《Grievous Lady》解题报告

杭州学军中学教育集团文渊中学 叶开

引言

本题原本为我给为我们联考 NOI 模拟赛投的 T3，结果最后被胡伟栋老师毙了，故我将本题放到互测来作为备用题。

本题最初做法是 $O(n^2)$ 左右的，后来周康阳同学将本题复杂度优化到了 $O(n\sqrt{n})$ 甚至优化到了 $O(n\log n)$，非常感谢他的帮助！

本题正解也饶有趣味，希望大家会喜欢！

题目大意

共有 n 个元素，标号 $1\sim n$，每个元素 j 有两个正整数权值 a_j,b_j。

你要确定一个 $[1,n]\cap\mathbb{N}$ 的子集 S，从而最大化

$$(\sum_{k=1}^{n}a_k[k\in S])(\sum_{k=1}^{n}b_k[k\notin S])$$

保证每个 a_j,b_j 分别在 $[1,A]\cap\mathbb{N},[1,B]\cap\mathbb{N}$ 内独立均匀随机生成。

现在给定 n,A,B 和每个元素的两个权值 (a_j,b_j)，请求出这个最大的答案。

数据范围

时间限制：2s。

空间限制：1GB。

本题每个测试点内部有 T 组测试数据。

对于所有的数据，保证 $10 \le n \le 3000$，$10^4 \le A, B \le 10^{12}$，$1 \le T \le 50$。

具体的测试点分布可以见下表，各测试点等分。

测试点编号	n	A	B	测试点编号	n	A	B
$1 \sim 2$	$= 10$	$= 10^4$	$= 10^4$	$55 \sim 56$	$= 500$	$= 10^5$	$= 10^{12}$
$3 \sim 4$	$= 10$	$= 10^9$	$= 10^9$	$57 \sim 58$	$= 500$	$= 10^9$	$= 10^9$
$5 \sim 6$	$= 10$	$= 10^{12}$	$= 10^{12}$	$59 \sim 60$	$= 500$	$= 10^{12}$	$= 10^{12}$
$7 \sim 8$	$= 20$	$= 10^4$	$= 10^4$	$61 \sim 62$	$= 800$	$= 10^5$	$= 10^{12}$
$9 \sim 10$	$= 20$	$= 10^9$	$= 10^9$	$63 \sim 64$	$= 800$	$= 10^9$	$= 10^9$
$11 \sim 12$	$= 20$	$= 10^{12}$	$= 10^{12}$	$65 \sim 66$	$= 800$	$= 10^{12}$	$= 10^{12}$
$13 \sim 14$	$= 30$	$= 10^4$	$= 10^4$	$67 \sim 68$	$= 1000$	$= 10^5$	$= 10^{12}$
$15 \sim 16$	$= 30$	$= 10^9$	$= 10^9$	$69 \sim 70$	$= 1000$	$= 10^9$	$= 10^9$
$17 \sim 18$	$= 30$	$= 10^{12}$	$= 10^{12}$	$71 \sim 72$	$= 1000$	$= 10^{12}$	$= 10^{12}$
$19 \sim 20$	$= 40$	$= 10^4$	$= 10^4$	$73 \sim 74$	$= 1500$	$= 10^5$	$= 10^{12}$
$21 \sim 22$	$= 40$	$= 10^9$	$= 10^9$	$75 \sim 76$	$= 1500$	$= 10^9$	$= 10^9$
$23 \sim 24$	$= 40$	$= 10^{12}$	$= 10^{12}$	$77 \sim 78$	$= 1500$	$= 10^{12}$	$= 10^{12}$
$25 \sim 26$	$= 50$	$= 10^4$	$= 10^4$	$79 \sim 80$	$= 2000$	$= 10^5$	$= 10^{12}$
$27 \sim 28$	$= 50$	$= 10^4$	$= 10^9$	$81 \sim 82$	$= 2000$	$= 10^9$	$= 10^9$
$29 \sim 30$	$= 50$	$= 10^9$	$= 10^9$	$83 \sim 84$	$= 2000$	$= 10^{12}$	$= 10^{12}$
$31 \sim 32$	$= 50$	$= 10^{12}$	$= 10^{12}$	$85 \sim 86$	$= 2500$	$= 10^4$	$= 10^9$
$33 \sim 34$	$= 100$	$= 10^4$	$= 10^4$	$87 \sim 88$	$= 2500$	$= 10^5$	$= 10^{12}$
$35 \sim 36$	$= 100$	$= 10^5$	$= 10^5$	$89 \sim 90$	$= 2500$	$= 10^9$	$= 10^9$
$37 \sim 38$	$= 100$	$= 10^5$	$= 10^9$	$91 \sim 92$	$= 2500$	$= 10^{12}$	$= 10^{12}$
$39 \sim 40$	$= 100$	$= 10^9$	$= 10^9$	$93 \sim 94$	$= 3000$	$= 10^4$	$= 10^9$
$41 \sim 42$	$= 100$	$= 10^{12}$	$= 10^{12}$	$95 \sim 96$	$= 3000$	$= 10^5$	$= 10^{12}$
$43 \sim 44$	$= 200$	$= 10^5$	$= 10^{12}$	$97 \sim 98$	$= 3000$	$= 10^9$	$= 10^9$
$45 \sim 46$	$= 200$	$= 10^9$	$= 10^9$	$99 \sim 100$	$= 3000$	$= 10^{12}$	$= 10^{12}$
$47 \sim 48$	$= 200$	$= 10^{12}$	$= 10^{12}$				
$49 \sim 50$	$= 300$	$= 10^5$	$= 10^{12}$				
$51 \sim 52$	$= 300$	$= 10^9$	$= 10^9$				
$53 \sim 54$	$= 300$	$= 10^{12}$	$= 10^{12}$				

