怦然心动 解题报告

南京市第一中学 邱梓轩

2025 年 10 月

1 题目大意

给定两个长度分别为 n, m 的正整数序列 $a_1, \cdots, a_n$ 和 $b_1, \cdots, b_m$。有两个人进行博弈，交替进行操作，每次操作可以选择 a 或 b 中的一个元素将其删去。

在博弈开始前，两人确定一个阈值 B。在任意时刻，如果某一个序列被删空或 $\forall_{1\le i\le n,1\le j\le m} a_i+b_j \le B$，后手获胜。如果 $\forall_{1\le i\le n,1\le j\le m} a_i + b_j > B$，先手获胜。

求最优策略下，最小能够使得后手获胜的阈值 B。对于部分数据，你作为后手，需判定一个局面能否获胜，如果可以还需要与交互库进行博弈并最终获胜。

2 数据范围

共有 T 组多测，对于全部数据，$\sum n + m \le 5 \times 10^5, 1 \le a_i, b_i \le 10^9$。

部分分如下表，其中求值和交互下的数分别表示两个部分在对应 n, m 数据范围下的分值。每个子任务依赖所有在其左上方的子任务。

$n + m \le$	$\sum n + m \le$	求值	交互
50	1000	3	2
500	5000	13	7
2500	10^4	15	10
10^5	5×10^5	14	8
5×10^5	5×10^5	12	16

3 解题过程

3.1 算法 1

显然序列中数的顺序和博弈的结果无关，所以不妨设序列元素单调不降 $a_1 \le a_2 \le \cdots \le a_n$，$b_2 \le b_2 \le \cdots \le b_m$。

我们首先限定一下两个人的操作，一个直觉是先手希望尽可能删除小的元素，后手则反之。事实上，我们有下述结论：

定理 3.1. *一次操作中，先手只会选择删除 a_1 或 b_1，后手只会选择删除 a_n 或 b_m。*

证明. 由于对称性,只证明先手一侧的结论。不妨设先手选择删除 $a_i (i > 1)$。此时令得到序列为 $a'_1, a'_2, \cdots, a'_{n-1}$。而先手删除 a_1 后的序列为 $a''_1, a''_2, \cdots, a''_{n-1}$。由于 $a_1 \le a_2 \le \cdots \le a_n$，对于任意 k 均有 $a'_k \ge a''_k$。

考虑任意对于局面 (a', b), 先手能取得最优结果的操作策略。如果现在的局面变为 (a'', b), 先手可以重复 (a', b) 的策略。由于 $\forall k, a'_k \ge a''_k$, 在任意时刻, $\max a''_i + b_j \ge \max a'_i + b_j$, 且 $\min a''_i + b_j \le \min a'_i + b_j$。故在 (a', b) 的博弈结束之前，(a'', b) 一定不会以更劣的结果结束。即删除最小元素不会比删除其它元素更劣。

□

容易看出博弈的结果具有单调性。具体地：

定理 3.2. *如果对于一个 B，博弈的结果为先手获胜，则 $B-1$ 的结果同样为先手获胜。同样地，如果对于一个 B，博弈的结果为后手获胜，则 $B+1$ 的结果同样为后手获胜。*

这个结论的证明与上面的结论是相似的，只需要让获胜者在阈值更紧的时候重复之前的策略即可，故不再展开。因此我们可以二分答案，下面考察对于一个固定的阈值 B，如何判定博弈的结果。

现在两个序列时刻都是原序列的子区间，故一个状态可以用 (l_1, r_1, l_2, r_2) 表示。直接对这个状态进行 dp，复杂度 $O(n^2m^2 \log v)$。期望得分 2.

对于交互部分，如果每一步重新计算一遍 dp，复杂度为 $O(n^2m^2(n+m))$。不过注意到即使交互库没有做最优操作，由于一开始我们是能够获胜的，所以如果交互库删除了任意一个 a 的元素，我们将其视作删除了第一个元素，剩余的局面只会更强，但我们仍然能够保证必胜，故可以这样假定。对 b 也是同理。

因此我们可以认为交互库始终在做最优操作，即只会删除第一个元素，因此任何时刻的局面都被 dp 算过了，可以直接查表得到操作。复杂度 $O(n^2m^2)$，期望得分 5.

