第 51 届 ICPC 亚洲区域赛（成都站）题解

The 2026 ICPC Asia Chengdu Regional Contest - Tutorial

电子科技大学出题组

UESTC

2026 年 10 月 25 日

预期难度

- Easy: D, K, M
- Easy-Mid: C, I
- Mid: A, E, G, H
- Mid-Hard: B, F
- Hard: J, L

D. Duel of Wrath

题目大意

两名观者 A, B 在决斗，基础攻击力分别为 a, b，初始生命值分别为 h_A, h_B。
每轮双方各处于普通或愤怒姿态，愤怒使其造成的伤害和受到的伤害都翻倍。
双方同时攻击并扣血，回合结束若有一方生命值小于等于 0，战斗立即结束。
只有 B 死亡且 A 存活才算 A 获胜。判断无论每轮双方姿态如何变化，A 是否都能获胜。
$1 \leq a, b, h_A, h_B \leq 10^{18}$，$1 \leq T \leq 10^5$。

- *Standard Solution: 394 bytes (Cpp)*

- 决斗进行若干轮后，A, B 受到的伤害可以分别写成 $S \cdot b$ 和 $S \cdot a$，其中 S 是一个正整数。
- 假设决斗在某一轮结束。上一轮结算时，B 的生命值仍然大于 0，满足 $S \cdot a < h_B$，此时 S 至多达到 $S_{\max} = \left\lfloor \dfrac{h_B - 1}{a} \right\rfloor$。不难发现，只要双方此前一直保持 Normal 状态，这个上界总是能取到。
- 而在决斗的最后一轮，S 至多再增加 4，故整场战斗 A 受到的伤害至多是 $(S_{\max} + 4) \cdot b$。
- 综上，A 在所有情形下均能获胜的充要条件为 $h_A > (S_{\max} + 4) \cdot b$。
- 单组数据时间复杂度 $O(1)$。

M. Minimum Records

题目大意

给定 n 个区间 $[l_i, r_i]$，从每个区间选一个整数 a_i，构成序列 $\{a\}$。
若 a_i 严格大于此前的所有元素，则称位置 i 刷新了纪录。位置 1 总是刷新纪录。
求可以达到的最少纪录次数。$1 \le n \le 10^6$，$\sum n \le 10^6$，$-10^9 \le l_i \le r_i \le 10^9$。

- *Standard Solution: 822 bytes (Cpp)*

- 每当选择一个新的极大值位置时，都可以将该位置的值增大到区间右端点。这样不会改变它已经是极大值位置的事实，并且只会让后面的选择更容易。因此我们只需要考虑把极大值位置取为对应的 r_i。
- 设当前已经确定的最大值为 M。从上一次确定答案之后开始扫描，并维护这一段内出现过的最大右端点 R。
- 若 $l_i \leq M$，则位置 i 可以选择一个不超过 M 的整数，因此暂时不必增加极大值位置。此时只需要用 r_i 更新 R。
- 若 $l_i > M$，那么位置 i 不可能选择一个不超过 M 的数。因此，从上一段开始到位置 i 之间，任何方案都必须至少增加一个极大值位置。贪心地选择其中右端点最大的区间作为这个极大值位置，并令新的最大值为 R。
- 时间复杂度 $O(n)$。

K. Keep Your Distance

题目大意

给定数轴上的 n 个坐标 $a_1 \le a_2 \le \cdots \le a_n$，还给定 $l_2, l_3, \ldots, l_n$。
要为每个位置选择一个整数坐标 b_i，要求 $|b_i - b_{i-1}| \ge l_i$。问 $\max\limits_{1 \le i \le n} |a_i - b_i|$ 的最小值。

$2 \le n \le 10^5$，$\sum n \le 10^5$，$1 \le a_i, l_i \le 10^9$。

- *Standard Solution: 1.04 kB (Cpp)*

- 二分答案 d。问题转化为判定是否存在 $\{b\}$，使得 $b_i \in [a_i - d, a_i + d]$ 且 $|b_i - b_{i-1}| \geq l_i$。
- 令 $L_i = a_i - d$，$R_i = a_i + d$。从左到右的顺序考虑每个位置，设 $S_i \subseteq [L_i, R_i]$ 为满足前 i 个位置所有约束的合法取值 b_i 的集合，初始有 $S_1 = [L_1, R_1]$。
- 令 $mn_{i-1} = \min S_{i-1}$ 与 $mx_{i-1} = \max S_{i-1}$。对于位置 i 的任意候选坐标 $x \in [L_i, R_i]$：
  - x 能由某个位于其左侧的 $y \in S_{i-1}$ 转移而来（即 $y \leq x - l_i$）当且仅当 $mn_{i-1} \leq x - l_i$；
  - x 能由某个位于其右侧的 $y \in S_{i-1}$ 转移而来（即 $y \geq x + l_i$）当且仅当 $mx_{i-1} \geq x + l_i$。
- 综上，位置 i 的可行集 S_i 恒等于区间 $[L_i, R_i]$ 与 $(-\infty, mx_{i-1} - l_i] \cup [mn_{i-1} + l_i, +\infty)$ 的交集。我们只需在从左到右扫描的过程中 $O(1)$ 维护 mn_i 与 mx_i。若在某一步交集为空则当前二分的 d 不可行。
- 时间复杂度 $O(n \log w)$，其中 w 为值域。
- Bonus：本题可以做到 $O(n)$。

C. Counting Arrays

题目大意

给定 n, m，求满足 $a_1 = 0$、$a_{i+1} - a_i \in \{0, 1\}$ 的整数序列 $\{a\}$ 的数量。
另有 m 个前缀和限制：$\sum_{i=1}^{p_j} a_i = v_j$，以二元组 (p_j, v_j) 形式给出。
输出同时满足所有限制的序列数量，结果模 998244353。
$1 \le n \le 100$，$0 \le m \le n$，$0 \le v_j \le 10^9$，$1 \le T \le 10^4$。

