解题报告

宁波市镇海中学 虞皓翔

Codeforces 626G Raffles

简要题意

有 n 个奖池，第 i 个奖池的奖金是 p_i，已经有 l_i 张彩票押在上面。

现在 Johnny 有 t 张彩票，他需要将他的彩票分配到这些奖池中，并需要保证他在每个奖池中押的彩票数**不能超过该奖池原有的彩票数**。

设它在第 i 个奖池中押了 t_i 张彩票，则他中奖的概率为 $\dfrac{t_i}{t_i+l_i}$。

你需要给 Johnny 制定一组方案，使他获得的奖金总数的期望最大。

赌场中，各个奖池的彩票数在不断地变化。一共有 q 次事件，每次事件形如：奖池 i 的彩票数量增加 1，或奖池 i 的彩票数量减少 1。

你需要在每次变化后输出 Johnny 获得的奖金总数的期望的最大值。

数据范围

$1 \le n, t, q \le 2 \times 10^5; 1 \le p_i, l_i \le 1000$。

题解

算法一 ($q = 0$)

定义**某个奖池中，第 k 张彩票的贡献**为，在奖池中押第 k 张彩票后时，所得的奖金期望与押 $k-1$ 张彩票后所得的奖金期望的差。

经过探索，可以发现一个简单的结论：当在一个固定的奖池中，押的彩票越多，单张彩票对奖金总数期望的贡献越少。

简单的证明如下：对于一个奖池，不妨假设它的奖金为 1。

则在该奖池扔第 k ($k \in \mathbb{N}^*$) 张彩票时，它对奖金总数的贡献等于

$$\frac{k}{k+l_i} - \frac{k-1}{k+l_i-1} = \frac{k\cdot(k+l_i-1)-(k-1)\cdot(k+l_i)}{(k+l_i)\,(k+l_i-1)} = \frac{l_i}{(k+l_i)\,(k+l_i-1)} \quad (1)$$

可以发现，(1) 式是一个关于 k 的减函数，证毕。

在这个条件下，就有一个贪心策略：每次选取能使当前对奖金总数期望贡献最大的 (奖池)，并向其中投入一张彩票，直到彩票用完。

下面证明这个贪心策略是可行且正确的：

- 可行性：

  由于一个奖池中，后面的彩票的贡献不超过前面的彩票的贡献，因此任意时刻，当前贡献最大的彩票一定是目前可以**押的** (即某个奖池中下一个要押的彩票)。于是持续这个过程，即可完成整个贪心。

- 正确性：

假设最优策略所取的彩票集合 S，不是所有奖池中可取的彩票集合中贡献前 t 大的，那么，取 S 中贡献最小的彩票 τ，由于一个奖池中，后面的彩票的贡献一定比前面的小，因此 τ 一定是某个奖池中贡献最小的彩票。

将其撤回后，考虑剩下的 $t-1$ 张彩票。如果这 $t-1$ 彩票是所有彩票中前 t 大的，那么，由期望的线性性，欲使奖金总数的期望尽可能大，就要使最后一张彩票的贡献尽可能大，从而彩票 τ 应取贡献最大的，从而 S 就是所有彩票中可取的彩票集合中贡献前 t 大的，与假设矛盾。

如果这 $t-1$ 张彩票不是所有彩票中前 $t-1$ 大的，设彩票 β 是没有押的彩票中最大者。那么彩票 β 的贡献一定比这 $t-1$ 张彩票中贡献最小的彩票的贡献大，从而比彩票 τ 的贡献大。

由贪心可行性知，β 这张彩票当前是可以押的，因此我们将 τ 替换为 β，得到了一个更优的方案，与假设矛盾。

综上，这个贪心算法是正确的。

因此，当 $q=0$ 时，只需要实现这个贪心即可。由于彩票的总数量比较多，因此我们需要维护一个能查询最大值的数据结构，具体可以使用优先队列 (堆) 来完成。

算法二

当有修改时，相当于单点改一个系数 (l_i)。那我们在堆中更改系数的时候，有可能当前的策略已经不是最优的了。

由上面贪心的正确性证明析知，一旦当前的策略如果不是最优的话，说明将当前最劣 (贡献最小) 的一张彩票弹回后，再选择最优 (贡献最大) 的一张彩票后，总的答案 (期望值) 会增加。