3.2 算法 2

由于先手只操作左边，后手只操作右边，所以 $l_1 + l_2$ 与 $(n - r_1 + 1) + (m - r_2 + 1)$ 之差只可能是 0 或 1，故状态可以缩减一维，复杂度 $O(n^2m \log v)$，交互部分复杂度 $O(n^2m)$。

进一步注意到 dp 数组只有 01，所以可以使用 bitset 优化，期望得分 25.

3.3 算法 3

我们对这个博弈做一个转化：考虑一个 $(n+1)\times(m+1)$ 的网格图，左下角格子记为 $(0,0)$，右上角格子记为 (n,m)。令 $a_0=b_0=-\infty, a_{n+1}=b_{m+1}=\infty$，我们称一个格子 (i,j) 为黑色，当且仅当 $a_i+b_j\le B$ 且 $a_{i+1}+b_{j+1}>B$。可以发现，黑色格子形成了一条 $(n,0)\to(0,m)$ 的只向左和向上的折线，且其左下方为 $a_i+b_j\le B$ 的区域，右上方为 $a_i+b_j>B$ 的区域。

现在这个博弈可以改写为下面这个形式：初始先手站在 $(0,0)$ 处，后手站在 (n,m) 处。每一步先手可以向右或向上移动一格，后手可以向下或向左移动一格。如果某一次移动后两人的横坐标或纵坐标相同，则后手获胜。否则如果先手走到了黑色格子，则他获胜。

分析一下这个转化之后的问题为什么等价于原问题。可以发现先手站在 (lx,ly)，后手站在 (rx,ry) 时的局面相当于原序列中剩余 $a[lx+1,rx]$ 和 $b[ly+1,ry]$ 这两个子序列的子问题。先后手的两种操作即分别对应删除 a_1,b_1 和 a_n,b_m。

故为了说明等价性，我们只需要说明判定胜负的方式也与原问题相同即可。

定理 3.3. *原问题中的获胜者与新问题中的获胜者相同。*

证明. 令先手的坐标为 (lx,ly)，后手的坐标为 (rx,ry)。

如果在任意时刻两人横坐标或纵坐标相同，则说明 $lx=rx$ 或 $ly=ry$，此时存在序列为空，故后手获胜。

否则如果先手位于一个黑格子上，说明 $a_{lx+1}+b_{ly+1}>B$，即先手获胜。

注意上述分析过程中没有出现后手通过 $\forall_{1\le i\le n,1\le j\le m}a_i+b_j\le B$ 获胜的情形。如果在某一时刻后手位于黑格子上，此时 $a_{rx}+b_{ry}\le B$，即后手获胜。不过按照我们的判定方法，博弈会继续进行。然而此时先手若想走到黑格子，一定要越过后手的位置。但两人每次只将坐标改变 1，所以一定存在一个时刻两人某一个坐标相同。故即使游戏继续进行，也一定是后手获胜。

如果从博弈的开始到现在都没有出现过上述情况，根据黑格子的定义，始终有 $a_{lx+1}+b_{ly+1}\le B$ 且 $a_{rx}+b_{ry}>B$，即博弈还没有结束。

□

下面我们只需解决这个转化后的博弈问题即可。从整体观察一下这个博弈，后手获胜的条件是比较容易的，他只需要在行或者列的任意一个中走足够远的距离，就能赶上先手。但先手必须要走到一个黑格子，而这对于行列均有要求。

因此我们猜测后手的策略大概形如，先往一个方向一直走从而在这个方向上给先手带来很多不能走的限制。直到某一个时刻，他开始兼顾两个方向，此时他会和先手走对称的走法（即先手往右走，他就往左走，先手往上走，他就往下走）。

下面我们形式化地描述一下后手的策略，并给出正确性的证明。后手在博弈开始前先确定一个方向，以及一个阈值 T。如果选择的方向为横向，则：如果先手操作后他的横坐标仍不足 T，后手就向下走一步，否则他走和先手对称的方向（如果先手向上，则他向下；如果先手向右，则他向左）。对称地，如果选择的方向为纵向，则他会在先手的纵坐标 $\leq T$ 时一直向左走，否则和先手走对称的方向。