- *Standard Solution: 1.68 kB (Cpp)*

- 我们可以按约束位置把整个数组分成几段，一段一段来考虑。
- 对于相邻两个约束位置 L 和 R（设距离为 $\ell = R - L$），假设左端点的值为 a_L。如果不做任何加 1 操作，这一段里所有数都是 a_L，区间和的增量就是 $\ell \cdot a_L$。如果在某一步让数组的值加 1，这个加 1 会一直影响到位置 R。具体来说，在这 ℓ 步中每加一次 1，就能对区间和产生一个介于 1 到 ℓ 之间的贡献，并且每步的贡献互不相同。
- 因此，如果在这一段内我们把数值一共加了 c 次 1（即 $a_R = a_L + c$），并且需要让区间和多出 w（也就是实际需要的区间和增量减去 $\ell \cdot a_L$），问题就直接等价于：
  - 从 1 到 ℓ 这 ℓ 个互不相同的整数中，选出 c 个数，使其和恰好为 w。
- 只要位置 R 的前缀和满足要求，前面具体是怎么加 1 的就不用管了，接下来的状态只和 a_R 的值有关。

- 由于数据范围不大，我们可以预先用 01 背包把“从 1 到 n 中选 c 个数和为 w”的方案数全算出来，这部分的时间复杂度为 $O(n^4)$。
- 处理每组测试数据时，只需要从左到右做简单的 DP。维护到达每个约束位置时，该位置可能的 a_R 值的方案数。转移时枚举上一段结束时的值 a_L 以及加 1 的次数 c，缺的区间和 w 就唯一确定了，直接去预处理的表里查出方案数即可 $O(1)$ 转移。处理完最后一个约束后，后面没有约束的元素每个都可以随便选（加 0 还是加 1），所以把所有方案数加起来乘上 2 的剩余长度次方就是最终答案。
- 单组数据的 DP 耗时 $O(n^2)$，总时间复杂度 $O(n^4 + Tn^2)$。

I. Invisible Edges

题目大意

这是一道通信题。
第一次运行菜猫得到一片森林的所有边，需要给每个顶点 u 赋一个整数 $a_u \in [1, 2n]$。
随后顶点被任意重新编号，赋值随顶点保留。
第二次运行菜狗只得到 n 和新编号下各顶点的数值，需要恢复新编号下的所有边。
$1 \leq n \leq 2 \cdot 10^5$，$\sum n \leq 2 \cdot 10^5$。

- *Standard Solution: 1.44 kB (Cpp)*

- 对森林做 DFS，依次遍历每个连通分量，首次进入顶点 u 时在序列末尾写下一个左括号，离开 u 时在序列末尾写下一个右括号。至此整片森林和一串长度为 $2n$ 的括号序列 S 相对应。
- 在菜猫的回合，令每个顶点 u 的权值 $a_u \in [1, 2n]$ 等于首次进入该顶点时所写下的左括号在序列 S 中的位置下标。这样赋予的 n 个顶点权值两两不同。在菜狗的回合，无论原图的顶点编号经过怎样的打乱，收到的序列 $\{b\}$ 依然两两不同，且集合 $\{b_1, b_2, \ldots, b_n\}$ 恰好指出了 S 中所有左括号的位置，即位置 b_i 唯一对应重标号后的顶点 i。
- 因此，菜狗只需按 $t = 1, 2, \ldots, 2n$ 的顺序从左到右扫描并维护一个顶点栈，即可完整还原 DFS 的过程：若位置 t 等于某个顶点的权值 b_i（对应左括号），则当栈非空时，此时的栈顶顶点正是顶点 i 在 DFS 树中的父结点，在栈顶顶点与 i 之间连一条边后将 i 压入栈中；若位置 t 未在 b 中出现（对应右括号），则直接弹出栈顶顶点。遍历结束后即可还原森林的所有边。
- 两次运行的时间复杂度均为 $O(n)$。

H. Hidden Order

题目大意

有一个长度为 n 的隐藏数组 a，每个元素都是 0、1 或 2。
每次选择两个下标 i, j，询问 $Q(i,j) = [a_i > a_j]$.
请在至多 $2n-3$ 次询问内，输出下标的一个排列 p，使得 $a_{p_1} \leq a_{p_2} \leq \cdots \leq a_{p_n}$.
交互器是自适应的。$2 \leq \sum n \leq 500$。

- *Standard Solution: 899 bytes (Cpp)*

- 本题存在多种解法，这里给出较为简单的两种解法。
- 定义一遍操作如下：
  - 用一个初始为空的双端队列，按给定顺序处理下标。对新下标 x，若队列为空则直接入队；
  - 否则设队尾为 y，询问 $Q(y, x)$。回答 1 将 x 放队首，回答 0 放队尾。
- 注意到，首次遇到全局最大值 M 后，此后等于 M 的元素放入队尾，小于 M 的元素放入队首。因此，一遍操作会将所有最大值集中到末尾。
- 对 $1, 2, \ldots, n$ 做一遍，得到至多含两种值的前段 B，以及仅由最大值 M 组成的后缀。取走末尾一个 M，对剩余序列再做一遍。此时 B 被排好，其后的 M 依次放入队尾。最后追加取走的下标，整个序列有序。
- 时间复杂度 $O(n)$，恰好用了 $(n-1)+(n-2)=2n-3$ 次询问。

- 另一种解法如下。
- 从左到右询问 $Q(i, i+1)$；若全为 0，原序列有序，直接输出，共用 $n-1$ 次询问。
- 否则，在第一次得到 1 时停止。设此时问到 $(r, r+1)$，已用 r 次，且 $a_1 \le \cdots \le a_r > a_{r+1}$。记 $p = r+1$，由 $a_r > a_p$ 知 $a_p \in \{0, 1\}$。
- 令 $L = \{i : a_i \le a_p\}$，$R = \{i : a_i > a_p\}$。两个集合均至多含两种值，且 L 中的值都小于 R 中的值。
- 用两个双端队列充当 L, R。先对 $i = 1, \ldots, r-1$ 依次询问 $Q(i, p)$ 分组，并追加至对应队列队尾，恰用 $r-1$ 次。再将 p 追加到 L，r 追加到 R。此时两队列均有序且非空。
- 对 $i = r+2, \ldots, n$，先问 $Q(i, p)$ 分组，再与所在队列的队尾比较，按上一页规则插入。每个元素用两次询问，两队列始终有序，最终输出 L 再输出 R。
- 时间复杂度 $O(n)$，恰好用了 $r + (r-1) + 2(n-r-1) = 2n-3$ 次询问。