一个简单的思路是，我们暴力去取走贡献最小的彩票，将其换成贡献更大的彩票。

于是，我们还需要使用另一个堆来维护当前贡献最小的彩票，从而来实现整个问题。

对于 $t_i \le l_i$ 的限制，只需要存储当前的 t_i 后检验 t_i 是否大于 l_i，对于这个奖池 i，我们只将前 l_i 张彩票的贡献设为的 $\dfrac{l_i}{(k+l_i)(k+l_i-1)}$ (其中 $k \in [1, l_i]$ 表示是第 k 张彩票)，将编号大于 l_i 的彩票的贡献设为 0，这样，根据“选取最大”的策略，我们就就不会选它了。

类似地，如果 $t_i = 0$，则这个奖池中无法再去除彩票，因此我们可以在小根堆中设置去除这张彩票的**损失**为 $+\infty$，这样它也就无法被去除。

这里的堆需要支持修改，可以通过以下方法实现：

第一种是比较容易写的，通过线段树的单点修改和区间 (全局) 查询最值的功能，即可实现查找 min 和 max，具体过程就不再赘述了。

第二种就是一般的堆 (`priority_queue`) 来实现带修改的堆。注意到修改可以看成一次删除和一次加入，因此我们只需要实现可以删除元素的堆。

- 具体地，我们维护两个堆 Q_I, Q_D，插入元素 v 时，我们在 Q_I 中插入 v，删除元素 v 时，我们在 Q_D 中插入 v。

- 当需要查询全堆最小值时，我们不断从 Q_I 中弹出元素，同时检查它是否是 Q_D 的堆顶，如果它等于 Q_D 的堆顶，说明这个元素已经被删除了，将两个堆中同时弹出这个元素。直到 Q_I 的堆顶不等于 Q_D 的堆顶，此时说明 Q_I 确实是新的堆顶，从而直接返回即可。

- 这种可删除堆的实现方式的均摊时间复杂度是正确的，因为每个元素至多被 Q_I 和 Q_D 加入一次和删除一次，因此这种可删除堆的均摊时间复杂度是每次 $O(\log n)$ 的。

当然，也可以通过 `__gnu_pbds` 中的 `priority_queue` 或者手写二叉堆等的方法来完成这些操作，这里就不展开了。

最后来分析一下这个“暴力算法”时间复杂度。

首先，单次操作，不管是用堆还是用线段树，都是 $O(\log n)$，因此接下来需要分析一下操作次数。

其次，我们需要证明，对于某个奖池 i，如果它的彩票数量增加 1，则引起的修改次数至多为 1 次。

这是因为，设原来奖池 i 中，已经有 l 张彩票，Johnny 放了 t 张彩票，则由 (1) 式，最后一张彩票的贡献是 $F(l,t) = \dfrac{l}{(t+l)(t+l-1)}$。

这说明，在修改之前，所有贡献**大于** $F(l,t)$ 的彩票均被选取。

当彩票数量增加 1 后，即 $l \leftarrow l+1$，那么这张彩票的贡献就变成了 $F(l+1,t) = \dfrac{l+1}{(t+l+1)(t+l)} < F(l,t)$。

若剩下的没有一张贡献在 $[F(l+1,t), F(l,t)]$ 的彩票，则该方案仍为最优方案。

如果有，则将第 t 张彩票撤回，押到最优方案中，则奖池 i 要么已经没有彩票了 $(t=1)$，要么它目前贡献最小的彩票是**第** $t-1$ **张**。

而它的贡献是 $F(l+1,t-1) = \dfrac{l+1}{(t+l)(t+l-1)} > F(l,t)$。

而目前贡献最小的彩票的贡献式不超过 $F(l,t)$ 的，因此这个彩票就不再会被撤回，从而其它的彩票也不会被撤回。

综上，当奖池中的彩票数量增加 1 时，引起修改的数量至多 1 次。对于减少 1 的情况，分析也是类似的 (或者也可以通过“减少 1 是增加 1 的逆变换”来说明)。

于是，每个事件引起的修改次数为 $O(1)$，算上初始的押彩，总时间复杂度为 $O(n+(t+q)\log n)$。

Codeforces 666D Chain Reaction

简要题意

给定坐标平面上四个整点，你需要给每个点指定一个平行于坐标轴的方向 (i.e. 左、右、上、下)，并指定一个距离 d_i ($d_i \in \mathbb{N}$)，使得将四个整点分别按照它所对应的方向移动对应距离 d_i 后，成为一个**边平行于坐标轴**的正方形的四个顶点。

