步；对于左端点不在同一块内的相邻询问，至多有 $O(\frac{n}{B})$ 个。复杂度为 $O(\frac{n}{B}n+Bq)$ 次扩展的复杂度。取 $B=\frac{n}{\sqrt{q}}$，最优复杂度为 $O(n\sqrt{q})$ 次扩展的复杂度。

普通莫队算法

莫队的本质也是一种序列分块算法。

对于长度为 n 的序列上的 q 个区间询问问题，取正整数 B，把序列元素 B 个分为一块。

对于询问的区间，按左端点所在块的编号为第一关键字，右端点为第二关键字从小到大排序。按顺序求每个区间的答案，每次直接从上一个区间暴力扩展移动到这个区间，即从 $[l,r]$ 转移到 $[l,r-1]$，$[l,r+1]$，$[l-1,r]$ 或 $[l+1,r]$。上述操作称为一次扩展。

对于左端点在同一块内的相邻询问，右端点递增，一共至多扩展 $O(n)$ 次，左端点每次询问至多扩展 $O(B)$ 步；对于左端点不在同一块内的相邻询问，至多有 $O(\frac{n}{B})$ 个。复杂度为 $O(\frac{n}{B}n+Bq)$ 次扩展的复杂度。取 $B=\frac{n}{\sqrt{q}}$，最优复杂度为 $O(n\sqrt{q})$ 次扩展的复杂度。

高维莫队

中国计算机学会 China Computer Federation

对于三维（带修）、四维的查询，同样可以使用类似的排序 + 暴力扩展的算法，需要适当选取 B 的大小。

例题

给定长度为 n 的序列 a_i。

例题

给定长度为 n 的序列 a_i。

q 次询问，每次询问给定区间 $[l, r]$，假设 $a_l, a_{l+1}, ..., a_r$ 排序后为 $b_1, b_2, ..., b_{r-l+1}$，需要求出 $\max_{i=1}^{r-l} b_{i+1} - b_i$。

例题

给定长度为 n 的序列 a_i。

q 次询问，每次询问给定区间 $[l, r]$，假设 $a_l, a_{l+1}, ..., a_r$ 排序后为 $b_1, b_2, ..., b_{r-l+1}$，需要求出 $\max_{i=1}^{r-l} b_{i+1} - b_i$。

$n, q, a_i \leq 10^5$

性质分析

用两个 multiset<int> 分别维护当前区间内的 a_i 以及 $b_{i+1} - b_i$，扩展时可以方便维护。根据上述分析，运用普通莫队可以做到 $O(n\sqrt{q}\log_2 n)$ 的复杂度。

性质分析

用两个 multiset<int> 分别维护当前区间内的 a_i 以及 $b_{i+1} - b_i$，扩展时可以方便维护。根据上述分析，运用普通莫队可以做到 $O(n\sqrt{q}\log_2 n)$ 的复杂度。

使用 multiset<int> 是为了以下目的：

- 插入/删除元素，查询这个元素的前驱和后继
- 插入/删除元素，查询所有元素的最大值

性质分析

用两个 multiset<int> 分别维护当前区间内的 a_i 以及 $b_{i+1}-b_i$，扩展时可以方便维护。根据上述分析，运用普通莫队可以做到 $O(n\sqrt{q}\log_2 n)$ 的复杂度。

使用 multiset<int> 是为了以下目的：

- 插入/删除元素，查询这个元素的前驱和后继
- 插入/删除元素，查询所有元素的最大值

如果扩展操作只有删除元素，即只从 $[l,r]$ 扩展到 $[l+1,r]$ 和 $[l,r-1]$，那么删除元素并查询这个元素的前驱/后继可以使用链表完成，且 $b_{i+1}-b_i$ 的最大值不会变小，所以用一个变量维护即可做到 $O(1)$ 的扩展。

回滚莫队算法

中国计算机学会 China Computer Federation

根据上述分析，我们希望找到一种排序方式使得大部分的扩展操作都是从 $[l,r]$ 扩展到 $[l+1,r]$ 和 $[l,r-1]$ 。

回滚莫队算法

中国计算机学会 China Computer Federation

根据上述分析，我们希望找到一种排序方式使得大部分的扩展操作都是从 $[l,r]$ 扩展到 $[l+1,r]$ 和 $[l,r-1]$。

取正整数 B，对于序列每 B 个元素分一块。对于询问的区间，按左端点所在块的编号为第一关键字升序，右端点为第二关键字降序排序。对于每一块，记这块的左右端点为 $[u,v]$，考虑回答所有左端点属于这块的询问。

回滚莫队算法

中国计算机学会 China Computer Federation

根据上述分析，我们希望找到一种排序方式使得大部分的扩展操作都是从 $[l,r]$ 扩展到 $[l+1,r]$ 和 $[l,r-1]$。

取正整数 B，对于序列每 B 个元素分一块。对于询问的区间，按左端点所在块的编号为第一关键字升序，右端点为第二关键字降序排序。对于每一块，记这块的左右端点为 $[u,v]$，考虑回答所有左端点属于这块的询问。

先对 $a_u, a_{u+1}, \ldots, a_n$ 桶排序，建立初始的链表，以及当前 $b_{i+1}-b_i$ 的最大值。也就是说，现在维护的是区间的 $[u,n]$ 的信息。按顺序回答询问，假设当前询问区间为 $[l,r]$，那么通过不断减小区间的右端点，可以在单次扩展 $O(1)$ 的复杂度下维护出区间 $[u,r]$ 的信息。