A. Average of Averages

题目大意

给定序列 $a_i = i$ $(1 \le i \le n)$，在线段树上维护区间平均值。
区间 $[l, r]$ 按 $m = \lfloor (l + r)/2 \rfloor$ 分成 $[l, m]$ 和 $[m + 1, r]$。叶子存 a_i，父节点错误地存两个子节点值的算术平均。查询 $[L, R]$ 时，完全覆盖的节点直接返回存储值；若只涉及一侧，则递归查询该侧；若涉及两侧，则返回两侧查询结果的算术平均。
求有多少个区间 $[L, R]$ 的查询结果恰好等于真实平均值 $(L + R)/2$。
所有运算均为精确有理数运算。$1 \le n \le 10^6$，$\sum n \le 2 \cdot 10^6$。

- *Standard Solution: 1.01 kB (Cpp)*

- 显然，符合条件的区间满足 $\mathrm{Query}(1, n, L, R) = \dfrac{L+R}{2}$.
- 考虑分治，在每个线段树节点 $[l, r]$ 处统计内部符合条件的区间 $[L, R]$，记 $m = \lfloor \frac{l+r}{2} \rfloor$。在当前节点仅需考虑跨中点的区间。对于跨中点的区间 $[L, R]$，它由左子区间的一个后缀 $[L, m]$，以及右子区间的一个前缀 $[m+1, R]$ 组成。
- 设 $P_{l,r}(i) = \mathrm{Query}(l, r, l, i)$、$S_{l,r}(i) = \mathrm{Query}(l, r, i, r)$，分别表示线段树节点的前缀、后缀查询结果。记节点存储值为 $v_{l,r}$，则有

$$P_{l,r}(i) = \begin{cases} P_{l,m}(i), & l \le i \le m, \\ \dfrac{v_{l,m} + P_{m+1,r}(i)}{2}, & m < i \le r, \end{cases}$$

$$S_{l,r}(i) = \begin{cases} \dfrac{S_{l,m}(i) + v_{m+1,r}}{2}, & l \le i \le m, \\ S_{m+1,r}(i), & m < i \le r. \end{cases}$$

- 即统计符合以下条件的二元组 (L, R) $(l \le L \le m < R \le r)$：

$$\frac{S_{l,m}(L) + P_{m+1,r}(R)}{2} = \frac{L+R}{2} \iff S_{l,m}(L) - L = R - P_{m+1,r}(R).$$

- 注意到等式左侧只与左端点 L 有关，右侧只与右端点 R 有关。因此，可以先枚举所有右端点 $R \in [m+1, r]$，用哈希表记录 $R - P_{m+1,r}(R)$ 的出现次数；再枚举所有左端点 $L \in [l, m]$，查询 $S_{l,m}(L) - L$ 的出现次数并累加。每个节点的合并与统计需要期望 $O(r-l+1)$ 时间，逐层处理的总时间为期望 $O(n \log n)$。使用 map 实现则时间复杂度为 $O(n \log^2 n)$。
- 注意到相同长度的节点本质等价，可以按照节点长度记忆化 $S_{l,r}, P_{l,r}$ 和节点答案，期望时间复杂度降低到 $O(n)$。若使用 map 实现，则时间复杂度为 $O(n \log n)$。
- 一些基于 sort 实现的 $O(n \log^2 n)$ 方法常数极小，也可能通过本题。
- 注意到存在除法运算，将序列乘以 2 的极大幂即可完全使用整数实现本题。

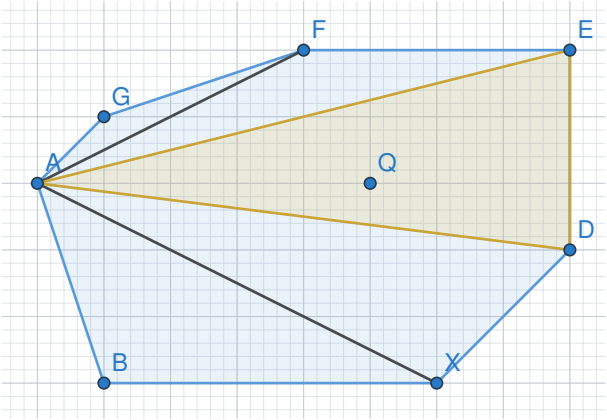
E. Enclose the Point

题目大意

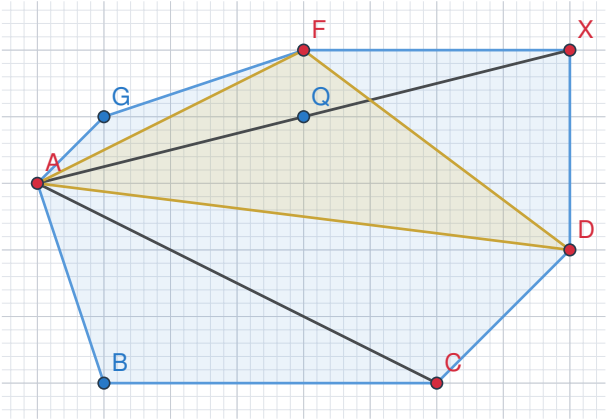
平面上给定 n 个互不相同的点，以及 q 次询问，每次给定一个点 Q。
从给定点中选出若干个作为顶点，构造一个严格包含 Q 的简单多边形，并使顶点数最少。
输出最少顶点数及按顺时针或逆时针排列的顶点编号。若无解，输出 -1。
$3 \le n \le 10^5$，$\sum n \le 10^5$，$\sum q \le 10^5$，所有坐标均为绝对值不超过 10^9 的整数。

- *Standard Solution: 3.26 kB (Cpp)*

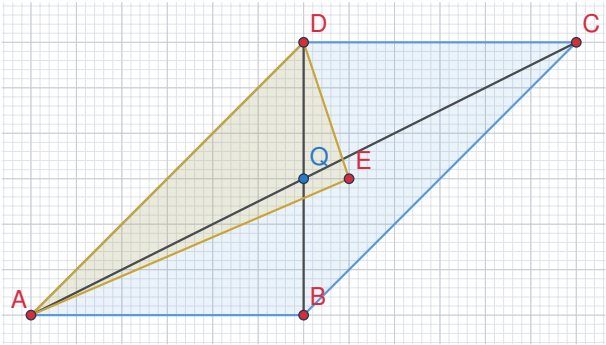
- 首先求给定点集 P 的凸包，如果询问点 Q 严格在凸包内就有解，否则一定无解。
- 对于有解的情况，选取任意一个凸包顶点（例如左下角那个点，记为 A）对凸包做三角剖分，如果 Q 严格落在某一个三角形中，由于多边形边数至少为 3，则这个三角形就是所求的多边形。