实际互测时由于数据资源限制，这些测试点被分批绑成了若干个包。

解题过程

本题由于数据随机比较玄学，分布很古怪，只能打表说明复杂度了。

$O(n^3)$ 算法

考虑到，每一种选法都对应一个 $(\sum A, \sum B)$ 对，我们发现如果有两个对构成严格偏序，被偏序的那一个肯定对答案不做贡献。

因此我们对 n 增量，每次用单调队列维护出当前最优 $(\sum A, \sum B)$ 对。

打个表容易发现单调队列中的元素个数是 $O(n^2)$ 级别的，每次插入一个元素只用归并，复杂度 $O(n^3)$。

仔细观察一下打表所得数据，容易猜测元素个数是 $O(n^{2-\varepsilon})$ 的，这样复杂度即为 $O(n^{3-\varepsilon})$ 了。

$O(n^2)$ 算法

考虑沿用上面的做法，但是我们注意到仍然有点对是没用的。

如果 (a_1, b_1) 比 (a_2, b_2) 在之后选择 (x, y) 时的方案更优，那么我们就有

$$(a_1 + x)(b_1 + y) \geq (a_2 + x)(b_2 + y)$$

也即

$$a_1 b_1 + a_1 y + b_1 x \geq a_2 b_2 + a_2 y + b_2 x$$

假设 (a_1, b_1) 在所有情况下都不如 (a_2, b_2) 或 (a_3, b_3)，则有

$$\exists (x, y, z)' \geq \mathbf{0} \wedge (x, y, z)' \neq \mathbf{0}, \begin{bmatrix} a_1 b_1 - a_2 b_2 & a_1 - a_2 & b_1 - b_2 \\ a_1 b_1 - a_3 b_3 & a_1 - a_3 & b_1 - b_3 \end{bmatrix} (x, y, z)' \leq \mathbf{0}$$