回滚莫队算法

中国计算机学会 China Computer Federation

根据上述分析，我们希望找到一种排序方式使得大部分的扩展操作都是从 $[l,r]$ 扩展到 $[l+1,r]$ 和 $[l,r-1]$。

取正整数 B，对于序列每 B 个元素分一块。对于询问的区间，按左端点所在块的编号为第一关键字升序，右端点为第二关键字降序排序。对于每一块，记这块的左右端点为 $[u,v]$，考虑回答所有左端点属于这块的询问。

先对 $a_u, a_{u+1}, \ldots, a_n$ 桶排序，建立初始的链表，以及当前 $b_{i+1}-b_i$ 的最大值。也就是说，现在维护的是区间的 $[u,n]$ 的信息。按顺序回答询问，假设当前询问区间为 $[l,r]$，那么通过不断减小区间的右端点，可以在单次扩展 $O(1)$ 的复杂度下维护出区间 $[u,r]$ 的信息。

接下来不断增大当前区间的左端点，就可以得到 $[l,r]$ 的信息。为了保证大部分扩展操作都是删除操作，需要进行 $O(B)$ 次撤销，回滚使得当前维护的是 $[u,r]$ 的信息。

回滚莫队算法

根据上述分析，我们希望找到一种排序方式使得大部分的扩展操作都是从 $[l,r]$ 扩展到 $[l+1,r]$ 和 $[l,r-1]$。

取正整数 B，对于序列每 B 个元素分一块。对于询问的区间，按左端点所在块的编号为第一关键字升序，右端点为第二关键字降序排序。对于每一块，记这块的左右端点为 $[u,v]$，考虑回答所有左端点属于这块的询问。

先对 $a_u, a_{u+1}, \ldots, a_n$ 桶排序，建立初始的链表，以及当前 $b_{i+1}-b_i$ 的最大值。也就是说，现在维护的是区间的 $[u,n]$ 的信息。按顺序回答询问，假设当前询问区间为 $[l,r]$，那么通过不断减小区间的右端点，可以在单次扩展 $O(1)$ 的复杂度下维护出区间 $[u,r]$ 的信息。

接下来不断增大当前区间的左端点，就可以得到 $[l,r]$ 的信息。为了保证大部分扩展操作都是删除操作，需要进行 $O(B)$ 次撤销，回滚使得当前维护的是 $[u,r]$ 的信息。

这样就把需要插入/删除变成了需要删除/撤销，复杂度降为 $O(n\sqrt{q})$。

思考

如果维护的区间信息插入容易而删除难（比如图的连通性），同样可以使用回滚莫队，按右端点第二关键字升序排序即可。

例题

给定长度为 n 的排列 p_i。q 次询问，每次给定区间 $[l, r]$，需要计算 $l \leq i < j \leq r, p_i > p_j$ 的 (i, j) 对数，即区间逆序对数。

例题

给定长度为 n 的排列 p_i。q 次询问，每次给定区间 $[l, r]$，需要计算 $l \leq i < j \leq r, p_i > p_j$ 的 (i, j) 对数，即区间逆序对数。

$n, q \leq 10^5$

二次离线算法

中国计算机学会 China Computer Federation

考虑到是区间询问信息问题，先按普通莫队的方式排序。

二次离线算法

中国计算机学会 China Computer Federation

考虑到是区间询问信息问题，先按普通莫队的方式排序。

为了表述方便，记 $f(l,r)=\sum_{l\le i<j\le r}[p_i>p_j]$ 即区间逆序对数；记 $g(l_1,r_1,l_2,r_2)=\sum_{l_1\le i\le r_1,l_2\le j\le r_2}[p_i>p_j]$，即区间 $[l_1,r_1]$ 到区间 $[l_2,r_2]$ 的贡献。

二次离线算法

考虑到是区间询问信息问题，先按普通莫队的方式排序。

为了表述方便，记 $f(l,r)=\sum_{l\le i<j\le r}[p_i>p_j]$ 即区间逆序对数；记 $g(l_1,r_1,l_2,r_2)=\sum_{l_1\le i\le r_1,l_2\le j\le r_2}[p_i>p_j]$，即区间 $[l_1,r_1]$ 到区间 $[l_2,r_2]$ 的贡献。

考虑排序后，上一个询问是 $[l,r]$，下一个询问是 $[l',r']$，考虑不通过一步一步扩展，而是直接计算 $[l,r]$ 转移到 $[l,r']$，然后 $[l,r']$ 再转移到 $[l',r']$ 的贡献。

二次离线算法

考虑到是区间询问信息问题，先按普通莫队的方式排序。

为了表述方便，记 $f(l,r)=\sum_{l\le i<j\le r}[p_i>p_j]$ 即区间逆序对数；记 $g(l_1,r_1,l_2,r_2)=\sum_{l_1\le i\le r_1,l_2\le j\le r_2}[p_i>p_j]$，即区间 $[l_1,r_1]$ 到区间 $[l_2,r_2]$ 的贡献。

考虑排序后，上一个询问是 $[l,r]$，下一个询问是 $[l',r']$，考虑不通过一步一步扩展，而是直接计算 $[l,r]$ 转移到 $[l,r']$，然后 $[l,r']$ 再转移到 $[l',r']$ 的贡献。

假设当前要从 $[l,r]$ 转移到 $[l,r'](r'>r)$，发现有 $f(l,r')-f(l,r)=f(1,r')-f(1,l-1)-g(1,l-1,r+1,r')$。

二次离线算法

考虑到是区间询问信息问题，先按普通莫队的方式排序。

为了表述方便，记 $f(l,r)=\sum_{l\le i<j\le r}[p_i>p_j]$ 即区间逆序对数；记 $g(l_1,r_1,l_2,r_2)=\sum_{l_1\le i\le r_1,l_2\le j\le r_2}[p_i>p_j]$，即区间 $[l_1,r_1]$ 到区间 $[l_2,r_2]$ 的贡献。