定理 3.4. *原问题中后手获胜，当且仅当存在一个方向与阈值 T，使得无论先手采用何种策略，后手均可按照上述策略获胜。*

证明. 首先如果后手采用这种策略能够获胜，则显然原问题中他也能获胜，因为这个策略并没有对先手的操作进行限制。下面重点关注原问题必胜时，这种策略是否也一定能获胜。

对序列长度归纳。首先可以手动检查一些边界情况，例如初始已经结束，只操作一次就结束，或只有两行或两列的情况。下面假定网格充分大，并且在足够多步内博弈不会结束。

我们接下来给出一种根据子问题的策略推出新问题的策略的方式，并证明这种策略无论对于先手的何种策略都是必胜的。

我们首先认为先手在这一步中向右移动，并考察移动之后的情形。由于原问题后手必胜，所以要么他向下走之后剩下的子问题也是后手必胜，要么他向左走之后剩下的子问题后手必胜。无论哪种情况，考虑走了这一步之后的子问题。根据归纳假设，此时后手一定有必胜策略，形如 “选择一个方向和阈值 T”。则我们在原问题中将方向设为这个子问题策略的方向，如果这个方向和后手第一步移动的方向一致，则阈值设为 T，否则设为 $T-1$。下面我们只需证明采用这个策略一定是必胜的。

按照先手的方向和后手策略中的方向分类讨论：

1. 先手第一步向右移动，并且当前选定的方向为向下。此时如果原来后手第一步就是向下，那么策略完全相同，结论显然成立。否则后手会始终位于原策略的右下 1 个位置，直到先手突破 $T-1$ 这一列。此时原来的策略会向下走，而当前的策略会向左走，故从这里开始路径完全重合了。一个特别的情形是如果先手在突破 $T-1$ 列之前就已经输了，不过此时一定是因为后手与他处于同一行，因为后手根本不会向左走，所以当前策略能更快地击败先手，也是合法的。

2. 先手第一步向右移动，并且当前选定的方向为向左。这和上一种情况是对称的。

3. 先手第一步向上移动，并且当前选定的方向为向下。类似于第一类情况的证明，如果先手突破了 $T-1$ 列，那么他第一步是向上还是向右并不重要，所以可以推到第一种情况。否则如果先手没有突破 $T-1$ 列，他唯一能和第一步向右产生区别的地方在于，如果他一直向上走而从不向右，则他能走到的地方是第一步向右所走不到的。

但在这种策略下，由于先手一直无法突破 $T-1$ 列，所以后手只会不断向下走。因此只要第一列最低的一个黑格子不在下半部分，后手采用当前策略是必胜的。但如果最低的格子在下半部分，唯一可能使得后手获胜的方式是当网格的高大于网格的长。由于对称性可以不妨假设不会出现这种情况，严格的说明在讨论结束后给出。

4. 先手第一步向上移动，并且当前选定的方向为向左。大部分情况和上一种是一致的，唯一的区

别在于此时先手可以从第 0 列突破 $T-1$ 这个界，这是他向右走无法做到的。但他如果第一步向右，并始终保持在当前行进路径的右下一个格子，直到当前策略向右走了一步，可以发现这两条路径是等价的。所以仍然只有一直保持在第一列的情况是特别的，与上面一种情况同理。

最后需要解决上面的这个小问题，即对于一直保持在一列的情况，如果整个网格特别高又特别窄，那么其实后手只要一直向左走，就可以保证获胜，即使第一列存在一个特别靠下的黑点。

注意到我们在最一开始决定策略的时候，使用的是先手向右移动的这个子问题。那么如果整个网格的高度大于宽度，我们可以翻转一下网格，即相当于选择先手向上移动的子问题，即可避免之前的那个小问题。这个决策只依赖于网格的长和宽，所以可以在后手制定策略的时候被考虑进去，故不影响正确性。

□

综合上述分析，我们得到了后手的策略：确定一个方向以及一个阈值 T，在先手的坐标不足 T 时一直沿着另一方方向走，否则走和先手对称的操作。

如果确定了方向和阈值，则对于先手来说，他希望能找到一个黑格子，使得走到这个格子的路径上不会被后手追到。不妨设后手选定的方向为横向，则对于一个在 T 列之前的黑格子 (x,y)，先手能走到它当且仅当 $x+y\le b-y$。

而对于一个在 T 列之后的黑格子 (x,y)，如果先手在 (T,k) 这个位置突破 T 这一列，则后手会位于 $(a,b-(T+k))$ 这个位置。此时先手要想获胜，就要求 (x,y) 距离它比距离后手更近，即 $x-T+y-k\le a-x+b-(T+k)-y$。特别地，如果它恰好位于两个点的正中间，即 $x-T=a-x$ 且 $y-k=b-(T+k)-y$，此时先手若想走到它，必须要在某一步和后手同一行/同一列，故是不合法的。可以发现上述判定标准和 k 无关，所以也可以得到一个只和 x,y,T 有关的限制。