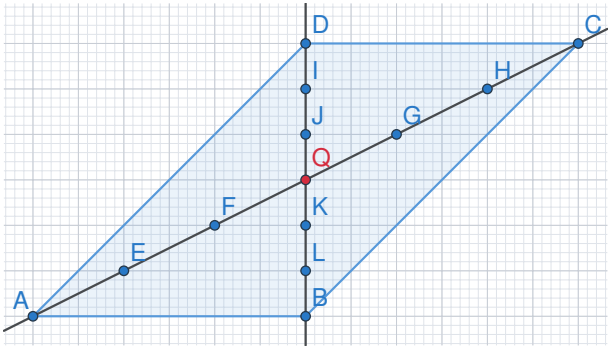
- 如果它在某个对角线上（记对角线两个端点中 A 以外另一端点记为 X），则从凸包顶点中补选点，保证直线 AX 的两侧各至少补选一个点。包括 A 和 X 在内凑够五个，就一定能在这五个点里找到三个点，使得 Q 严格落在以这三个点为顶点的三角形内。
- 注意不能通过简单的换三角剖分的顶点来找三角形。



- 凑不够五个的情况是凸包只有四个点的情况（因为三个点的情况没有对角线）。
- 这种情况下如果 Q 点落在一条对角线上而没有落在另一条对角线上，则显然有三个点的解。
- 对于 Q 点落在两条对角线交点上的情况，考虑点集 P 是否存在在两条对角线以外的点，如果存在则这五个点一定存在三个点构成答案。



- 如果不存在这样的第五个点，则所有点都落在两条对角线所在的直线 l_1 和 l_2 上。任取三个点，至少有两个点落在同一条直线上，第三个点位于这条直线上或其中一侧，因此三角形不可能严格包含交点 Q。此时答案取整个四边形凸包即可。



- 具体实现时，先求出点集 P 的凸包。如果凸包只有四个顶点，再扫描点集，判断是否存在位于两条对角线所在直线之外的点，并记录这样的点。每次询问通过二分凸包，判断 Q 是否严格位于凸包内部，并确定它所在的三角形或对角线。
- 如果已经找到五个候选点，并根据前面的结论确定其中存在三个点构成答案，则直接在这五个点上枚举三角形，判断是否严格包含 Q。
- 时间复杂度 $O(n \log n + q \log n)$。

G. Graph Reconstruction

题目大意

给定一个 $n \times n$ 矩阵 A。当 $i = j$、$i = 1$ 或 $j = 1$ 时，$a_{ij} = -1$。其余位置为非负整数。
构造一个顶点编号为 $1, 2, \ldots, n$ 的无向简单图，使得对任意不同的 $i, j > 1$，删除这两个顶点及其关联边后，恰好剩下 a_{ij} 条边。
每对顶点的删除均独立地从原图开始。保证有解，输出任意合法图的邻接矩阵。
$4 \leq n \leq 10^3$，$\sum n^2 \leq 1.2 \cdot 10^6$。

- *Standard Solution: 2.61 kB (Cpp)*

- 令 $e_{i,j} = 0/1$ 表示 G 的邻接矩阵，H_i 表示 G 删去点 i 后的图，$\deg_{H_i}(j)$ 表示 H_i 中点 j 的度数，$E(H_i)$ 表示图 H_i 的边数。另外，对于题目中给出的矩阵 A，令 $a_{i,j}$ 表示其元素，r_i 表示第 i 行求和的结果（不包含 -1）。
- 首先，考虑描述 $e_{i,j}$。

$$\begin{aligned} e_{i,j} &= \deg_G(j) - \deg_{H_i}(j) \\ &= (E(G) - E(H_j)) - (E(H_i) - a_{i,j}) \\ &= a_{i,j} + E(G) - E(H_i) - E(H_j) \end{aligned} \tag{1}$$

- 这启发我们继续描述 $E(G)$ 和 $E(H_i)$，其中的关键是握手定理。

- 对 $i \geq 2$ 的 H_i 列握手定理。

$$\begin{aligned} 2E(H_i) &= \deg_{H_i}(1) + \sum_{j\geq 2,\ j\neq i} \deg_{H_i}(j) \\ &= \deg_{H_i}(1) + \sum_{j\geq 2,\ j\neq i} (E(H_i) - a_{i,j}) \\ &= \deg_{H_i}(1) + (n-2)E(H_i) - r_i \end{aligned}$$

- 当 $n > 4$ 时，整理得

$$E(H_i) = \frac{r_i - \deg_{H_i}(1)}{n-4} \qquad (2)$$

- 在原图 G 上列握手定理。

$$\begin{aligned} 2E(G) &= \deg_G(1) + \sum_{i=2}^{n} \deg_G(i) \\ &= \deg_G(1) + \sum_{i=2}^{n} (E(G) - E(H_i)) \end{aligned}$$

- 于是

$$E(G) = \frac{\sum_{i=2}^{n} E(H_i) - \deg_G(1)}{n-3} \qquad (3)$$

- 对于 $n \geq 6$，枚举 $\deg_G(1) = 0, 1, \ldots, n-1$。注意到 $\deg_{H_i}(1)$ 只有两种取值且相差 1，分别代入 (2) 式，由于分母 $n-4 \geq 2$，因此至多一种取值能使 $E(H_i)$ 为非负整数。
- 确定了每个 $E(H_i)$，就能用 (3) 式求出 $E(G)$，再用 (1) 式求得所有 $e_{i,j}$。最后在构造出的图上重新统计边数和各点度数，检查得到的 $a_{i,j}$ 是否和题目给出的一致。
- 枚举 $\deg_G(1)$ 并逐行算 $E(H_i)$ 是 $O(n^2)$ 的。把图完整建出来 check 每次也需 $O(n^2)$，但因为当 $i = 2$ 时看 (2) 式

$$\deg_G(1) = r_2 + e_{1,2} - (n-4)E(H_2)$$

固定 $e_{1,2}$ 后，可行的 $\deg_G(1)$ 之间至少相差 $n-4$。由于 $n-1 < 3(n-4)$，区间 $[0, n-1]$ 中至多有三个这样的值。$e_{1,2}$ 只有两种取值，因此至多只会进行 6 次 check。