考虑排序后，上一个询问是 $[l,r]$，下一个询问是 $[l',r']$，考虑不通过一步一步扩展，而是直接计算 $[l,r]$ 转移到 $[l,r']$，然后 $[l,r']$ 再转移到 $[l',r']$ 的贡献。

假设当前要从 $[l,r]$ 转移到 $[l,r'](r'>r)$，发现有 $f(l,r')-f(l,r)=f(1,r')-f(1,l-1)-g(1,l-1,r+1,r')$。

观察上述式子，发现关心的 f 的第一维都为 1，可以用树状数组直接计算所有的 $f(1,i)$。所有 g 的第一维也都是 1，可以把查询 $(r+1,r')$ 挂在位置 $l-1$ 表示需要计算 $g(1,l-1,r+1,r')$。

二次离线算法

考虑到是区间询问信息问题，先按普通莫队的方式排序。

为了表述方便，记 $f(l,r)=\sum_{l\le i<j\le r}[p_i>p_j]$ 即区间逆序对数；记 $g(l_1,r_1,l_2,r_2)=\sum_{l_1\le i\le r_1,l_2\le j\le r_2}[p_i>p_j]$，即区间 $[l_1,r_1]$ 到区间 $[l_2,r_2]$ 的贡献。

考虑排序后，上一个询问是 $[l,r]$，下一个询问是 $[l',r']$，考虑不通过一步一步扩展，而是直接计算 $[l,r]$ 转移到 $[l,r']$，然后 $[l,r']$ 再转移到 $[l',r']$ 的贡献。

假设当前要从 $[l,r]$ 转移到 $[l,r'](r'>r)$，发现有 $f(l,r')-f(l,r)=f(1,r')-f(1,l-1)-g(1,l-1,r+1,r')$。

观察上述式子，发现关心的 f 的第一维都为 1，可以用树状数组直接计算所有的 $f(1,i)$。所有 g 的第一维也都是 1，可以把查询 $(r+1,r')$ 挂在位置 $l-1$ 表示需要计算 $g(1,l-1,r+1,r')$。

注意根据普通莫队的复杂度分析，有 $\sum r'-r\le n\sqrt{q}$。

二次离线算法

考虑枚举 i 然后回答所有的询问 $g(1, i, l, r)$。

二次离线算法

中国计算机学会 China Computer Federation

考虑枚举 i 然后回答所有的询问 $g(1,i,l,r)$。

如果用树状数组维护，复杂度为 $O((n+n\sqrt{q})\log_2 n)$，发现出现了维护信息和查询的复杂度不平衡，考虑用序列分块进行平衡。

二次离线算法

考虑枚举 i 然后回答所有的询问 $g(1,i,l,r)$。

如果用树状数组维护，复杂度为 $O((n+n\sqrt{q})\log_2 n)$，发现出现了维护信息和查询的复杂度不平衡，考虑用序列分块进行平衡。

每次加入 p_i，相当于区间加，即对于 $j<p_i$ 的位置 $+1$。取正整数 B 把序列 B 个元素分一块。对于 $j<p_i$ 的整块打整体加的 tag，对于散点暴力 $+1$。这样复杂度就变成了 $O(n(\frac{n}{B}+B)+n\sqrt{q})$，当 $B=\sqrt{n}$ 时取到最优值 $O(n(\sqrt{n}+\sqrt{q}))$。

思想介绍

中国计算机学会 China Computer Federation

当维护对象的总规模有限制时，可以考虑将它们分成两类：规模较小的和规模较大的（通常将阈值设为根号），那么后者的个数不会太多。再对于两类对象分别使用不同的算法维护，经过分析能得到较优的复杂度。

例题[5]

给定 n 个点 m 条边的无向图 $G=(V,E)$，每个点 u 都有一个权值 a_u。定义 $s_u=\{a_v|(u,v)\in E\}$。

[5][2020 Multi-University Training Contest 1]Finding a MEX

例题[5]

给定 n 个点 m 条边的无向图 $G=(V,E)$，每个点 u 都有一个权值 a_u。定义 $s_u=\{a_v|(u,v)\in E\}$。

需要支持以下操作：

- 修改某个 a_u。
- 对于某个 u，询问 s_u 中未出现的最小的数。

[5] [2020 Multi-University Training Contest 1]Finding a MEX

例题[5]

