Problem A. 平方王国

在平方王国中，有 n 名标号从 1 到 n 的居民各自生活在高高的石柱上。第 i 名居民的石柱高度是距离地面向上 $\left(i+\frac{b}{a}\right)^2$ 单位。

由于每个人都生活得如此高，拜访邻居的唯一方式是使用梯子。王国为每对居民都建造了一个梯子。每个梯子的长度正好是连接的两个柱子之间高度之差的绝对值。

总共有 $\frac{n(n-1)}{2}$ 个梯子，管理它们变得困难。你需要找出按长度升序排列的第 k 个梯子的长度。这些信息将帮助王国计划维修、配送和社区活动。

Input

仅有的一行包含四个整数 n ($2 \le n \le 10^{12}$)、k ($1 \le k \le \min\left\{\frac{n(n-1)}{2}, 10^{12}\right\}$)、$a$ ($1 \le a \le 10^6$) 和 b ($0 \le b \le 10^{12}$)。

Output

输出一行包含两个整数 p 和 q，表示按长度升序排列的第 k 个梯子的长度可以表示为 $\frac{p}{q}$，其中 p 和 q 是两个整数 ($p \ge 0$, $q \ge 1$, $\gcd(p,q)=1$)，其中 $\gcd(p,q)$ 是 p 和 q 的最大公约数。

Examples

standard input
3 1 3 1
standard output
11 3

standard input
3 2 3 1
standard output
17 3

standard input
3 3 3 1
standard output
28 3

standard input
1414215 1000000000000 1000000 1000000000000
standard output
4823373069559 1

此页刻意留白

Problem B. 出 Bug 的绘画软件 I

耳廓狐亚砜是一名喜欢在闲暇时间创作数字艺术的码农。在调了一整天的 Bug 后，他已精疲力尽，但当他像往常一样打开绘画软件时，却不曾想软件本身在最近一次更新后也充斥着 Bug，一个主要问题是：软件在他每编辑一个像素时都会明显卡顿。

该绘画软件将画布视作一个有序的图层栈。每个图层是一个 n 行、m 列的像素网格，每个像素拥有一个在 $\{0, 1, \ldots, nm\}$ 范围内的**可见值**，其中 0 表示透明，正数值则为 nm 种不同颜色的标识符。由于图层之间可能存在遮挡，在任意位置上，合成图像的可见值等于所有图层中最顶层的非透明像素的可见值；如果该位置上的所有图层均是透明像素，则可见值为 0。

亚砜已经确切地想好了他要画什么：一个由可见值矩阵 $(p_{i,j})_{n\times m}$ 表示的目标图像。为了尽快画完，他希望最小化由软件 Bug 造成的总“卡顿代价”。从没有任何图层的状态开始，亚砜可以执行以下操作任意次：

- **创建图层**：免费在顶部插入一个新图层，并将其初始化为一个纯色图层（每个像素设为同一种所选颜色 c $(1 \le c \le nm)$），或初始化为空图层（每个像素均为透明）；
- **画笔工具**：花费 a 将任一图层的任一单个像素设置为任一种所选颜色 c $(1 \le c \le nm)$；
- **橡皮工具**：花费 b 将任一图层的任一单个像素变为透明。

请帮助亚砜确定所需的最小总代价，使得经过操作后合成图像与目标图像匹配，即两张图像在对应位置的可见值相等。

Input

输入的第一行包含一个整数 T $(1 \le T \le 10^5)$，表示测试数据的组数。对于每组测试数据：

第一行包含四个整数 n、m $(1 \le n, m \le 500)$、a 和 b $(1 \le a, b \le 10^6)$，分别表示图像的尺寸、使用画笔工具的代价和使用橡皮工具的代价。

接下来 n 行的第 i 行包含 m 个整数 $p_{i,1}, p_{i,2}, \ldots, p_{i,m}$ $(0 \le p_{i,j} \le nm)$，其中 $p_{i,j}$ 表示目标图像在第 i 行、第 j 列上的可见值。

保证所有测试数据中 nm 的和不超过 10^6。

Output

对于每组数据，输出一行包含一个整数，表示最小总代价。

Example

standard input	standard output
3 1 2 3 2 0 1 2 2 1 1 1 0 2 3 3 3 5 3 2 4 4 4 1 4 4 4 2	2 3 11

Note

对于样例的第一组测试数据，亚砜可以先创建颜色为 1 的图层，再花费 $b = 2$ 的代价将第一行、第一列上的像素变为透明。

Problem C. 出 Bug 的绘画软件 II

这是一道通信题。

呃……在挺过了 **Problem B. 出 Bug 的绘画软件 I** 中那噩梦般的卡顿之后，亚砜很快又遇到了一个更严重的 Bug — 他的绘画软件会把任意彩色图像都以黑白格式保存。

这里，彩色图像支持标号为 1 到 m 的 m 种不同颜色，并可看成一个长度为 $3m$、由颜色标号构成的序列，其中每种颜色都**恰好出现三次**。黑白图像仅支持标号为 0 的黑色和标号为 1 的白色，并可看成一个二进制序列。在保存彩色图像时，软件会选择 $S = \{1, 2, \ldots, m\}$ 的一个非空二划分 (S_0, S_1)（即 $S_0 \cup S_1 = S$、$S_0 \cap S_1 = \varnothing$，且 S_0 与 S_1 均非空）并通过将 S_0 中的每种颜色映射为 0、将 S_1 中的每种颜色映射为 1 的方式，将原始序列转换为一个等长的二进制序列。得到的二进制序列将作为保存后的黑白图像。