你需要最小化 $\max\limits_{1\le i\le 4} d_i$ 的值，或说明不存在满足条件的方案。

共有 t 组数据。

数据范围

$1 \le t \le 50; -10^8 \le x_i, y_i \le 10^8$。

题解

由于只有四个点，我们可以考虑稍暴力的做法：先枚举 $4! = 24$ 种原先的点和正方形的四个点的对应关系，并分别判断，最后取最小值。

对于每种特定的情况，我们将其分为两大类：

1. 所有点的移动方向 (水平/垂直) 一致。

   不失一般性，设所有点的移动方向均为**水平方向** (和 x 轴平行的方向)，则容易发现最终所得到的 4 个点一定位于两条不同的水平线上，且这一对平行线之间的距离就等于正方形的边长 a。

   于是，此时正方形的边长是确定的，我们需要**水平移动**四个点，使之成为一个正方形。

   由于我们已经枚举了哪两个点位于**左边**，哪两个点位于**右边**。因此，最终位于右边的点的横坐标一定比位于左边的点的横坐标大 a。从而我们可以把初始规定在右边的点的横坐标**减去** a，就变成了这样一个稍简单的问题：

   给定四个实数 a_1, a_2, a_3, a_4，求一个实数 x，使得 $\max\limits_{1\le i\le 4} |x - a_i|$ 最小。

   经过容易的推导发现，满足条件的最小 x 就是**四个数中最大数和最小数的平均数**，这个最小值就等于这四个数的**极差**的一半。

   不过，在本题中由于规定 x 是整数，因此，如果算出来的 x 是 $\dfrac{1}{2}$ 的奇数倍的话，取它最近的一个整数即可。

2. 存在两个点的移动方向不一致。

   此时，一定有一个点是水平移动的，一个点是竖直移动的。

   那么，这两条直线的交点一定是原正方形的一个顶点。

另外，由抽屉原理知，一定有两个点的移动方向相同，那么这两个点的移动方向所在直线的距离就是正方形的边长 a。

继续枚举这样的 a (由于有横坐标和纵坐标，因此至多 $2 \cdot \binom{4}{2} = 12$ 种边长)，再枚举这个交点在正方形中的位置 (共 4 种)，所以一共有不超过 $12 \cdot 4 \cdot 4^2 = 768$ 种情形。

对于每种情形，检验是容易的——只需要检验对应顶点连线是否平行于一条坐标轴即可，并可以算出 (唯一的) 答案。

因此这种情况只需要暴力检验即可完成。

最后分析总时间复杂度：

首先我们需要枚举对应关系，在确定 4! 种排列后，对于第一种情形 (所有点的移动方向一致)，只有不超过 2 种不同情形；对于第二种情形 (存在两个点的移动方向不一致)，上面已经说明不超过 768 种。于是单组数据的枚举量不超过 $4! \cdot (2 + 768) = 18480$，可以接受。

ARC 102F Revenge of BBuBBBlesort!

简要题意

给定一个 $1 \sim N$ 的排列 $p_1, p_2, \cdots, p_N$，你需要对其进行“**超级冒泡排序**”，其中“超级冒泡排序”的操作如下：

- 每次选择满足 $p_{i-1} > p_i > p_{i+1}$ 的下标 i $(2 \le i \le N-1)$，并将 p_{i-1} 和 p_{i+1} 交换。

不难发现，有些排列是无法通过“超级冒泡排序”将其变为有序的。你需要判定给定的排列是否能通过“超级冒泡排序”使得它变为升序序列。

数据范围

$3 \le N \le 3 \times 10^5$。

题解

我们逆着考虑这个问题，即对于一个有序序列，考虑对 $p_{i-1} < p_i < p_{i+1}$ 的三元组，交换两侧元素，看看能得到哪些序列。

首先，我们需要证明一个基本的性质：

如果 i 位置进行了一次操作，则 $i-1$ 和 $i+1$ 位置将来不会再进行操作，从而 p_i 的值不再改变。

- 反设结论不成立。由对称性，设 i 操作完成后，$i-1$ 和 $i+1$ 中**较早操作者**为 $i+1$ (当然也包含 $i-1$ 未操作的情形)。
- 设这个 i 操作的时刻与最早的 $i+1$ 操作的时刻的**时间间隔**为 t，我们对 t 进行归纳证明。
- 当 $t = 1$ 时，结论显然。
- 设结论对小于 t 的正整数成立，考虑 t 的情形。
- 首先，由于 i 操作完成后，有 $p_{i-1} > p_i > p_{i+1}$，从而更有 $p_i > p_{i+1}$。
- 在 $i+1$ 操作前，由题意，应有 $p_i < p_{i+1} < p_{i+2}$，从而有 $p_i < p_{i+1}$。
- 又由假设，在这段过程中，$i-1$ 不操作，从而 p_i 的值不会改变。
- 于是，p_{i+1} 的值一定发生了改变，如果改变它的操作为 i，则这一个 i 的操作与 $i+1$ 的操作的时间间隔小于 t，与归纳假设矛盾；如果改变它的操作为 $i+2$，由对称性，这个 $i+2$ 的操作与 $i+1$ 的操作的时间间隔也小于 t，从而也得到矛盾。
- 因此结论对 t 成立，由归纳原理可知，原结论成立。