给定 n 个点 m 条边的无向图 $G=(V,E)$，每个点 u 都有一个权值 a_u。定义 $s_u=\{a_v|(u,v)\in E\}$。

需要支持以下操作：

- 修改某个 a_u。
- 对于某个 u，询问 s_u 中未出现的最小的数。

$n,m,q\leq 10^5$

[5] [2020 Multi-University Training Contest 1]Finding a MEX

按点的度数分治算法

取正整数 B，把度数大于 B 的称为大点，否则为小点。容易发现大点个数是 $O(\frac{m}{B})$ 的。

按点的度数分治算法

取正整数 B，把度数大于 B 的称为大点，否则为小点。容易发现大点个数是 $O(\frac{m}{B})$ 的。

每次修改时只维护相邻的大点的信息。询问时如果是小点就枚举所有邻居暴力，否则直接在大点上维护的数据结构上查询。

按点的度数分治算法

取正整数 B，把度数大于 B 的称为大点，否则为小点。容易发现大点个数是 $O(\frac{m}{B})$ 的。

每次修改时只维护相邻的大点的信息。询问时如果是小点就枚举所有邻居暴力，否则直接在大点上维护的数据结构上查询。

发现对数据结构的修改次数是 $O(\frac{m}{B}q)$ 的，查询次数是 $O(q)$ 的，故可以用序列分块平衡复杂度。

按点的度数分治算法

取正整数 B，把度数大于 B 的称为大点，否则为小点。容易发现大点个数是 $O(\frac{m}{B})$ 的。

每次修改时只维护相邻的大点的信息。询问时如果是小点就枚举所有邻居暴力，否则直接在大点上维护的数据结构上查询。

发现对数据结构的修改次数是 $O(\frac{m}{B}q)$ 的，查询次数是 $O(q)$ 的，故可以用序列分块平衡复杂度。

最终复杂度为 $O(q(\sqrt{n}+\sqrt{m}))$。

例题

中国计算机学会 China Computer Federation

给定 n 个字符串 s_i。q 次询问，每次给出 i, j，询问 s_i 和 s_j 的最长公共字串。

例题

给定 n 个字符串 s_i。q 次询问，每次给出 i, j，询问 s_i 和 s_j 的最长公共字串。

$n, \sum_{i=1}^{n} |s_i| \leq 10^5$

按字符串长度分治算法

记 $t = \sum_{i=1}^{n} |s_i|$。

按字符串长度分治算法

记 $t = \sum_{i=1}^{n} |s_i|$。

取正整数 B，对于 $|s_i| > B$ 的 i，建立后缀自动机。容易发现这样的 i 的个数是 $O(\frac{t}{B})$ 的。枚举 j 将 s_j 在后缀自动机上转移可以求得所有 (i, j) 的答案。

按字符串长度分治算法

记 $t = \sum_{i=1}^{n} |s_i|$。

取正整数 B，对于 $|s_i| > B$ 的 i，建立后缀自动机。容易发现这样的 i 的个数是 $O(\frac{t}{B})$ 的。枚举 j 将 s_j 在后缀自动机上转移可以求得所有 (i, j) 的答案。

每次询问时，如果 $\max(|s_i|, |s_j|) > B$，那么直接回答预处理的答案；否则对于 s_i 建立后缀自动机，将 s_j 在后缀自动机上转移。

按字符串长度分治算法

记 $t = \sum_{i=1}^{n} |s_i|$。

取正整数 B，对于 $|s_i| > B$ 的 i，建立后缀自动机。容易发现这样的 i 的个数是 $O(\frac{t}{B})$ 的。枚举 j 将 s_j 在后缀自动机上转移可以求得所有 (i, j) 的答案。

每次询问时，如果 $\max(|s_i|, |s_j|) > B$，那么直接回答预处理的答案；否则对于 s_i 建立后缀自动机，将 s_j 在后缀自动机上转移。

最终复杂度为 $O(t\sqrt{q})$。

例题[6]

给定长度为 n 的序列 a_i。定义 x 为区间 $[l,r]$ 的众数当且仅当不存在 y 使得 y 在区间 $[l,r]$ 中的出现次数大于 x 在区间 $[l,r]$ 中的出现次数。

[6] [THUPC 2021 初赛] 区间众数

例题[6]

中国计算机学会 China Computer Federation

给定长度为 n 的序列 a_i。定义 x 为区间 $[l,r]$ 的众数当且仅当不存在 y 使得 y 在区间 $[l,r]$ 中的出现次数大于 x 在区间 $[l,r]$ 中的出现次数。

有 q 次询问，每次询问给出 l,r，求有多少二元组 (l',r') 满足 $l \le l' \le r' \le r$，且 $[l',r']$ 的区间长度为奇数，且 $(l'+r')/2$ 是区间 $[l',r']$ 中的众数。

[6][THUPC 2021 初赛] 区间众数

例题[6]

给定长度为 n 的序列 a_i。定义 x 为区间 $[l,r]$ 的众数当且仅当不存在 y 使得 y 在区间 $[l,r]$ 中的出现次数大于 x 在区间 $[l,r]$ 中的出现次数。

有 q 次询问，每次询问给出 l,r，求有多少二元组 (l',r') 满足 $l \le l' \le r' \le r$，且 $[l',r']$ 的区间长度为奇数，且 $(l'+r')/2$ 是区间 $[l',r']$ 中的众数。

$n \le 5 \times 10^5, q \le 10^6$

[6] [THUPC 2021 初赛] 区间众数

按数字出现次数分治算法

枚举区间中点 x。假设 x 在序列中一共出现了 c 次，对于 $1 \le i \le c$，记 l_i 为最小的数使得 $[x-l_i, x+l_i]$ 包含至少 i 个 x。那么一定存在一个 $r_i < l_{i+1}$ 使得对于任意 $l_i \le j \le r_i$ 都有 $[x-j, x+j]$ 合法且对于任意 $r_i < j < l_{i+1}$ 都有 $[x-j, x+j]$ 不合法。只要求得 l_i 和 r_i 就可以用扫描线和树状数组求得答案。

按数字出现次数分块算法

取正整数 B。

按数字出现次数分块算法

取正整数 B。

对于区间中点 x 以及出现次数 c，如果 $c > B$，可以直接枚举 i 维护 $[x-i, x+i]$ 的众数；如果 $c \leq B$，可以对于每个位置 i 以及 $1 \leq j \leq c$ 处理出 $f(i,j)$ 表示从点 i 开始向右延伸，要求当前区间里每个数的出现次数都 $\leq j$，最远能延伸的位置。有了 f 之后在 l_i 和 l_{i+1} 之间二分即可得到 r_i。

按数字出现次数分块算法

取正整数 B。

对于区间中点 x 以及出现次数 c，如果 $c > B$，可以直接枚举 i 维护 $[x-i, x+i]$ 的众数；如果 $c \leq B$，可以对于每个位置 i 以及 $1 \leq j \leq c$ 处理出 $f(i,j)$ 表示从点 i 开始向右延伸，要求当前区间里每个数的出现次数都 $\leq j$，最远能延伸的位置。有了 f 之后在 l_i 和 l_{i+1} 之间二分即可得到 r_i 。