为了智取这个 Bug，亚砜想要创作 n 张彩色图像，它们分别具有 1 到 m 范围内的整数特征值 $x_1, x_2, \ldots, x_n$（值可能重复）。你的任务是帮助亚砜设计这些彩色图像，使得每张图像的特征在颜色信息丢失后仍然可被识别。更具体地，第 i 张彩色图像 $(1 \le i \le n)$ 必须满足无论何种非空二划分 (S_0, S_1) 被选择，其特征值 x_i 一定能从保存后的黑白图像中恢复出来。

为了检验你设计的正确性，你的程序将在每个测试点上**被运行两次**：第一次用于构造 n 张彩色图像，第二次用于从 n 张黑白图像中恢复原始特征值。第二次运行将会收到 n 个二进制序列，这些序列正是由你第一次运行时输出的 n 个序列经过各自独立的非空二划分映射得到的。除此之外，第二次运行将不会收到来自第一次运行的任何额外信息。如果你对每张黑白图像都能正确地恢复出其对应的特征值，则你的解决方案通过此测试点。

注意，采用的二划分是不可预测的，并且可能**自适应**于你的设计，具有相同特征值的彩色图像可能会采用不同的二划分。

Input

第一行包含三个整数 op $(op \in \{1, 2\})$、n 和 m $(2 \le n, m \le 5 \times 10^5,\ nm \le 10^6)$，分别表示程序的第几次运行、彩色图像的数量和不同颜色的数量。保证 n 和 m 在两次运行中保持一致。

如果 $op = 1$，你的程序必须构造彩色图像。接下来 n 行，其中第 i 行包含一个整数 x_i $(1 \le x_i \le m)$，代表将设计的第 i 张图像的特征值。

如果 $op = 2$，你的程序必须恢复特征值。接下来 n 行，每行描述一张黑白图像，即每行包含 $3m$ 个由空格隔开的、取值 0 或 1 的整数，构成一个从你某张设计的彩色图像得到的二进制序列。这 n 个序列将以**任意顺序**给出，不一定对应第一次运行时特征值给出的顺序。

Output

如果 $op = 1$，输出 n 行，其中第 i 行描述针对输入中第 i 个特征值 x_i 设计的彩色图像。每行包含 $3m$ 个由空格隔开的整数。每行的序列必须是一张合法的彩色图像，即从 1 到 m 的各整数恰好出现三次的序列。

如果 $op = 2$，输出 n 行。对于输入中给定的 n 个二进制序列的每一个，输出一行包含一个整数，表示从该图像中恢复的原始特征值。你输出的顺序必须对应这些二进制序列在输入中的顺序。

Example

standard input	standard output
1 2 3 1 2	1 1 1 2 2 2 3 3 3 1 2 3 1 2 3 1 2 3
2 2 3 1 1 0 1 1 0 1 1 0 0 0 0 1 1 1 0 0 0	2 1

Note

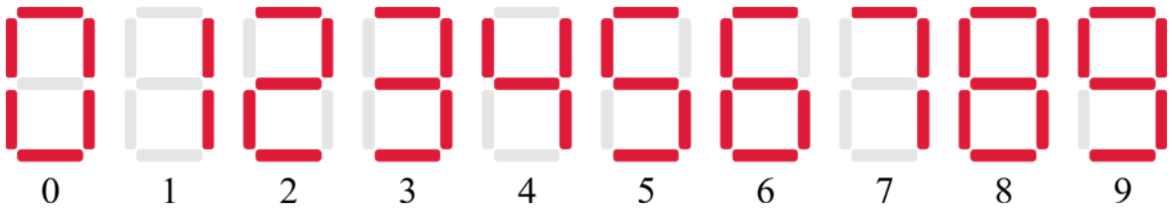
示例展示了某个解决方案在样例数据中的两次运行。

第一次运行采用了一个简单（对一般情况不正确）的策略：特征值 1 被设计成三个相同颜色的连续段，特征值 2 被设计成三种不同颜色的循环结构。

第二次运行中，第一个二进制序列（来自二划分 $(S_0 = \{3\}, S_1 = \{1, 2\})$）可通过辨识其循环结构恢复出特征值 2，第二个二进制序列（来自二划分 $(S_0 = \{1, 3\}, S_1 = \{2\})$）可通过辨识其连续段恢复出特征值 1。由于评测机以交换顺序提供了这两个序列，输出为先 2 再 1。

Problem D. LED 显示器翻新

你有一个能够显示 n 位整数的 LED 显示器，每位数字均采用标准的 7 段显示布局。



开始时，每个数字块中的 7 个数字段可能处于三种状态之一：

- 正常（用 'W' 表示）：表现正确，即根据显示的数字点亮或熄灭；
- 常亮（用 '1' 表示）：无论显示什么数字，始终点亮；
- 常灭（用 '0' 表示）：无论显示什么数字，始终熄灭。

你可以翻新最多 k 个数字段，也就是转换最多 k 个数字段为正常状态。你需要最大化可以正确显示的整数数量，并计算得到最大值的方案数。请注意，你可以翻新开始时就处于正常状态的数字段，每个数字段最多可以被翻新一次，两种翻新方案不同当且仅当存在至少一个数字段在一种方案中被翻新而在另一种方案中未被翻新。

一个整数被正确显示当且仅当：

- 除非整数恰好为 0，否则不能有前导零；
- 在显示器上右对齐：如果整数有 d $(1 \le d \le n)$ 位，则仅使用最右侧的 d 个数字块，最左侧的 $(n-d)$ 个块应留空且不显示任何内容；
- 对于每个使用的 d 个数字块，当正常段显示该数字的正确状态且故障段保持其状态时，所有 7 个数字段都符合目标数字的显示模式。

