MSS 命题报告

王子昱

题目描述

维护带权点的集合，支持操作

- 合并两个集合；
- 将一个集合中，某一维坐标不大于 v 的移动到一个新集合，大于 v 的移动到另一新集合中；
- 查询一个集合中点权的最值与和；
- 将一个集合中所有点的权值增加 d。

点数和 $N \leq 5 \times 10^4$；操作数 $Q \leq 10^5$。

部分分设置

- 对于 10% 的数据，$N, Q \leq 500$；
- 对于 20% 的数据，数据中不包含分割操作；
- 对于 10% 的数据，分割操作只在所有合并操作结束后出现；
- 对于 20% 的数据，所有点满足 $x_i = y_i$。

暴力算法

- 用链表维护集合，模拟所有操作
- 时间复杂度 $O(NQ)$
- 空间复杂度 $O(N)$
- 期望得分 10 分

启发式合并

- 对于只有合并操作的情况，我们可以用平衡树维护每个集合，合并两个集合时暴力地将较小集合中的点插入较大集合中
- 由于每个点被新插入到一个集合时，它所在的集合的大小至少增加一倍，每个点至多被插入 $\log N$ 次
- 时间复杂度 $O(N\log^2 N + Q)$
- 空间复杂度 $O(N)$
- 结合暴力可以得到 30 分

“启发式分割”

- 只有分割操作时，问题可以类比启发式合并解决
- 对每个集合，维护以横坐标为序和以纵坐标为序的平衡树
- 分割一个集合时，我们可以在 $O(\log N)$ 的时间内确定分割后较小的集合
- 之后从两棵平衡树中暴力地将对应的点移动到新集合中
- 时间复杂度？

“启发式分割”

- 一个点被移动到新集合中的次数至多为 $\log N$
- 时间复杂度 $O(N\log^2 N + Q)$
- 对于第三类数据，开始时用启发式合并处理所有的合并，之后使用上述策略处理分割
- 期望得分 40 分

一维算法

- $x_i = y_i$ 时，对 y 的分割可以转化为对 x 的分割，因此问题等价于维护数的集合
- 继续用平衡树维护集合？
- 分割平衡树、合并范围不相交的平衡树的复杂度都是 $O(\log n)$
- 还需要一个一般情况下的合并算法

一维算法

```
  X = 1, 2, 3          7, 8, 9
  Y =         4, 5, 6           10, 11
X+Y = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
```

- 在一次合并中，有一部分点的前驱和后继不被改变
- 我们希望设计一种可以跳过这些点的合并策略

一维算法

```
  X = 1, 2, 3          7, 8, 9
  Y =         4, 5, 6           10, 11
X+Y = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
```

- 在一次合并中，有一部分点的前驱和后继不被改变
- 我们希望设计一种可以跳过这些点的合并策略

- `merge'(x,y) = merge'(merge(splitL(x,y.min),y),splitR(x,y.min))`
- 进行的分割、合并操作次数正比于后继改变的点的数目

一维算法

- 一个合并过程的例子：

```
merge'({1, 2, 3, 7, 8, 9}, {4, 5, 6, 10, 11}) ->
```

一维算法

- 一个合并过程的例子：

```
merge'({1, 2, 3, 7, 8, 9}, {4, 5, 6, 10, 11}) ->
merge'({1, 2, 3, 4, 5, 6, 10, 11}, {7, 8, 9}) ->
```

一维算法

- 一个合并过程的例子：

```
merge'({1, 2, 3, 7, 8, 9}, {4, 5, 6, 10, 11}) ->
merge'({1, 2, 3, 4, 5, 6, 10, 11}, {7, 8, 9}) ->
merge'({1, 2, 3, 4, 5, 6, 7, 8, 9}, {10, 11}) ->
```

一维算法

- 一个合并过程的例子：

```
merge'({1, 2, 3, 7, 8, 9}, {4, 5, 6, 10, 11}) ->
merge'({1, 2, 3, 4, 5, 6, 10, 11}, {7, 8, 9}) ->
merge'({1, 2, 3, 4, 5, 6, 7, 8, 9}, {10, 11}) ->
          {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}
```

一维算法

```
  X+Y = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
   X' = 1, 2, 3, 4, 5, 6
   Y' =                   7, 8, 9, 10, 11
X'+Y' = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
```

- 随着操作的进行，每个集合中数的差越来越小
- 可以证明算法的均摊复杂度为 $O(\log^2 N)$

一维算法

- 用动态存储的线段树维护集合
- 分割：利用线段树的结构做到 $O(\log N)$
- 合并：自顶向下遍历两棵线段树，跳过在至少一个集合中为空的结点
- 时间复杂度？