最终复杂度为 $O(n\sqrt{n} + (n+q)\log_2 n)$。

例题[7]

给定一棵 n 个节点的树，每个点有一个字符 s_i。记 $str_{u,v}$ 为从 u 走到 v 的简单路径上经过的点上的字符顺次连接而成的字符串。

[7] [CTSC2010] 珠宝商

例题[7]

给定一棵 n 个节点的树，每个点有一个字符 s_i。记 $str_{u,v}$ 为从 u 走到 v 的简单路径上经过的点上的字符顺次连接而成的字符串。

给定长度为 m 的字符串 t，令 $f(s)$ 为字符串 s 在 t 中出现的次数。需要求出 $\sum_{1\leq u,v\leq n} f(str_{u,v})$。

[7] [CTSC2010] 珠宝商

例题[7]

给定一棵 n 个节点的树，每个点有一个字符 s_i。记 $str_{u,v}$ 为从 u 走到 v 的简单路径上经过的点上的字符顺次连接而成的字符串。

给定长度为 m 的字符串 t，令 $f(s)$ 为字符串 s 在 t 中出现的次数。需要求出 $\sum_{1\le u,v\le n} f(str_{u,v})$。

$n, m \le 5 \times 10^4$

[7] [CTSC2010] 珠宝商

性质分析

首先匹配类问题，可以先建出 t 的后缀自动机。

性质分析

首先匹配类问题，可以先建出 t 的后缀自动机。

一个显然的暴力是枚举 u 然后 dfs，对于每个 v 记录当前字符串在后缀自动机上的节点位置，将这个节点的 occurance 计入答案。

性质分析

首先匹配类问题，可以先建出 t 的后缀自动机。

一个显然的暴力是枚举 u 然后 dfs，对于每个 v 记录当前字符串在后缀自动机上的节点位置，将这个节点的 occurance 计入答案。

这个做法的复杂度是 $O(n^2)$ 。

性质分析

另一个不那么暴力的做法是考虑对于树上路径点分治。

性质分析

另一个不那么暴力的做法是考虑对于树上路径点分治。

假设当前重心是 rt，那么考虑 $u \to rt \to v$ 这条路径。从 rt 开始 dfs，沿着反串的后缀树，即正串的后缀自动机的 parent tree 走，树上走到 u 时给走到的当前后缀自动机节点打上 $+1$ 的标记，表示这个节点的 Right 集合向前可以匹配 $u \to rt$ 这条路径。

性质分析

另一个不那么暴力的做法是考虑对于树上路径点分治。

假设当前重心是 rt，那么考虑 $u \to rt \to v$ 这条路径。从 rt 开始 dfs，沿着反串的后缀树，即正串的后缀自动机的 parent tree 走，树上走到 u 时给走到的当前后缀自动机节点打上 $+1$ 的标记，表示这个节点的 Right 集合向前可以匹配 $u \to rt$ 这条路径。

同样处理在反串的后缀自动机上走，处理 $rt \to v$ 这条路径。

性质分析

另一个不那么暴力的做法是考虑对于树上路径点分治。

假设当前重心是 rt，那么考虑 $u \to rt \to v$ 这条路径。从 rt 开始 dfs，沿着反串的后缀树，即正串的后缀自动机的 parent tree 走，树上走到 u 时给走到的当前后缀自动机节点打上 $+1$ 的标记，表示这个节点的 Right 集合向前可以匹配 $u \to rt$ 这条路径。

同样处理在反串的后缀自动机上走，处理 $rt \to v$ 这条路径。

这样，在两个后缀自动机上把标记下放后就知道了对于每个位置 i，i 向前能匹配 $u \to rt$ 以及 i 向后能匹配 $rt \to v$ 的 u 和 v 的个数，直接相乘就能得到答案。对于 u 和 v 在同一个子树的情况，减去这个子树单独做的答案即可。

性质分析

另一个不那么暴力的做法是考虑对于树上路径点分治。

假设当前重心是 rt，那么考虑 $u \to rt \to v$ 这条路径。从 rt 开始 dfs，沿着反串的后缀树，即正串的后缀自动机的 parent tree 走，树上走到 u 时给走到的当前后缀自动机节点打上 $+1$ 的标记，表示这个节点的 Right 集合向前可以匹配 $u \to rt$ 这条路径。

同样处理在反串的后缀自动机上走，处理 $rt \to v$ 这条路径。

这样，在两个后缀自动机上把标记下放后就知道了对于每个位置 i，i 向前能匹配 $u \to rt$ 以及 i 向后能匹配 $rt \to v$ 的 u 和 v 的个数，直接相乘就能得到答案。对于 u 和 v 在同一个子树的情况，减去这个子树单独做的答案即可。

点分治时 dfs 的复杂度是 $O(n \log_2 n)$ 的，每次都需要遍历整个后缀自动机，那么这个做法的复杂度是 $O(n \log_2 n + nm)$。

分治结合分块算法

中国计算机学会 China Computer Federation

上述两个算法都不能通过所有数据，但是可以考虑根号分治的思想来平衡复杂度。

分治结合分块算法

中国计算机学会 China Computer Federation

上述两个算法都不能通过所有数据，但是可以考虑根号分治的思想来平衡复杂度。

取正整数 B，如果当前点分连通块大小 $\leq B$ 就采用 $O(n^2)$ 的暴力直接返回，否则采用 $O(n\log_2 n + nm)$ 的点分继续递归。