Input

输入的第一行包含一个整数 T $(1 \le T \le 100)$，表示测试数据的组数。对于每组测试数据：

第一行包含两个整数 n $(1 \le n \le 9)$ 和 k $(0 < k < 7n)$，表示 LED 显示器可以显示的数字位数，以及你可以翻新的数字段数量。

接下来的 7 行每行包含一个恰好 $(5n-1)$ 个字符的字符串，以 ASCII 图像的形式描述 LED 显示器的初始状态。

显示器由 n 个数字块组成，每个占用 4 列，由单列间隙分隔。所有数字段由两个相同的字符 'W'、'0' 或 '1' 表示，所有其他位置用字符 '.' 填充，仅用于格式化。有关详细信息，请参见示例输入和说明。

Output

对于每组测试数据，输出一行包含两个整数，表示通过翻新最多 k 个数字段时可以正确显示的整数的最大数量，以及得到最大值的方案数。

Example

standard input
2 3 2 .00...00...00. 0..0.0..0.0..1 0..0.0..0.0..1 .00...00...00. 0..W.0..0.0..1 0..W.0..0.0..1 .00...00...00. 9 62 .WW...WW...WW...WW...WW...WW...WW...WW...WW. W..W.W..W.W..W.W..W.W..W.W..W.W..W.W..W.W..W W..W.W..W.W..W.W..W.W..W.W..W.W..W.W..W.W..W .WW...WW...WW...WW...WW...WW...WW...WW...WW. W..W.W..W.W..W.W..W.W..W.W..W.W..W.W..W.W..W W..W.W..W.W..W.W..W.W..W.W..W.W..W.W..W.W..W .WW...WW...WW...WW...WW...WW...WW...WW...WW.
standard output
2 23 1000000000 9223372036854775807

Note

在以 ASCII 图像的形式描述 LED 显示器的每个数字块中，数字段的分布如下：

段	部分	位置（行，列）
0	顶部水平段	$(1,2)$, $(1,3)$
1	左上竖直段	$(2,1)$, $(3,1)$
2	右上竖直段	$(2,4)$, $(3,4)$
3	中间水平段	$(4,2)$, $(4,3)$
4	左下竖直段	$(5,1)$, $(6,1)$
5	右下竖直段	$(5,4)$, $(6,4)$
6	底部水平段	$(7,2)$, $(7,3)$

数字 $0,1,2,\ldots,9$ 的显示模式（‘1’ = 点亮，‘0’ = 熄灭）如下：

数字	模式（段 $0,1,2,\ldots,6$）	解释
0	1110111	除了中间段
1	0010010	右上段和右下段
2	1011101	顶部、右上、中间、左下、底部段
3	1011011	顶部、右上、中间、右下、底部段
4	0111010	左上、右上、中间、右下段
5	1101011	顶部、左上、中间、右下、底部段
6	1101111	除了右上段
7	1010010	顶部、右上、右下段
8	1111111	所有段
9	1111011	除了左下段

Problem E. 出奇制胜

你正在玩一款知名卡牌游戏。有 $2n$ 张标号为 1 到 $2n$ 的不同卡牌，它们被均匀随机地洗牌并叠成一副牌堆。你抽取牌堆顶部的 n 张牌作为你的起始手牌，而牌堆中剩余牌的顺序对你是**不可见**的。每一轮，你从手牌中任选一张牌打出，即将其置于牌堆底部然后抽取当前牌堆顶部的牌，你的手牌数量从而保持为 n。

你有 m 个任务需要按顺序完成：第 i 个任务要求你打出标号为 a_i 的牌，并且该任务只有在第 $(i-1)$ 个任务完成后才会被激活（第一个任务除外，它初始就是激活的）。在第 i 个任务被激活之前打出标号为 a_i 的牌**不计入**第 i 个任务的完成进度。你事先知道完整的 $a_1, a_2, \ldots, a_m$（值可能重复）。

假设你采取最优的纯策略，请计算在初始洗牌随机的情形下完成全部 m 个任务所需轮数的最小期望值，对 998 244 353 取模。

形式化地，*纯策略*指的是在每轮选择打出的牌仅取决于可观测的历史，且相同的已知信息（初始手牌、过去每轮分别打出及抽到的牌、任务完成状态）会确定性地导致你打出相同的牌。对于一个纯策略 σ 和一个初始洗牌 π，记 $f_\sigma(\pi)$ 为遵循 σ 时完成全部 m 个任务所需的轮数；需要计算的值等于 $(\min_\sigma E[f_\sigma(\pi)]) \bmod 998\,244\,353$，其中期望 $E[\cdot]$ 是在所有 $(2n)!$ 种可能的初始牌堆排列的均匀分布上计算的。

Input

输入的第一行包含一个整数 T $(1 \le T \le 1\,000)$，表示测试数据的组数。对于每组测试数据：

第一行包含两个整数 n 和 m $(2 \le n \le 5\,000, 1 \le m \le 2 \times 10^5)$。

第二行包含 m 个整数 $a_1, a_2, \ldots, a_m$ $(1 \le a_i \le 2n)$。

保证所有测试数据中 n 的总和不超过 $5\,000$，且所有测试数据中 m 的总和不超过 2×10^5。

Output

对于每组测试数据，由于轮数的最小期望值能被表示为一个最简分数 $\frac{p}{q}$，你只需在一行中输出一个整数 r 满足 $0 \le r < 998\,244\,353$ 和 $r \cdot q \equiv p \pmod{998\,244\,353}$，可以证明这样的 r 存在且唯一。