- 当 $n \leq 5$ 时是简单的，暴力算就行。总时间复杂度 $O(n^2)$。

F. First Word

题目大意

给定一个长为 n 的小写字符串 S。选出 k 个本质不同的非空子串作为词典中的单词。
读者可以指定一个 $\{\mathtt{a},\mathtt{b},\mathtt{c},\ldots,\mathtt{z}\} \to \{\mathtt{a},\mathtt{b},\mathtt{c},\ldots,\mathtt{z}\}$ 的置换，并逐个替换词典中的单词。
要求无论指定什么样的置换，字典序最小的始终是同一个单词。
对每个 $k = 1, 2, \ldots, n$，求满足要求的词典数量，结果模 998244353。$1 \le n \le 10^5$。

- *Standard Solution: 6.93 kB (Cpp)*

- 要求等价于某个被选单词是其余单词的前缀。先反转 S 及所有被选单词，将前缀转为后缀，再建出后缀链接树。
- 对状态 u，记 $d_u = \mathsf{len}_u - \mathsf{len}_{\mathsf{link}_u}$，即 u 表示的不同子串数。再记 $D_u = \sum_{v \in \mathsf{subtree}(u)} d_v$，即 u 的子树中所有不同子串的数量。
- 合法选择中，设最短被选串属于状态 u，那么其他被选状态都在 u 的子树内。反过来，从这个子树中选 k 个串，且至少选一个属于 u 的串，也一定合法。
- 枚举这个唯一的状态 u，得到

$$\mathsf{ans}_k = \sum_u \left(\binom{D_u}{k} - \binom{D_u - d_u}{k} \right)$$

- 记 $X_u = D_u$，$Y_u = D_u - d_u$。展开组合数得到

$$\mathsf{ans}_k = \frac{1}{k!} \underbrace{\sum_u ((X_u)_k - (Y_u)_k)}_{Q_k}$$

- 这里用 $(x)_k = x(x-1)\cdots(x-k+1)$ 表示下降阶乘。下降阶乘和普通幂可以通过斯特林数相互转换，因此分两步走。
- 第一步，对所有 i 算出幂和 $P_i = \sum_u (X_u^i - Y_u^i)$。
- 第二步，已知所有 P_i，对所有 k 算出 $Q_k = \sum_{i=0}^{k} s(k,i) P_i$。这里 $s(k,i)$ 表示有符号第一类斯特林数。

- 先来做第一步，即求 $P_i = \sum_u (X_u^i - Y_u^i)$。
- 构造 $A(x) = \prod_u (1 - X_u x)$，$B(x) = \prod_u (1 - Y_u x)$。
- 由泰勒公式，$\log(1 - tx) = -\sum_{i \geq 1} \frac{t^i}{i} x^i$，也就是说

$$\log A(x) - \log B(x) = -\sum_{i \geq 1} \frac{P_i}{i} x^i$$

- 因此构造出 A, B 后，只需进行两次多项式 $\log$ 即可得到全部 $P_1, \ldots, P_n$。而 A, B 都是若干一次多项式的乘积，可用分治配合 NTT 在 $O(n \log^2 n)$ 内求解。

- 下面做第二步，即求所有 $Q_k = \sum_{i=0}^{k} s(k,i)P_i$。
- 目标等价于将多项式 $(x)_k = \sum_{i=0}^{k} s(k,i)x^i$ 的各项系数与序列 P 的对应项相乘并求和。
- 考虑分治，当前区间 $[l,r]$，中点 $m = \lfloor (l+r)/2 \rfloor$，左半区间长度 $len_L = m - l + 1$。对右半区间的第 s 个位置（即区间第 $len_L + s$ 个元素），其连续乘积多项式可拆分为

$$\prod_{t=l}^{m+s}(x-t) = \underbrace{\left(\prod_{t=l}^{m}(x-t)\right)}_{G(x)} \times \underbrace{\left(\prod_{t=m+1}^{m+s}(x-t)\right)}_{H_s(x)}$$

- 设 $G(x) = \sum_{i=0}^{len_L} g_i x^i$，$H_s(x) = \sum_{j=0}^{s} h_{s,j} x^j$，展开相乘后 x^{i+j} 的系数为 $\sum g_i h_{s,j}$。

- 将该系数与当前 P 的对应项相乘并求和，得到右半区间的求和目标

$$\sum_{i=0}^{len_L} \sum_{j=0}^{s} g_i h_{s,j} P_{i+j}$$

- 交换求和号

$$\sum_{j=0}^{s} h_{s,j} \left(\sum_{i=0}^{len_L} g_i P_{i+j} \right)$$

- 定义新序列 $P'_j = \sum_{i=0}^{len_L} g_i P_{i+j}$，上式化简为 $\sum_{j=0}^{s} h_{s,j} P'_j$。
- 此时右半区间的计算结构与原问题完全一致。只需将 P' 作为右半区间的新输入序列，即可完全独立地递归计算。
- 其中 P' 是一个差卷积，将 P 翻转后做一次 NTT 即可算出。分治时每一层是 $O(n \log n)$ 的，因此整个分治为 $O(n \log^2 n)$。本题总复杂度也为 $O(n \log^2 n)$。

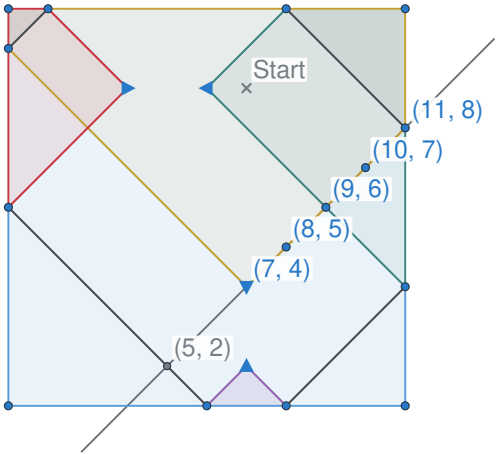
B. Bouncing Particles

题目大意

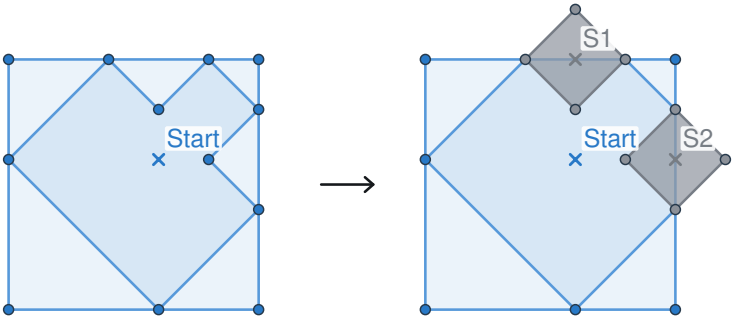
在 $n \times n$ 网格中，初始有四个粒子位于 (x_0, y_0)，方向分别为右上、左上、右下、左下。
每一步，粒子先分别检查水平、竖直方向，若下一格越界，就反转对应方向；随后消失，并在水平、竖直方向的相邻格各产生一个继承其两个方向的子粒子。
所有粒子同时运动，允许重叠且互不影响。
每次询问给定 A, B, C, t，求时刻 t 时，直线 $Ax + By + C = 0$ 上有粒子的网格的数量。
$2 \le n \le 10^9$，$0 \le t \le 10^9$，$|A|, |B|, |C| \le 10^9$，$(A, B) \neq (0, 0)$，$\sum q \le 10^5$。