这样我们就得到了一个复杂度 $O(n^2)$ 的做法：枚举方向和阈值，然后暴力枚举每个黑格子 (x,y)，判定是否合法。

上述做法的复杂度 $O((n+m)^2\log v)$，注意到上述做法直接能给出一个策略，所以只需一开始找到对应的方向以及阈值 T，然后对于交互库的操作，按照是否越过 T 分类给出应对即可。交互部分复杂度也为 $O((n+m)^2)$，期望得分 50.

3.4 算法 4

注意到确定了方向之后，每个格子 (x,y) 会使得 $O(1)$ 个区间内的 T 变得不合法。所以我们枚举 (x,y)，通过差分维护每个 T 被标记为不合法的次数，最后找一个合法的 T 即可。这样复杂度可以做到 $O((n+m)\log v)$，交互部分复杂度 $O(n+m)$，期望得分 100。

4 其它部分分做法

有一些针对算法 2 中 dp 的思考和优化，例如猜测 dp 中合法的位置形成一段区间。这样的结论并不正确，也容易给出反例。但如果综合考虑这些结论，加上一些剪枝和优化，例如只保留最后一段区间，实际上可以通过 $O(n^2)$ 的部分分。对于这些做法，出题人无法给出反例或给出证明，但从直观上，这些做法已经初步具备了正解所需要的感觉，所以如果从这些方法入手，也可能能够得到正解。

对于 $\sum n + m \leq 10^5$ 的部分，存在一些能够分析出与正解相似的性质，但没有想到这个策略的做法。这些做法可以使用数据结构优化 dp 的方式得到 $O(\text{polylog}(n))$ 或带根号的做法，不过对于交互部分需要多一个 n 的复杂度。由于这些做法相比于正解，实现起来过于复杂，在上述题解中不再呈现，可以留作思考。

5 交互库的实现

本题出题过程中的一大难点在于交互库的实现。实际上最终使用的交互库代码长度是 std 代码长度的数倍，下面给出交互库的参考实现方式。

首先对于前三档部分分，由于 $n + m$ 较小，故交互库采取的策略是每一步都暴力判定一下当前是否仍是后手必胜。如果在任何时刻是先手必胜，那么暴力枚举下一步使得后手最终无法获胜。

然而在先手无法获胜时，如何尝试诱导后手走出错误的操作是比较困难的。这个问题没有必然的解法，需要针对后手的策略针对性构造，所以交互库选择了几种较为通用的策略。例如每次在网格上选择距离先手最近的一个黑格子，以两维的距离为比例随机走一步，以及每次更倾向于迎着后手的方向或走与其相反的方向等。

对于后两档部分分，没有办法每次进行判定，故交互库采取的策略是一开始就锚定一个距离最近的点，然后一直向着这个方向前进。

经过实际测试，这些策略足以卡掉笔者想出的所有错误解法。不过由于不同的策略需要使用不同的方式进行针对性的应对，确实可能会出现交互库无法将错误的解法诱导进入不合法答案的情况，也欢迎大家赛后提出潜在的错误解法，并尝试将它们卡掉。

具体的交互库策略将不会公开，以防止有人针对交互库的策略使用错误的做法通过本题。

6 总结与致谢

本题由一道 QOJ 的题目 [1] 受到启发改编并加强而来。原来的题目经过转化最后的部分与本题相似，均为在网格上的博弈问题，但原题给出的做法仅能进行单次判定。出题人经过深入思考后得到了这样一种简单的策略，不仅能快速完成判定，还可以多次交互完成整个博弈的过程。

本题思维难度较大，但代码量很小，核心部分的代码只有几行，如果思考清楚可以在很短时间内完成。实际上这个问题仍能拓展，例如，目前先手方向的交互问题仅能通过较复杂的数据结构维护，并不优美。既然后手存在很简单的策略，可以尝试思考是否存在简洁的先手的必胜策略？如果针对拓展问题有更多想法，欢迎大家进行交流。

感谢许淇文、赵海鲲同学参与本题的验题工作。

参考资料

[1] QOJ. *Glass Stepping Stones*. URL: https://qoj.ac/contest/1917/problem/10088.