Example

standard input	standard output
3 2 1 1 3 7 2 5 1 2 6 2 1 1000 2 1116 1116	249561090 299473317 748684517

Note

对于样例的第一组测试数据，一个最优的纯策略是：如果标号为 1 的牌在你的初始手牌里，立即将其打出；否则重复打出任意牌直到你抽到了它并将其在下一轮打出。因此轮数的最小期望值为 $\frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{4} \times 3 = \frac{7}{4}$。

此页刻意留白

Problem F. 友谊地久天长

耳廓狐亚砜建造了一个迷宫，并邀请了他的两位朋友 Alice 和 Bob 来尝试。这个迷宫可以被抽象为一个包含 n 个顶点和 m 条边的简单连通无向图，其中 Alice 和 Bob 从两个不同的指定顶点出发。

在任何玩家移动之前，亚砜必须为每条无向边选择一个方向，从而将迷宫变成一个有向图。然后，为了找到彼此，两名玩家在定向后的迷宫里分回合移动。在每个回合中，他们同时、独立地行动，既不留下标记也不进行交流；而是采用“最无脑”的策略：

- 如果一名玩家当前所在的顶点有一条或多条出边，他们会任意地沿着一条出边移动到相邻的顶点；
- 否则，如果他们当前所在的顶点没有出边，他们会停留在该顶点。

亚砜对自己的迷宫设计技巧感到自豪，他想要为通道定向，使得 Alice 和 Bob 永远不会相遇。给定该无向图以及 Alice 和 Bob 的起始顶点，请帮助亚砜为每条边定向，使得无论 Alice 和 Bob 在每个回合中如何选择，他们都永远不会在任何回合结束时位于同一个顶点。如果这样的定向方案不存在，请报告这一点。

Input

输入的第一行包含一个整数 T ($1 \le T \le 1\,000$)，表示测试数据的组数。对于每组测试数据：

第一行包含四个整数 n ($2 \le n \le 300$)、m ($n-1 \le m \le \frac{n(n-1)}{2}$)、$x$ 和 y ($1 \le x, y \le n$, $x \ne y$)，分别表示迷宫的点数、边数，以及 Alice 和 Bob 的出发顶点。

接下来 m 行每行包含两个整数 u 和 v ($1 \le u, v \le n$)，描述一条连接顶点 u 和 v 的无向边。保证给定图是连通的并且不含自环或重边。

保证有至多 1 组测试数据满足 n 大于 30。

Output

对于每组测试数据：

如果可以定向所有边使得 Alice 和 Bob 永远无法在任一回合结束时占据同个顶点，在第一行中打印“`Yes`”（没有引号）。

接下来 m 行，每行包含两个整数 u' 和 v' ($1 \le u', v' \le n$)，描述一条从 u' 到 v' 的有向边。无序对 (u', v') 必须对应输入里一条存在的无向边。你可以以任意顺序打印这些有向边。

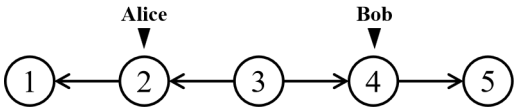
如果不存在合法的定向方案，输出一行包含“`No`”（没有引号）。

Example

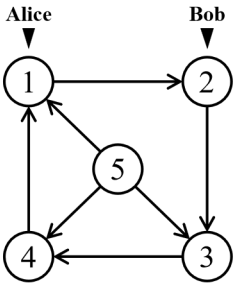
standard input	standard output
3 5 4 2 4 1 2 2 3 3 4 4 5 5 7 1 2 1 2 2 3 3 4 4 5 1 5 3 5 1 4 2 1 1 2 1 2	Yes 2 1 3 2 3 4 4 5 Yes 1 2 2 3 3 4 4 1 5 1 5 3 5 4 No

Note

对于样例的第一组测试数据，一个合法的定向方案如下。在第一个回合，Alice 必须从顶点 2 移动到顶点 1，而 Bob 必须从顶点 4 移动到顶点 5。此回合后，两名玩家都处于没有出边的顶点，并将永远停留在那里，因此永远不会相遇。



对于样例的第二组测试数据，一个合法的定向方案如下。两名玩家都被限制在环 $1 \to 2 \to 3 \to 4 \to 1$ 中。由于他们每回合都在环上前进一格，Alice 将始终落后 Bob 一步，因此永远不会相遇。



对于样例的第三组测试数据，只有一条边连接两个顶点，若将其定向为 $1 \to 2$，则会迫使 Alice 移动到 Bob 所在的顶点，而若将其定向为 $2 \to 1$，则会迫使 Bob 移动到 Alice 所在的顶点。由于总有一名玩家会移动到另一名静止玩家所在的顶点，因此无论如何定向，相遇都是不可避免的。

Problem G. 碰撞伤害

在一个基于二维物理的电子游戏中，两个角色 P 和 Q 在战场上移动。每个角色由一个凸多边形表示，该多边形作为其碰撞箱。当两个碰撞箱重叠时，游戏计算它们交集的面积作为碰撞伤害。

然而，由于竞技场中不可预测的风向，角色 Q 可以被任意平移向量 $\mathbf{t} = (t_x, t_y)$ 位移，但它不能旋转或翻转。你需要计算当 Q 被均匀且随机放置以确保它与 P 实际发生碰撞，即它们的交集具有正面积，的前提下的期望碰撞伤害。