- *Standard Solution: 4.84 kB (Cpp)*

- 处理这题的第一步是消除“反弹”这一设定。
- 对于在 t 时刻初始释放粒子的操作，在未来某时刻 $t + t_0$ 要想判断某一点有没有由它生成的粒子，可以判断它是否落在一个倾斜的正方形上。
- 粒子碰到墙面后会反弹，观察可以发现这等效于在碰撞时刻、碰撞位置释放四个方向的粒子。



- 同一时刻至多有八个中心点释放的总共十二个正方形需要考察。它们来自起点、与起点横坐标或纵坐标相同的边界点，以及边界顶点这九个候选点；其余正方形边上的所有点都完全在范围外。
- 这些正方形中的很多条边都在范围以外，所以实际需要考察的边不会超过 20 条。



- 每个要考察的正方形的边，都可以写成 $x+y+C=0$ 或 $-x+y+C=0$ 的形式，并限制 $x \in [l,r]$。于是问题就转化成了将询问直线与这些直线联立求解的问题：

$$\begin{cases} A_1 x + y + C_1 = 0, \\ A_2 x + B_2 y + C_2 = 0, \\ x \in [l,r]. \end{cases}$$

- 求解完成后过滤掉重复的解，解的个数即为答案。
- 时间复杂度 $O(q)$，常数较大但是依然能轻松通过。实现过程需要多注意细节，用代码常数的增大换取代码难度的降低。

J. Joint Relocation

题目大意

数轴上有 n 名员工，分属 k 个非空团队，给定每个团队所有员工的居住坐标。
将团队分成两组，并为每组选择一个任意实数坐标作为集合点。允许一组为空，同一团队的所有员工必须前往同一个集合点。求所有员工从居住位置到各自集合点距离之和的最小值。
$1 \le k \le n \le 3 \cdot 10^5$，$\sum n \le 3 \cdot 10^5$，居住坐标为 $[1, 10^7]$ 内的整数。

- *Standard Solution: 3.30 kB (Cpp)*

- 显然，集合点只需要选在输入坐标上。固定队伍划分后，每个非空集合的最优集合点都可以取为其员工坐标的中位数。将不同的输入坐标排序为 $v_0 < v_1 < \cdots < v_{D-1}$。记第 i 队的员工坐标多重集合为 S_i，定义

$$F_i(x) = \sum_{p \in S_i} |p - x|.$$

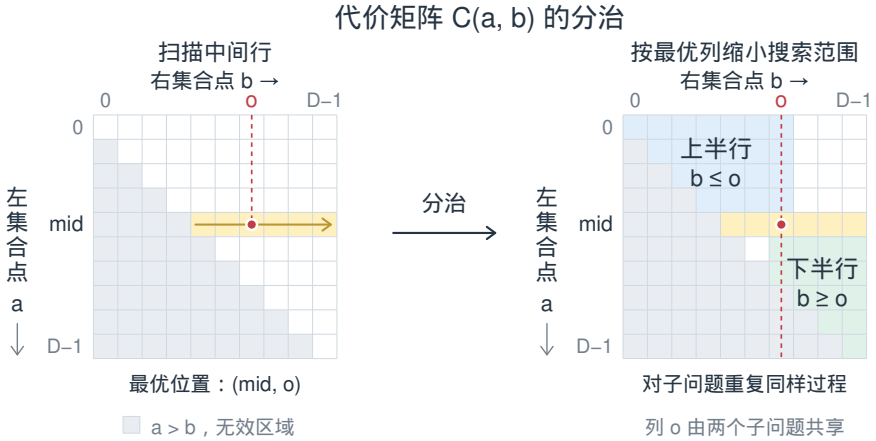
- 固定两个集合点 $v_a \le v_b$ 后，每队独立选择代价较小的集合点，因此总代价为

$$C(a, b) = \sum \min\big(F_i(v_a), F_i(v_b)\big).$$

- 答案即为所有 $0 \le a \le b < D$ 的 $C(a, b)$ 的最小值。对于单个队伍，将坐标排序、维护折点和前缀和后，可以二分在 $O(\log m_i)$ 时间内求出 $F_i(x)$，其中 m_i 为该队人数。
- 考虑枚举左集合点，求出对应的最优右集合点。由于每个 F_i 在中位数左侧不增、右侧不减，按中位数的位置分类，可以得到：对于 $a_1 \le a_2 \le b_1 \le b_2$，有

$$C(a_1, b_1) + C(a_2, b_2) \le C(a_1, b_2) + C(a_2, b_1).$$

- 因此最优右集合点具有决策单调性，即左集合点向右移动时，最左的最优右集合点不会向左移动。



- 可以据此分治：在左集合点的区间中取中点，枚举其候选右集合点，求出最优位置 o。左半区间的最优右集合点不会超过 o，右半区间的最优右集合点不会小于 o，据此缩小候选范围并递归。
- 这样每层只会产生 $O(D)$ 个代价查询。但单独计算每个查询仍然太慢，考虑把同一层的查询放在一起处理。
- 按分治节点从左到右、节点内右集合点从小到大的顺序排列查询，将实际集合点坐标记为 (x_j, y_j)。由于各节点的候选区间有序，只共享边界，x_j 和 y_j 都单调不减。
- 对于一个队伍，随着两个集合点右移，它只会从选择右集合点变成选择左集合点。
- 具体地，记它的下中位数为 μ_i：当 $y_j < \mu_i$ 时选择右集合点，否则比较两个集合点的代价，相等时选择左集合点。
- 这是因为 $y_j \geq \mu_i$ 后，$F_i(y_j)$ 不减；而在 $x_j < \mu_i$ 时，$F_i(x_j)$ 不增；一旦 $x_j \geq \mu_i$，左集合点就始终不差于右集合点。因此，可以二分出这一队第一次选择左集合点的位置 t_i。