我们在单调队列维护的同时判断一下是否和上下两个元素同时满足该式即可。

具体的判断方式类似于斜率优化的过程，但是要对三维两两之间各判一次。

最后会形成一个类似于凸壳的结构。

打个表容易发现单调队列中的元素个数是 $O(n)$ 级别的，增量 $O(n)$ 轮，总复杂度即为 $O(n^2)$。

$O(n\sqrt{n})$ 算法

注意到答案选择的集合期望只有 $n/2$ 个元素，然后误差大概率不会超过 $2\sqrt{n}$。设 $T = 200$，假设考虑了前 p 项，我们在状态中记录下当前状态中的 $|S|$，仅保留 $|2|S| - p| \leq T$ 的部分，就可以了。实际取 $T = 150$ 即可通过。

正确率分析可以直接套数轴上随机游走回归原点的证明，类似于 THUPC2021 中出现的混乱邪恶一题。使用 dp 也可以算出答案错误的概率是极低的。

打个表容易发现单调队列中的元素个数是 $O(\sqrt{n})$ 级别的，增量 $O(n)$ 轮，总复杂度即为 $O(n\sqrt{n})$。实际上这样就可以通过本题了。

$O(n \log n)$ 算法

我们考虑不采用增量法，而采用分治法，每次维护出左右两侧的凸壳，暴力求出其合并起来的每组结果，维护出单调队列，然后使用上一个做法的结论只保留一部分集合。

这样暴力合并的复杂度即为 $O(\sqrt{n} \times \sqrt{n}) = O(n)$，由主定理可得，复杂度即为 $T(n) = 2T(n/2) + O(n) = O(n \log n)$。

朴素实现是 $O(n \log^2 n)$ 的（合并的时候要排序），需要通过 $O(\sqrt{n})$ 次二路归并并同时维护单调栈，复杂度即为 $O(n \log n)$。

实际上这个做法的常数未必比上一个优秀。

其它做法

以下列举几种做法。

- 直接枚举所有情况然后判断是否合法即可 $O^*(2^n)$。
- 直接背包即可 $O(n^2 v)$。
- 把其中一维除以 10^6，然后背包。最后找到最优解可能在哪些位置，回推一下方案即可。可以预见有效的状态期望不会很多。
- 推广上一种做法，可以对所除的数从 10^{12} 开始倍减来逐步缩小状态，最后爆搜。
- $O(n^3)$ 算法没有写增量，而是写成了分治，退化成 $O(n^4)$ 或者 $O(n^4 \log n)$。
- 凸壳只保留中间 300 个点。这个也是能过的，复杂度也是 $O(n\sqrt{n})$。
- 调整法 / 模拟退火。复杂度未知，正确率未知。
- 等等。

代码实现

对于维护凸壳时判断是否合法部分，周康阳同学给出的代码实现是这样的：

```
#define i128 __int128
array < i128, 3 > make(pair < ll, ll > z) {
    array < i128, 3 > F;
    F[0] = (i128)z.first * z.second;
    F[1] = z.first;
    F[2] = z.second;
    return F;
}
bool DD(pair < ll, ll > L, pair < ll, ll > M, pair < ll, ll > R) {
    array < i128, 3 > a[3] = {make(L), make(M), make(R)};
    L(i, 0, 2) if(a[1][i] > a[0][i] && a[1][i] > a[2][i]) return false;
    double l = 0, r = 1;
    L(i, 0, 2) {
        if(a[1][i] <= a[0][i] && a[1][i] <= a[2][i]) continue;
        i128 x = a[1][i] - a[0][i];
        i128 y = a[1][i] - a[2][i];
        if(x > 0) {
            y = -y;
            r = min(r, 1. * y / (x + y));
        } else {
            x = -x;
            l = max(l, 1. * y / (x + y));
        }
    }
    return r - l >= 1e-9;
}
```

由于 `__int128` 和 `double` 运算常数实在太大，所以要做出一些小常数优化才可以通过本题；直接按上述方法实现有概率被卡常。

参考资料

感谢周康阳同学对本题更优做法的提供。