更正式地，定义 $f(\mathbf{t})$ 为表示角色 P 的多边形与沿着 $\mathbf{t}$ 平移后的表示角色 Q 的多边形之间的交集面积。设 $D \subseteq \mathbb{R}^2$ 为所有使得 $f(\mathbf{t}) > 0$ 的平移向量 $\mathbf{t}$ 的集合。可以证明 D 具有正面积，记为 $|D|$。你需要计算 $\frac{1}{|D|} \iint_D f(\mathbf{t})\, d\mathbf{t}$。

Input

输入的第一行包含一个整数 T $(1 \le T \le 300)$，表示测试数据的组数。对于每组测试数据：

- 第一行包含两个整数 n 和 m $(3 \le n, m \le 1\,000)$，分别表示两个凸多边形的顶点数量。
- 接下来 n 行，每行包含两个整数 x 和 y $(-10^4 \le x, y \le 10^4)$，给出表示角色 P 的多边形的顶点坐标 (x, y)。
- 接下来 m 行，每行包含两个整数 x 和 y $(-10^4 \le x, y \le 10^4)$，给出表示角色 Q 的多边形的顶点坐标 (x, y)。
- 保证凸多边形的顶点是按逆时针顺序给出的，并且任意三个顶点不共线。

保证所有测试数据中 n 的总和以及 m 的总和都不超过 $1\,000$。

Output

对于每组测试数据，输出一行包含一个实数，表示期望碰撞伤害。

如果你的答案的绝对误差或相对误差不超过 10^{-6}，则将被视为正确。更正式地，假设你的输出为 a，标准答案为 b，当且仅当 $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$ 时，你的输出会被接受。

Example

standard input	standard output
2 3 3 0 0 1 0 0 1 0 0 1 0 0 1 3 3 0 0 1 0 0 1 0 1 1 0 1 1	0.083333333333 0.125000000000

此页刻意留白

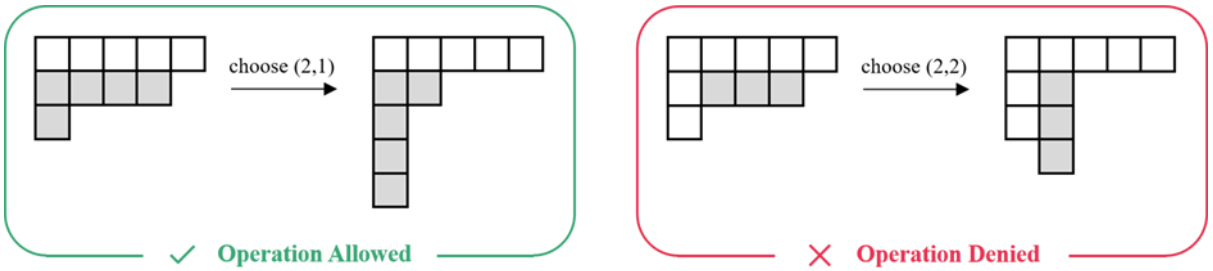
Problem H. 可爱的杨图计数

杨图是一个有限的单元格集合，单元格排列成左对齐的若干行，且行长按顺序非递增。它可以被一个满足 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r \geq 1$ 的无序整数分拆 $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_r)$ 唯一地表示。其中，r 对应杨图的行数，而 $i = 1, 2, \ldots, r$ 时 λ_i 对应第 i 行的单元格数量。

一个杨图 λ 的**共轭**是另一个杨图，它是通过将原图的行和列进行转置得到的，并由共轭分拆 λ^T 表示。具体来说，如果 λ 有 $c = \lambda_1$ 列，那么 $\lambda^T = (\mu_1, \mu_2, \ldots, \mu_c)$，其中 $j = 1, 2, \ldots, c$ 时 $\mu_j = |\{\, i : \lambda_i \geq j \,\}|$。

对于 λ 中位于第 i 行、第 j 列的单元格，我们将锚定于 (i, j) 的**子图**定义为 λ 中所有满足 $x \geq i$ 且 $y \geq j$ 的单元格 (x, y) 的集合。容易看出，当我们将 (i, j) 视为其左上角时，这个集合本身也是一个杨图。

我们将一个杨图 λ 的**可爱度**定义为从 λ 出发，通过任意（可能为零）次执行下述操作所能得到的不同杨图的数量。操作过程如下：选择当前杨图中的任意一个单元格 (i, j)，取出锚定于 (i, j) 的子图，将其替换为它的共轭并锚定在同一位置。当且仅当操作后整体的单元格集合仍然构成一个合法的杨图时，该操作才被允许；否则，该操作将被禁止并撤销，如下图所示。



给定一个非递增的正整数序列 $a_1, a_2, \ldots, a_n$。对于每个 $i = 1, 2, \ldots, n$，你需要计算杨图 $\lambda^{(i)}$ 的可爱度，对 $998\,244\,353$ 取模，其中 $\lambda^{(i)} = (a_1, a_2, \ldots, a_i)$。

Input

第一行包含一个整数 n $(1 \leq n \leq 10^6)$，代表给定序列的长度。

第二行包含 n 个整数 $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq n)$。保证 $a_1 \geq a_2 \geq \cdots \geq a_n$。

Output

输出 n 个整数，由空格隔开，其中第 i 个整数代表杨图 $\lambda^{(i)}$ 的可爱度，对 $998\,244\,353$ 取模。

Example

standard input	standard output
3 3 2 1	2 3 1

Note

对于样例：

- 从 $\lambda^{(1)}$ 出发所能得到的不同杨图有 (3) 和 $(1, 1, 1)$；
- 从 $\lambda^{(2)}$ 出发所能得到的不同杨图有 $(3, 2)$、$(3, 1, 1)$ 和 $(2, 2, 1)$；
- 从 $\lambda^{(3)}$ 出发所能得到的唯一杨图是 $(3, 2, 1)$ 本身.