分治结合分块算法

一方面，假设采用暴力的连通块大小分别为 $x_1, x_2, ..., x_k$，那么有 $\sum_{i=1}^{k} x_i \leq n$，那么 $\sum_{i=1}^{k} x_i^2 \leq (\sum_{i=1}^{k} x_i) * B \leq Bn$ 。

分治结合分块算法

一方面，假设采用暴力的连通块大小分别为 $x_1, x_2, ..., x_k$，那么有 $\sum_{i=1}^{k} x_i \leq n$，那么 $\sum_{i=1}^{k} x_i^2 \leq (\sum_{i=1}^{k} x_i) * B \leq Bn$ 。

另一方面，画出点分治树，记点 x 的子树大小为 sz_x。删去所有 $sz_x \leq B$ 的 x。此时所有叶子的 $sz > B$ 且叶子两两没有祖先后代关系，那么至多有 $\frac{n}{B}$ 个叶子。

分治结合分块算法

一方面，假设采用暴力的连通块大小分别为 $x_1, x_2, ..., x_k$，那么有 $\sum_{i=1}^{k} x_i \leq n$，那么 $\sum_{i=1}^{k} x_i^2 \leq (\sum_{i=1}^{k} x_i) * B \leq Bn$ 。

另一方面，画出点分治树，记点 x 的子树大小为 sz_x。删去所有 $sz_x \leq B$ 的 x。此时所有叶子的 $sz > B$ 且叶子两两没有祖先后代关系，那么至多有 $\frac{n}{B}$ 个叶子。

对于非叶节点 x，如果只有一个儿子 y，由于点分树大小的性质，那么有 $2B < 2sz_y \leq sz_x$，那么 $sz_x - sz_y > B$。即是说，至多有 $\frac{n}{B}$ 个非叶节点只有一个儿子。那么可得删去 $sz_x \leq B$ 的 x 后这棵树的点数是 $O(\frac{n}{B})$ 的。

分治结合分块算法

一方面，假设采用暴力的连通块大小分别为 $x_1, x_2, ..., x_k$，那么有 $\sum_{i=1}^{k} x_i \leq n$，那么 $\sum_{i=1}^{k} x_i^2 \leq (\sum_{i=1}^{k} x_i) * B \leq Bn$ 。

另一方面，画出点分治树，记点 x 的子树大小为 sz_x。删去所有 $sz_x \leq B$ 的 x。此时所有叶子的 $sz > B$ 且叶子两两没有祖先后代关系，那么至多有 $\frac{n}{B}$ 个叶子。

对于非叶节点 x，如果只有一个儿子 y，由于点分树大小的性质，那么有 $2B < 2sz_y \leq sz_x$，那么 $sz_x - sz_y > B$。即是说，至多有 $\frac{n}{B}$ 个非叶节点只有一个儿子。那么可得删去 $sz_x \leq B$ 的 x 后这棵树的点数是 $O(\frac{n}{B})$ 的。

可得总复杂度为 $O(n\log_2 n + Bn + \frac{n}{B}(n+m))$，取 $B = \sqrt{n+m}$ ，认为 n 和 m 同阶的话得到最优复杂度为 $O(n\sqrt{n+m})$。

例题[8]

平面上有 n 个点，第 i 个点在 (i, p_i)。保证 p_i 为一个 1 到 n 的排列。

[8] [NOI2020] 时代的眼泪

例题[8]

平面上有 n 个点，第 i 个点在 (i, p_i)。保证 p_i 为一个 1 到 n 的排列。

有 q 次询问，每次询问给出 r_1, r_2, c_1, c_2，需要计算 $r_1 \le i < j \le r_2, c_1 \le p_i < p_j \le c_2$ 的 (i, j) 对数，即是求给定矩形中顺序对的对数。

[8] [NOI2020] 时代的眼泪

例题[8]

平面上有 n 个点，第 i 个点在 (i, p_i)。保证 p_i 为一个 1 到 n 的排列。

有 q 次询问，每次询问给出 r_1, r_2, c_1, c_2，需要计算 $r_1 \le i < j \le r_2, c_1 \le p_i < p_j \le c_2$ 的 (i, j) 对数，即是求给定矩形中顺序对的对数。

$n \le 10^5, m \le 2 \times 10^5$

[8] [NOI2020] 时代的眼泪

分治结合分块算法

中国计算机学会 China Computer Federation

考虑对于平面建立一棵树套树。

分治结合分块算法

中国计算机学会 China Computer Federation

考虑对于平面建立一棵树套树。

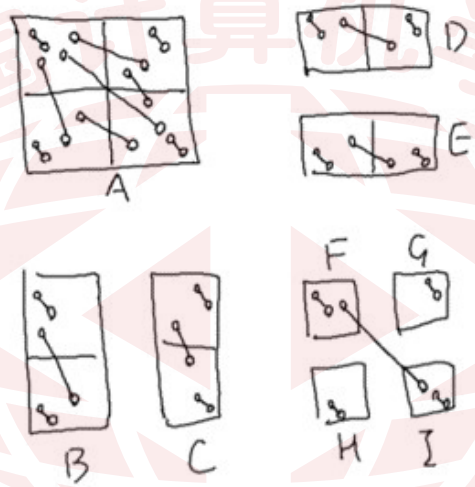
虽然总的节点数是 $O(n^2)$ 的，但是如果对应的某个节点对应的矩形里一个点都没有那么这个节点可以删去。平面上一个点至多出现在树套树上的 $O(\log_2^2 n)$ 个矩形中，故有用的节点只有 $O(n\log_2^2 n)$ 个。