由于 i 操作后 $i-1$ 与 $i+1$ 不再进行操作，由对称性可知，i 操作前 $i-1$ 与 $i+1$ 也不会进行操作。

这样一来，我们得到了：**相邻两个下标中，至多一个下标进行过操作**。

于是，由这个结论可知，我们可以把所有**操作过的下标**分为若干个**极长段**：每个极长段为一个**公差为 2 的等差数列**，不同极长段之间至少**间隔 2 个数** (即后一段的首项减去前一段的末项至少是 3)。

由于不同极长段之间的下标差至少为 3，因此它们对原序列的影响是**独立不相交的**。

于是我们只需要考虑一个极长段中的情形。

不妨序列长度为 $2k-1$ ($k \in \mathbb{N}^*$)，只有一个段：$2, 4, 6, \cdots, 2k-2$ ——其中位置 $1, 3, 5, \cdots, 2k-1$ 从未进行过操作，$2, 4, 6, \cdots, 2k-2$ 都进行过操作。

于是，位置 $2, 4, 6, \cdots, 2k-2$ 为**不动点**，即自始至终这些位置上的数都不改变。

因此，将这 $k-1$ 个不动点去掉后，每次操作可以看成是在一定条件下，交换相邻的两个数。

接下来我们分析一个奇数的**轨迹**。下面证明，一个数一定运动且以**恒定的方向运动**。

- 不妨设某一个下标为 t 的数在某个操作后向右移动了，即 $(t, 2i, s) \to (s, 2i, t)$，则有 $t < 2i$，且 t 在 $2i$ 的右边。

  则 t 不会移动到 $2i$ 的左边，这是因为如果 t 要回到左边，则一定会经过 $2i$，从而 $2i$ 又需要一次操作，而因为 $2i > t$，所以这次操作是不可能成功的。

- 于是，每个数的移动轨迹是 (非严格) 单调的。接下来我们证明一个数不可能不动。

  如果一个奇数下标 t 的数不动，则 $t-1$ 是 $t+1$ 不能进行操作的，否则将会导致 t 运动，这样就回不来了。因此偶数位置 $t-1$ 和 $t+1$ 都未进行过操作，从而与假设“$2, 4, 6, \cdots, 2k-2$ 都进行过操作”矛盾。

其次，如果两个数以相同的方向运动，它们的相对顺序不会改变。

- 因为如果两个向右运动的顺序交换了位置，则它们一定会碰到一起，于是一定有一个元素向左移动，这就产生了矛盾。

同时，这两个条件也是充分的——即对集合 $S = \{1, 3, \cdots, 2k-1\}$ 的任意一个划分 $S = L \cup R$ ($L \cap R = \varnothing$)，将 L 中的元素 (保持原序地) 向左运动，R 中的元素 (保持原序地) 向右运动，则所能得到的任一种情况都是**可达的**。

下面简单的证明一下：我们将 L 中元素看成“棋子”，R 中元素看成“空位”，考虑每次将一个棋子移到位于它左边的空位，直到棋子都移动到目的地。

首先，由于每次是将一个棋子移动到它左边的空位，因此 L 中元素是向左移动，R 中元素是向右移动的，因此这个交换是符合题目规则的。

又由于 L 中的所有元素都是**保持原序地**向左运动，因此第 k 个棋子的最终下标一定不超过原来的下标，整个过程是可以持续下去直到完成目的地的。

最后总结一下算法：

对于一个可达的排列，可以通过对应位置是否改变，将其分为若干个”段”：每个段要么所有元素都是不动点；要么首、尾都改变了，且中间的元素改变和不改变依次出现；

且改变的元素中，变大的元素单调 (即向左移动的元素)，变小的元素 (即向右移动的元素) 单调，且整个段是一个**连续段** (即下标**集合**和值**集合**相等，也就是说，整个段中元素不会移动到外面去)。

由上面已经说明，可知这就是一个序列“可以完成冒泡排序”的充要条件。

实现时，直接贪心寻找极长段后进行判定即可，注意不要漏写条件。时间复杂度 $O(N)$。