此页刻意留白

Problem I. 志愿者模拟器

在一场 ICPC 比赛中，有多个题目，且比赛持续时间较长。总共有 n 个正确提交，每个提交都代表一个队伍对某道题目的成功解答。当一个队伍对某道题目提交并通过时，我们称该队伍解决了该题目。需要注意的是，如果一个队伍对同一道题目有多个正确提交，只有第一次的提交才被视为首次解决该题目。

你需要根据以下规则判断每个提交是否需要发放气球：

- 在封榜前（提交时间 < 240 分钟）：一个队伍首次解决某道题目时，为该队发放对应题号的气球。
- 在封榜后（提交时间 ≥ 240 分钟）：一个队伍首次解决某道题目时，仅当该队伍截至目前获得的气球总数**严格少于** 3 个时，才为该队发放对应题号的气球。

Input

输入第一行包含 n $(1 \leq n \leq 5\,000)$，表示正确提交的数量。

接下来 n 行按时间顺序描述这些提交。每行包含三个整数 a $(1 \leq a \leq 410)$、b $(1 \leq b \leq 13)$ 和 c $(0 \leq c < 300)$，分别表示队伍编号、题目编号和提交时间（分钟）。保证提交时间单调不降。

Output

对于每个提交，如果需要发放对应题目的气球，则输出该题目编号，否则输出 0。

Example

standard input	standard output
8 1 1 10 1 2 20 1 3 30 1 4 250 1 5 260 1 6 270 1 7 280 2 1 290	1 2 3 0 0 0 0 1

此页刻意留白

Problem J. 时痕回响

公元3142年，在土星环深处，人类大崩溃前最后的记忆档案被封存在 “时痕密库” 之中。密库的入口由 “回响转盘” 守护，这是一种由量子纠缠驱动的锁具，由一排共 n 个时相环组成。每个时相环显示一个 0 到 $m-1$ 之间的整数值，表示其与宇宙节律的同步相位。初始时，第 i 个环显示的值为 a_i。

回响转盘只能通过 “共振调谐” 进行调整，规则如下：

- 在一次操作中，你可以选择任意相位值 $x \in [0, m)$，并将当前所有显示 x 的时相环同时进行如下调谐之一：
  - 向前调谐：$x \to (x+1) \bmod m$，或
  - 向后调谐：$x \to (x-1) \bmod m$。
- 此操作会影响整个转盘中所有当前相位为 x 的环，与它们的位置无关。

密库的智能守护者 “回响之灵” 会发出 q 次诊断试炼。每次试炼以查询 (l, r, v) 的形式给出，询问将位置 l 到 r（含）之间的所有时相环调整至相位 v 所需共振调谐的最少次数。

所有试炼均为假设性操作。每次查询均假设转盘从原始初始状态开始，且任何试炼都不会实际改变转盘的状态。作为最后一位量子档案师，你的职责是高效地回答全部 q 次查询。

Input

第一行包含三个整数 n、q $(1 \le n, q \le 2 \times 10^5)$ 和 m $(1 \le m \le 10^9)$。

第二行包含 n 个整数 $a_1, a_2, \ldots, a_n$ $(0 \le a_i < m)$ ，表示每个时相环的初始相位。

接下来的 q 行，每行包含三个整数 l、r $(1 \le l \le r \le n)$ 和 v $(0 \le v < m)$，描述一次诊断试炼。

Output

对于每个查询，输出一行包含一个整数，表示在转盘从初始状态出发、且各查询互不影响的前提下，将区间 $[l, r]$ 内所有环调整为相位 v 所需的最少操作次数。

Example

standard input	standard output
4 6 7 2 0 2 5 1 4 3 2 4 6 1 3 1 1 3 5 3 4 0 3 3 2	5 4 2 4 4 0

此页刻意留白

Problem K. 接力跳

有 n 只编号为 1 到 n 的青蛙，它们各自最初位于笛卡尔平面上的一个整点坐标处。一次外部的扰动突然刺激了其中一只青蛙，引发了一场连锁反应，青蛙们将根据以下规则行动。

当青蛙 i $(1 \le i \le n)$ 被刺激时，它可以选择：

- 跳向某只其它的青蛙 j $(1 \le j \le n, j \ne i)$，并瞬间落在其当前位置关于青蛙 j 当前所在点的对称点上（如果跳跃前青蛙 i 与青蛙 j 位置重合，则它会落在重合点上）；紧接着这次跳跃之后，青蛙 j 会成为下一只、也是唯一一只被刺激的青蛙；
- 或者停在原地，此时所有跳跃结束，不再有青蛙被刺激。

一只青蛙可能会在不同时刻被多次刺激，而有些青蛙可能永远不会被刺激。

你观察到第一只被刺激的青蛙编号为 s，但跳跃次数太多，你无法捕捉到整个过程。幸运的是，你认识每一只青蛙，所以你记录下了它们的初始位置 $P_1, P_2, \ldots, P_n$ 以及所有跳跃结束后的最终位置 $Q_1, Q_2, \ldots, Q_n$。你还注意到，所有的初始位置**两两不同**，并且最终位置也**两两不同**。