- 接下来计算所有队伍的贡献。先假设所有队伍都去右集合点，预处理

$$H(y) = \sum_i F_i(y) = \sum_{\text{所有员工}p} |p - y|.$$

- 全体坐标排序后，用人数和坐标和即可求出所有 $H(v_b)$。
- 对于第 i 队，只需要在查询后缀 $j \geq t_i$ 加上

$$F_i(x_j) - F_i(y_j) = \sum_{p \in S_i} \big(|x_j - p| - |y_j - p|\big).$$

- 对于一个员工坐标 p，$|x_j - p|$ 在 x_j 到达 p 时，由 $p - x_j$ 变为 $x_j - p$；$-|y_j - p|$ 也只会改变一次表达式。因此，上述修正量可以按 x_j、y_j 和常数项，分别维护三组系数差分。

- 用双指针预处理每个坐标在两个查询序列中的分界位置。对每名员工，在后缀起点加入系数，之后若 x_j 或 y_j 到达 p，再修改相应系数即可，每名员工只产生 $O(1)$ 次修改。
- 最后扫一遍差分数组，就能得到该层所有查询的代价，并确定各分治节点的最优决策。
- 一层中，处理差分和查询需要 $O(n+D)$，二分所有队伍的切换位置需要

$$O\left(\log D \sum_i \log(m_i+1)\right).$$

- 由于 $\sum_i m_i = n$，且 $\log_2(m_i+1) \le m_i$，每层复杂度为 $O(n \log D)$。
- 共有 $O(\log D)$ 层，加上排序预处理，总时间复杂度为 $O(n \log^2 n)$。
- Bonus：本题可以做到 $O(n \log n)$。

L. Long Slides

题目大意

平面网格图上给定一个起点和一个终点，以及一个长为 n 的指令串 p。
你需要预先放置不超过 n 个障碍物，使位于起点的机器人按顺序执行完 p 后停止在终点。
每条 `L`，`R`，`U`，`D` 指令让机器人沿指定方向一直移动，停在第一堵障碍物的前一格。
机器人可以原地不动，但不能沿着没有障碍物的射线无限移动。
输出一组合法的障碍物放置方案，或判定无解。$1 \leq \sum |p| \leq 10^5$。

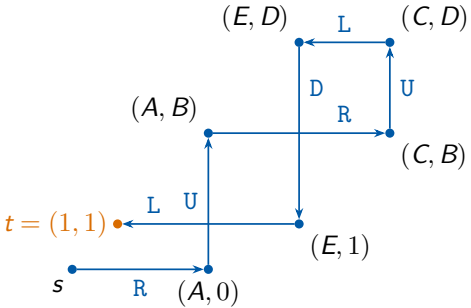
- *Standard Solution: 7.67 kB (Cpp)*

机器人在移动过程中至多撞到 n 个障碍物。因此我们构造方案时不必受障碍物数量的限制，只要最后模拟一遍机器人的路径，保留实际撞到的障碍物就行。

首先，对于连续的 L/R 段和连续的 U/D 段，我们只保留最后一条指令，设为 q。例如，若 $p =$ LRLRUDDUL，则 $q =$ RUL。这是因为如果我们在每一段最后一条指令反方向的无穷远处放置一个障碍，p 就和 q 等价。之后我们只考虑 q。

先看四个方向都出现的 q，并尝试让每条指令都必须移动。q 横竖交替，每步只改变一个坐标；最后两步则必须依次取终点的两个坐标。因此，长度为 r 的串只留下 $r-2$ 个未知数。

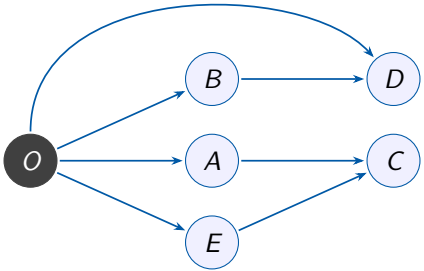
RURULDL　　$s=(0,0),\ t=(1,1)$



确定了这些数，就能在每个停点前方一格放障碍。

为了让这些障碍物不撞上其他中间路段，我们先把不同中间行列彼此拉开，并放到起终坐标范围之外。

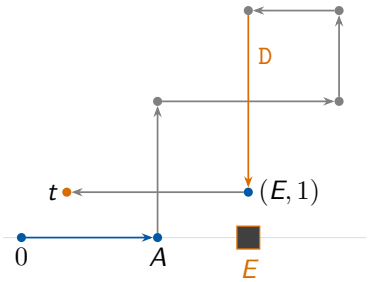
在这个例子中，移动方向要求 $0 < A < C$、$1 < E < C$，以及 $0 < B < D$、$1 < D$。既然中间坐标都在 $[0, 1]$ 之外，0 和 1 对它们的大小关系就相同，可以合成一个标记 O。用一条从小到大的边表示每个不等式，便得到下面的图：



四个方向都出现，所以两条坐标链各有正反向的边，合并首尾后不会成环；两条链只共用 O，因此图无环，可以按拓扑序依次取整数坐标。

一般情形中，O 代表包含起终点四个坐标的整个区间，排在它前后的变量分别放到区间两侧，彼此间隔开（实现时可以取一个固定的间隔，至少间隔两格）。O 本身并不是实际的坐标，放置障碍物时仍使用原来的 s, t。

这样选出的路线方向是对的，但还不能保证路径和障碍物互不干扰，因为后面要用的障碍物可能截断前面的路。具体看下一页的例子。



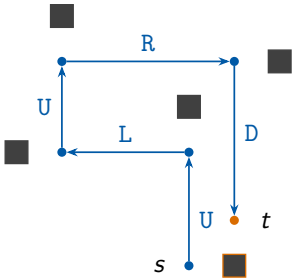
最后一条指令是 L，所以 $E > 1$。倒数第二条指令 D 要停在 $(E, 1)$，就必须在 $(E, 0)$ 放一个障碍。而第一条指令 R 又要经过 $[0, A]$，因此还需要