分治结合分块算法

首先考虑怎么维护树套树上的矩形的答案。

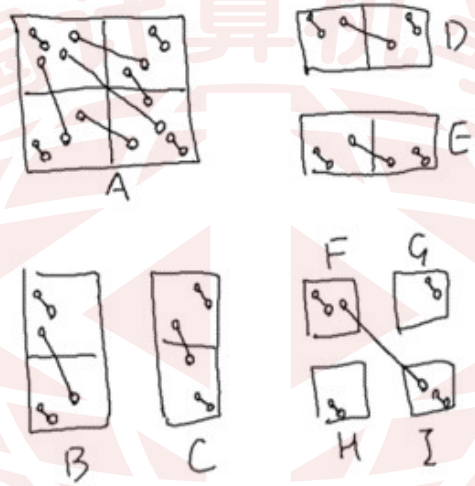
分治结合分块算法

首先考虑怎么维护树套树上的矩形的答案。



分治结合分块算法

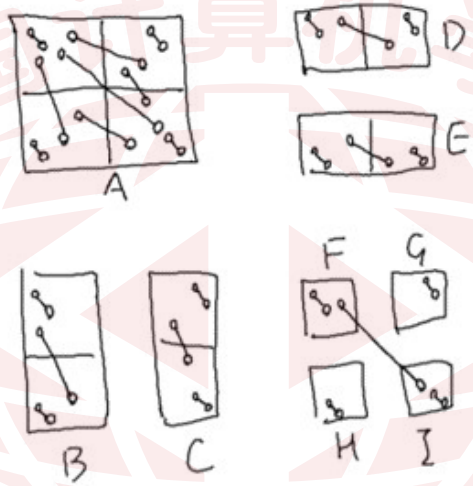
首先考虑怎么维护树套树上的矩形的答案。



显然图上 $A, B, C, D, E, F, G, H, I$ 均为树套树上的节点。

分治结合分块算法

首先考虑怎么维护树套树上的矩形的答案。



显然图上 $A, B, C, D, E, F, G, H, I$ 均为树套树上的节点。

记 $ans(A)$ 为矩形 A 的答案即顺序对数，$size(A)$ 为矩形 A 中的点数。容斥可以得到 $size(A) = size(B) + size(C)$，

$ans(A) = size(F) \times size(I) + ans(B) + ans(C) + ans(D) + ans(E) - ans(F) - ans(G) - ans(H) - ans(I)$ 。

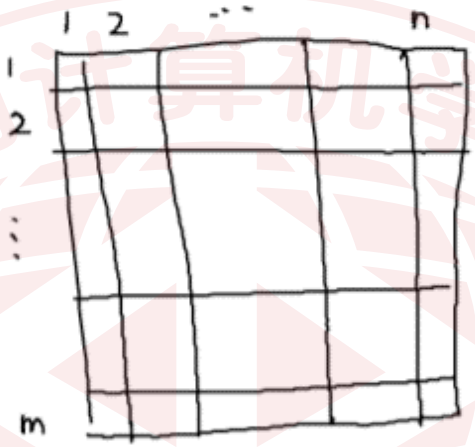
分治结合分块算法

中国计算机学会 China Computer Federation

接下来考虑怎么回答询问。

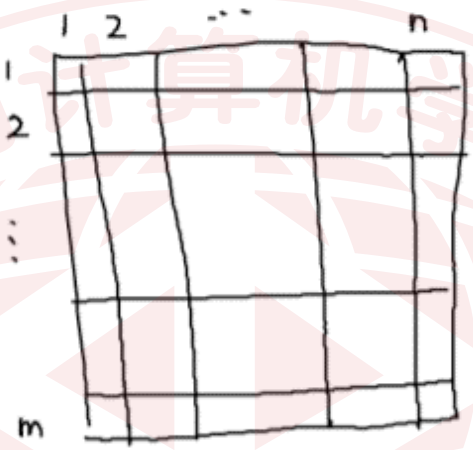
分治结合分块算法

接下来考虑怎么回答询问。



分治结合分块算法

接下来考虑怎么回答询问。



如图，把询问矩形划分成 $m \times n$ 个树套树节点。根据类似的容斥，这次询问的贡献可以分为四部分：

- $\sum_{i=1}^{m} \sum_{j=1}^{n} size(i,j) \sum_{i'=1}^{i-1} \sum_{j'=1}^{j-1} size(i',j')$
- $\sum_{i=1}^{m} ans(i, 1..n)$
- $\sum_{j=1}^{n} ans(1..m, j)$
- $-\sum_{i=1}^{m} \sum_{j=1}^{n} ans(i,j)$

分治结合分块算法

上述贡献中比较困难的 $ans(i, 1..n)$ 和 $ans(1..m, j)$ 都是某一棵线段树节点上的区间顺序对数，可以用二次离线莫队处理。

分治结合分块算法

中国计算机学会 China Computer Federation

上述贡献中比较困难的 $ans(i, 1..n)$ 和 $ans(1..m, j)$ 都是某一棵线段树节点上的区间顺序对数，可以用二次离线莫队处理。

分析一下复杂度。认为 n 和 m 同阶，线段树第 i 层有 $O(q)$ 次询问，复杂度为 $O(n(\sqrt{\frac{n}{2^i}}+\sqrt{\frac{q}{2^i}}))$。把上述式子对于 i 求和并加上树套树的复杂度得到总复杂度为 $O(n(\sqrt{n}+\sqrt{q})+(n+q)\log_2^2 n)$ 。

例题[9]

这是一道交互题。

给出了抽象数据类型 D 和 O，以及运算：

- $+ : D \times D \to D$，满足结合律、交换律，有单位元
- $\cdot : O \times O \to O$，满足结合律，有单位元
- $\cdot : O \times D \to D$，满足结合律，左分配律，左乘 O 中的单位元不变

可以认为 D 和 O 分别是列向量和方阵。

[9] [CTS2021] 半平面修改查询

例题[9]