设 t 为最后一只被刺激的青蛙的编号，即被刺激时选择停在原地，之后不再有跳跃发生的青蛙。请你找出 t。

Input

第一行包含两个整数 n $(2 \le n \le 10^5)$ 和 s $(1 \le s \le n)$，分别表示青蛙的个数和第一只被刺激的青蛙的编号。

接下来 n 行的第 i 行包含四个整数 px_i、py_i、qx_i 和 qy_i $(-10^9 \le px_i, py_i, qx_i, qy_i \le 10^9)$，其中 $P_i = (px_i, py_i)$ 和 $Q_i = (qx_i, qy_i)$ 代表青蛙 i 在平面里的初始和最终位置。

保证所有初始位置 $P_1, P_2, \ldots, P_n$ 互不相同。所有最终位置 $Q_1, Q_2, \ldots, Q_n$ 也互不相同，并且对应某个符合上述规则、至多 2×10^5 次刺激的刺激序列。

Output

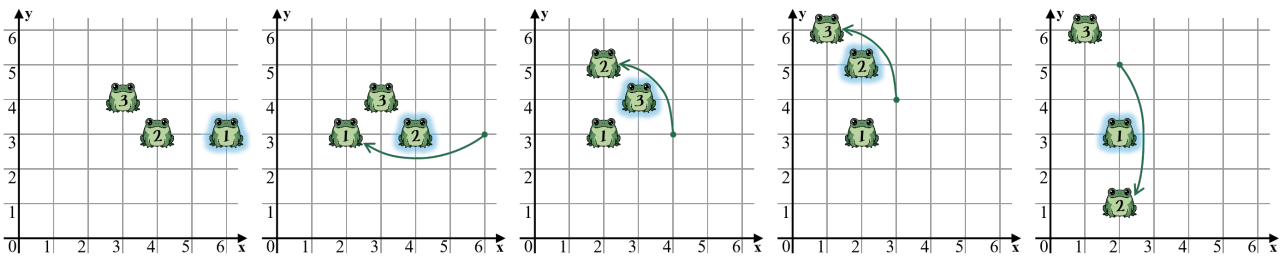
输出一个整数 t，表示最后一只被刺激的青蛙的编号。

Example

standard input	standard output
3 1 6 3 2 3 4 3 2 1 3 4 1 6	1

Note

样例中对应的刺激序列为 $[1, 2, 3, 2, 1]$。



此页刻意留白

Problem L. 狮子座

在数字逻辑已然超脱于二进制的未来，一种被称为三色逻辑的全新信号处理系统已成为标准。该系统分两类定义了的四种基本状态：

- **有色**：有三种不同的有色状态，分别表示为 R (红色)、G (绿色) 和 B (蓝色)。每种有色状态都可作为一种可区分的 "开" 状态，类似于传统逻辑中的 1 信号。
- **无色**：唯一的无色状态表示为 $*$ (透明)。它可作为一种 "关" 状态，类似于传统逻辑中的 0 信号。

耳廓狐亚砜正在设计一款基于三色逻辑技术的计算设备，其名为 Leo（狮子座）。它采用简单的组合逻辑结构，因此我们可以将其视为一个有向无环图，其中每个节点都持有一个来自集合 $\{\mathrm{R}, \mathrm{G}, \mathrm{B}, *\}$ 的状态。在将机器的输入规模固定为 n 后，该图按如下方式构造：

- 节点 $1, 2, \ldots, n$ 是**输入节点**，其状态将作为机器的输入给出。输入节点没有前驱。
- 节点 $n+1, n+2, \ldots, n+m$ 是**内部节点**，其状态由两个前驱的状态计算得出。每个内部节点的两个前驱可以相同，且从标号严格小于其自身标号的节点中选择。为保证机器的运行速度，m (内部节点的数量) **不得超过** $6n$。
- 内部节点 $n+m$ 同时也是唯一的**输出节点**，其状态即为机器的输出。

每个内部节点都恰好是以下两种类型之一，具体取决于它如何组合其两个前驱的信号：

- **三色 AND**：若其前驱的状态相同，则自身状态也为该状态。若状态不同，则自身状态为 $*$。
- **三色 OR**：若其前驱的状态相同，则自身状态也为该状态。若一个前驱的状态为 $*$ 而另一个为有色状态，则自身状态为该有色状态。若两个前驱的状态不同且均为有色状态，则自身状态为与这两个状态都不同的那个有色状态。

一切准备就绪，现在让我们规定 Leo 的预期功能。给定的 n 个输入节点的状态将保证其中至少包含一个 R、一个 G 和一个 B，而 $*$ 可能出现任意次数（包括零次）。对于每一个这样的输入，输出节点 $n+m$ 的状态必须与从节点 1 到节点 n 考察输入节点时**遇到的第三种不同的有色状态**相同（忽略所有 $*$）。换言之，在 R、G、B 中，它必须输出那个首次出现位置标号最大的有色状态。

给定机器固定的输入规模 n，请您完成内部节点的设计，即指定内部节点的数量以及每个内部节点的类型和其前驱。

为了检验你设计的正确性，对于每个测试点，评测机将生成 $\left\lfloor \frac{10^7}{n} \right\rfloor$ 组合法的输入用例，并依次作为你的机器的输入。若在所有输入用例中，你的机器的输出始终符合 Leo 的预期功能，则你的解决方案通过此测试点。