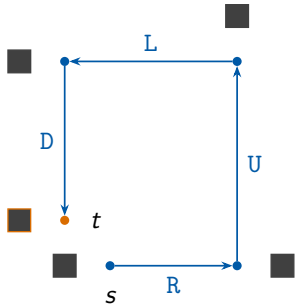
$$A < E$$

原图只知道 $A < C$、$E < C$，并没有确定这个顺序。

前面拉开中间行列，正是为了把这种冲突限制在首末几段。这些首末坐标在端点区间的哪一侧，已经由方向确定。首末同轴时，唯一还未确定的是首段终点 A 与末段起点 E 谁大。因此分别加入 $A < E$、$E < A$，生成两组候选再模拟即可。

不过，要求每一步都移动，仍会漏掉有解的情况。看下面第一幅图，$s = (0, 0)$，RULURD 要到达 $t = (1, 1)$，最后一条指令 D 必须在 $(1, 0)$ 放一个障碍，因而第一条指令 R 只能原地不动。类似地，第二幅图中，$s = (0, 0)$，RULDL 到 $t = (-1, 1)$，也可以先到终点，再让最后一条指令不动。

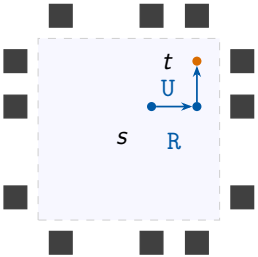




实现时，在起点或终点沿对应方向的下一格补障碍物，把第一条指令 / 最后一条指令从 q 中拿掉，对余串重复前面的构造。

总的来说，尝试全部移动、第一步不动、最后一步不动三种安排，每种再试两种坐标顺序（上上页提到的）。后续称这个构造为“三种拓扑构造”。

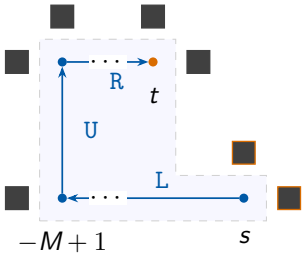
再看某个轴只有一个方向的情况。RURURU 从 $(0,0)$ 到 $(1,1)$ 时，六条指令中至少四条必须不动，不可能给每次转弯都安排一个新坐标。因此，我们还需要允许重复使用同一行、同一列的构造。



把起点围在矩形内，每次指令就只会把机器人送到某条边。因此，最后一次水平移动决定最终横坐标，最后一次竖直移动决定最终纵坐标；反复朝同一边移动会自动产生零步。

也就是说，如果 q 最后两条指令对应的移动方向，恰好是 t 相对于 s 的方位，该构造就能生效（即矩形就能将 s 包住）。

若最后一次水平移动是 R，矩形必须以 t_x 作右边界；目标在起点左侧时，它就包不住 s。如果竖直方向只有 U，可以改用 L 形通道，具体构造如下。（图中 $M = 2 \times 10^9$）



机器人执行最后一个 L 后到达左端。由于 q 横竖交替、没有 D，后面一定有 UR：先上到顶部，再向右到达 t。此后只剩 R/U，都会停在原地。

因此，在矩形失败的前提下，该构造有解恰好要求目标位于起点左上方，且 q 含有 L。这样，凡是某轴缺少一个方向的输入都已处理完（这里演示的是缺少 D 的情况，缺少 U，L，R 同理，对称/旋转一下这个构造即可）。

总结一下。

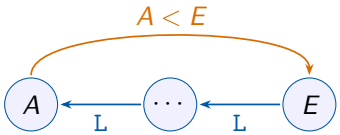
- q 只含 L,R,U,D 中的一个方向 $\longrightarrow$ 特判。
- q 只含 U,D 或 L,R $\longrightarrow$ 特判。
- q 只含 U,L, 或 U,R, 或 D,L, 或 D,R $\longrightarrow$ 矩形构造或无解。
- q 只含 L,R,U,D 中的三个方向 $\longrightarrow$ 矩形，L 形构造或无解。

最后就是 q 含所有四个方向的情况。后面两页会证明，前文提到的矩形构造和三种拓扑构造，能覆盖所有的情况。(也就是说，若两种构造都不行则无解)

这些都是常数次尝试，时间复杂度 $O(n \log n)$。

以下设矩形构造失败，$s=(0,0)$、$t=(x,y)$ 且前两条指令为 RU。其他情况都能通过将盘面旋转/翻转/平移从而转换为当前局面。

先看最后一条指令水平时，前一条竖直指令的障碍物落在起点行，即 $(E,0)$。最后为 R 时 $E<0$，不会挡路；为 L 时，沿用前文得到的 $A<E$。

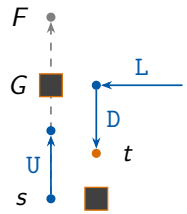


第一条 R 和最后一条 L 给出 $O\to A$、$O\to E$，所以从 E 返回 A 只能沿中间的水平链。中间若还有 R，返回路径就会断开，加入 $A<E$ 仍无环，全部移动的构造便成功。

若没有第二个 R，之后只能向左：$x\geq 0$ 时，障碍物就在第一次 R 必经的路上，无解；$x<0$ 时，令最后一步不动，使 $E=x$，障碍物便在起点左侧。此时至少有两个 L（首末水平，中间有 U,D），去掉最后一条后四个方向仍都出现。新增的障碍物 $(x-1,y)$ 也不会挡住第一步或最后一段竖直移动。

其余冲突中，障碍物的位置由终点固定：最后一步竖直时在 $(x, 0)$；最后一步水平且起终点同行时在 $(x \pm 1, 0)$。由前文的矩形条件，后一种情况以 R 结束时 $x < 0$，以 L 结束时 $x > 0$。

障碍物在起点左侧，全部移动即可；占住起点则无解。在起点右侧时，唯一的 R 必被挡。若还有 R，则采用第一步不动的构造。新增的障碍物 $(1, 0)$ 在目标左侧或与原障碍物重合，不妨碍结尾。



第一步不动的构造还可能遇到新冲突。即最后的 L 停在 $x = 1$ 所需的障碍物，正好位于第一条 U 所在的 $x = 0$ 列。最后若为 U，此时 $y = -1$，障碍物在起点下方；故只剩最后两条为 LD、$t = (1, 1)$ 的情况。

设第一次 U 计划到 $(0, F)$，最后 D 从 $(1, G)$ 出发。后续有 U 就可同理取 $F < G$；否则只能下降，$1 \le G \le F$，障碍物 $(0, G)$ 必在第一次 U 的路径上，无解。□