这是一道交互题。

给出了抽象数据类型 D 和 O，以及运算：

- $+ : D \times D \to D$，满足结合律、交换律，有单位元
- $\cdot : O \times O \to O$，满足结合律，有单位元
- $\cdot : O \times D \to D$，满足结合律，左分配律，左乘 O 中的单位元不变

可以认为 D 和 O 分别是列向量和方阵。

给定二维平面上的 n 个点，每个点有权值 $d_i \in D$。有 q 次询问，每次给定一个半平面和参数 $o_i \in O$，求半平面内的点的 d_i 之和，并将 d_i 修改为 $o_i \cdot d_i$。

[9] [CTS2021] 半平面修改查询

例题[9]

这是一道交互题。

给出了抽象数据类型 D 和 O，以及运算：

- $+ : D \times D \to D$，满足结合律、交换律，有单位元
- $\cdot : O \times O \to O$，满足结合律，有单位元
- $\cdot : O \times D \to D$，满足结合律，左分配律，左乘 O 中的单位元不变

可以认为 D 和 O 分别是列向量和方阵。

给定二维平面上的 n 个点，每个点有权值 $d_i \in D$。有 q 次询问，每次给定一个半平面和参数 $o_i \in O$，求半平面内的点的 d_i 之和，并将 d_i 修改为 $o_i \cdot d_i$。

运算由交互库给出，每次运算会在两个运算对象都不是单位元时花费 1 的代价。要求代价之和不超过 2.5×10^7 。

[9] [CTS2021] 半平面修改查询

例题[9]

这是一道交互题。

给出了抽象数据类型 D 和 O，以及运算：

- $+ : D \times D \to D$，满足结合律、交换律，有单位元
- $\cdot : O \times O \to O$，满足结合律，有单位元
- $\cdot : O \times D \to D$，满足结合律，左分配律，左乘 O 中的单位元不变

可以认为 D 和 O 分别是列向量和方阵。

给定二维平面上的 n 个点，每个点有权值 $d_i \in D$。有 q 次询问，每次给定一个半平面和参数 $o_i \in O$，求半平面内的点的 d_i 之和，并将 d_i 修改为 $o_i \cdot d_i$。

运算由交互库给出，每次运算会在两个运算对象都不是单位元时花费 1 的代价。要求代价之和不超过 2.5×10^7 。

$n \le 2 \times 10^5, q \le 10^4$

[9] [CTS2021] 半平面修改查询

分治结合分块算法

中国计算机学会 China Computer Federation

取正整数 B, 每 B 个询问分一块。

分治结合分块算法

取正整数 B, 每 B 个询问分一块。

假设 solve(l,r) 表示处理第 l 到第 r 个询问的函数。那么这 $r-l+1$ 个半平面把平面划分成了 $(r-l+1)^2$ 个区域，每个区域内的点可以合并处理。

分治结合分块算法

取正整数 B, 每 B 个询问分一块。

假设 solve(l,r) 表示处理第 l 到第 r 个询问的函数。那么这 $r-l+1$ 个半平面把平面划分成了 $(r-l+1)^2$ 个区域，每个区域内的点可以合并处理。

如果 $l=r$，直接操作并返回。否则令 $m=\lfloor\frac{l+r}{2}\rfloor$，将当前的 $(r-l+1)^2$ 个区间忽视掉第 $m+1$ 到 r 的操作进行合并，那么现在的区域实际上是操作 l 到 m 划分出的 $(m-l+1)^2$ 个区域，然后递归到 solve(l,m)。接下来将这 $(m-l+1)^2$ 个区域拆开变回 $O(r-l+1)^2$ 个，再忽视第 l 到 m 个操作进行合并，现在的区域实际上是操作 $m+1$ 到 r 划分出的 $(r-m)^2$ 个区域，再递归到 solve(m+1,r)。

分治结合分块算法

取正整数 B, 每 B 个询问分一块。

假设 solve(l,r) 表示处理第 l 到第 r 个询问的函数。那么这 $r-l+1$ 个半平面把平面划分成了 $(r-l+1)^2$ 个区域，每个区域内的点可以合并处理。

如果 $l=r$，直接操作并返回。否则令 $m=\lfloor\frac{l+r}{2}\rfloor$，将当前的 $(r-l+1)^2$ 个区间忽视掉第 $m+1$ 到 r 的操作进行合并，那么现在的区域实际上是操作 l 到 m 划分出的 $(m-l+1)^2$ 个区域，然后递归到 solve(l,m)。接下来将这 $(m-l+1)^2$ 个区域拆开变回 $O(r-l+1)^2$ 个，再忽视第 l 到 m 个操作进行合并，现在的区域实际上是操作 $m+1$ 到 r 划分出的 $(r-m)^2$ 个区域，再递归到 solve(m+1,r)。

分析操作一个整块的复杂度，记 $T(r-l+1)$ 为 solve(l,r) 的操作次数的话，有 $T(n)=n^2+2\,T(n/2)$, 分析得到 $T(n)=O(n^2)$，故操作次数的复杂度为 $O(\frac{q}{B}(n+B^2))$，当 $B=\sqrt{n}$ 时操作次数复杂度最优为 $O(q\sqrt{n})$。

思考

中国计算机学会 China Computer Federation

如果可以用采用分治的思想解决但直接分治复杂度不能接受或比较难处理，可以考虑分治结合分块。

思考

如果可以用采用分治的思想解决但直接分治复杂度不能接受或比较难处理，可以考虑分治结合分块。

注意到很多时候最终分析出的复杂度并不带 $\log_2 n$ 或者 $\log_2 n$ 和 $\sqrt{n}$ 是分离的。

致谢

中国计算机学会
China Computer Federation

谢谢大家。