一维算法

- 势能分析
- 令数据结构的势函数为每个集合对应的线段树的结点数之和
- 分割：实际代价 $O(\log N)$，势函数增量 $O(\log N)$
- 合并：实际代价正比于遍历的点数；
  除最下面一层外，线段树中每个被遍历到的点都会减少一个拷贝，由此势函数减少量正比于实际代价
  - 均摊代价为 0
- 数据结构的初始势能为 $O(N \log N)$，算法时间复杂度 $O((N+Q)\log N)$

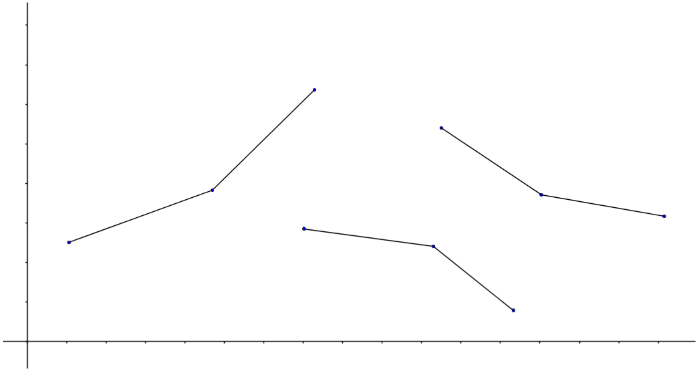
一维算法

- 势能分析
- 令数据结构的势函数为每个集合对应的线段树的结点数之和
- 分割：实际代价 $O(\log N)$，势函数增量 $O(\log N)$
- 合并：实际代价正比于遍历的点数；
  除最下面一层外，线段树中每个被遍历到的点都会减少一个拷贝，由此势函数减少量正比于实际代价
  - 均摊代价为 0
- 数据结构的初始势能为 $O(N \log N)$，算法时间复杂度 $O((N+Q)\log N)$
- 结合之前的算法，可以得到 60 分

通用算法

- 注意到只要所有的点的纵坐标随横坐标单调变化，一维算法都是适用的
- 所以我们将所有集合的并集 U 划分为集合 $U_1 .. U_k$，每个集合中点的纵坐标均随横坐标单调变化
- 维护 k 个一维算法中的数据结构，第 i 个数据结构中维护当前所有集合与 U_i 的交
- 所有操作在 k 个数据结构中分别完成
- 不考虑划分过程，算法复杂度 $O((N + Qk) \log N)$
- 如何完成划分？

通用算法



- 将所有点按横坐标排序，问题等价于将它们的纵坐标构成的序列划分为尽量少的上升子序列和下降子序列

通用算法

- 每次贪心地删去最长的上升子序列或下降子序列
- 设 $f(i)$ 为以序列中第 i 项结尾的最长上升子序列，于是 $f(i) = j$ 的所有点 S_j 构成下降子序列
- 如果 $\max(f(i)) < \sqrt{N}$，必然存在 j 使得 $|S_j| \geq \sqrt{N}$
- 我们总能删掉长度至少为 $\sqrt{N}$ 的子序列

通用算法

- 设 $k(N)$ 为按照上述策略，长度为 N 的序列划分出的序列数的最大值
- $k(N) = k(N - \lceil\sqrt{N}\rceil) + 1 = O(\sqrt{N})$
- 划分复杂度为 $O(N\sqrt{N}\log N)$，于是算法的时间复杂度为 $O((N+Q)\sqrt{N}\log N)$
- 问题得到解决

总结

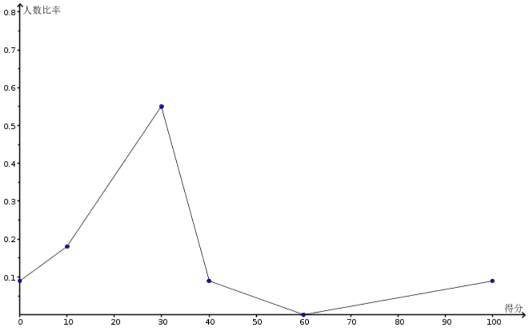
- 题目中坐标单调的情况比一般情况更容易解决，是因为关系"$A < B \Leftrightarrow x_A < x_B, y_A < y_B$（或 $x_A < x_B, y_A > y_B$）"在这种情况下是全序
- 通过将偏序集划分为链和反链，我们将一般情况转化到了存在全序的特殊情况。这种方法是可以推广的

总结

- 考察点：基础数据结构，均摊分析，数学知识

总结

- 考察点：基础数据结构，均摊分析，数学知识
- 得分情况



谢谢大家

- 欢迎提问。