Input

仅有的一行包含一个整数 n ($3 \le n \le 10^5$)，表示输入节点的个数。

Output

第一行中输出一个整数 m ($1 \le m \le 6n$)，表示你的设计里内部节点的数量。

然后输出 m 行，第 i 行包含一个字符 t_i ($t_i \in \{\text{'\&'}, \text{'|'}\}$) 和两个整数 u_i 和 v_i ($1 \le u_i, v_i < n+i$)，表示内部节点 $n+i$ 的类型及其两个前驱的标号。其中 '&' 代表三色 AND 类型，'|' 代表三色 OR 类型。

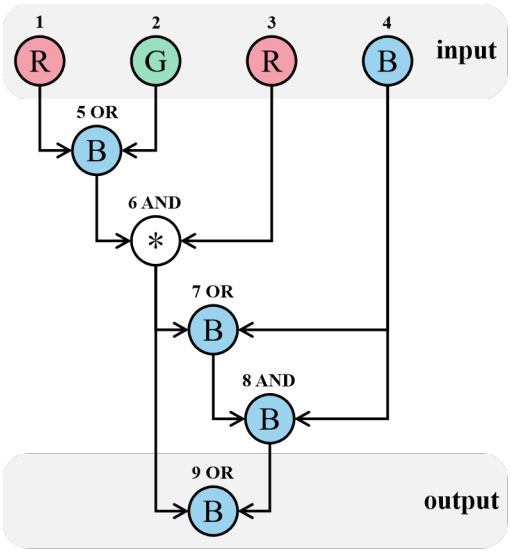
若存在多种设计方案，你可以输出任意一种。注意，你不需要最小化 m 的值。

Example

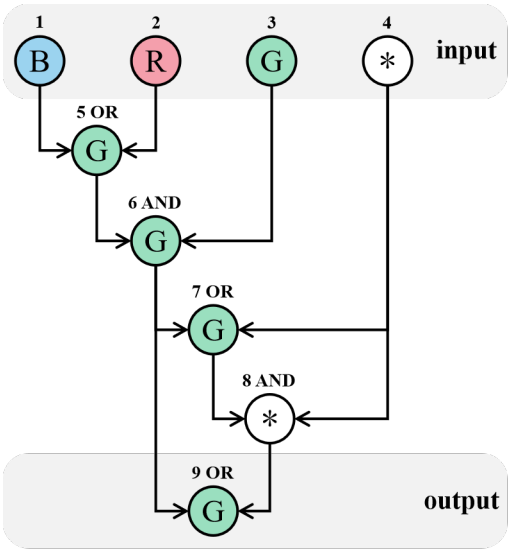
standard input	standard output
4	5 | 1 2 & 3 5 | 4 6 & 4 7 | 6 8

Note

可以证明，对于 $n = 4$ 的任意合法输入，样例中给出的设计总能找到遇到的第三种不同的有色状态。例如，下图展示了机器对输入 “RGRB” 的计算过程：



下图展示了机器对输入 “BRG∗” 的计算过程：



Problem M. 大结局?

在 2025 年《英雄联盟》S15 全球总决赛中，T1 战队与 KT 战队鏖战五局，最终 T1 以 3:2 的比分战胜 KT，成功夺冠。这场比赛意义非凡，诞生了多项新的历史纪录。

作为 T1 的核心中单，Faker 选手在此次夺冠后，个人职业生涯的 S 赛冠军数已达到 6 次，这让他“英雄联盟第一人”的地位更加无可撼动。除了 Faker，队内的 Oner、Gumayusi 和 Keria 也一同实现了 S 赛的三连冠伟业。同时，新加入的上单选手 Doran 也凭借此次冠军，填补了个人荣誉的空白，首度加冕全球总决赛冠军。

而 LPL 赛区的征程充满戏剧性。赛前被寄予厚望的 BLG 战队在瑞士轮首轮便爆冷输给北美战队 100T，随后在关键的出线战中不敌 TES，作为一号种子止步十六强，成为本届赛事的一大冷门。TES 在半决赛中被 T1 3:0 横扫，LPL 赛区无缘总决赛。LPL 是否迎来了大结局?

当我们回顾这两支决赛队伍的旅程，一切都始于那生死攸关的淘汰赛阶段。八队单败淘汰赛制规定，各支队伍根据其在瑞士轮阶段的表现被分配至赛程表中，进入逐轮的一对一鏖战，败者立即面临淘汰。整个赛程共分三轮：四分之一决赛由八支队伍决出四强，半决赛由这四支队伍决出双雄，决赛再从这两支队伍中加冕一位无上的冠军。

在 S16 中，我们的 1 号队伍能否夺冠呢？请你找到队伍 1 赢得整个锦标赛的最大概率。

更具体地，有 8 支队伍参加一场单败淘汰制锦标赛。每支队伍 i 有两个强度值 a_i 和 b_i。

在锦标赛开始前，你需要为每支队伍分配 1 到 8 的不同种子。锦标赛采用标准的单败淘汰制对阵格式：

- 第一轮：种子 1 对阵种子 2，种子 3 对阵种子 4，种子 5 对阵种子 6，种子 7 对阵种子 8
- 第二轮：$(1,2)$ 的胜者对阵 $(3,4)$ 的胜者，$(5,6)$ 的胜者对阵 $(7,8)$ 的胜者
- 第三轮（决赛）：上半区 $(1,2,3,4)$ 的胜者对阵下半区 $(5,6,7,8)$ 的胜者

当两支队伍相互比赛时，种子编号较小的队伍使用其 a 值作为强度，种子编号较大的队伍使用其 b 值作为强度。如果一支强度为 x 的队伍与一支强度为 y 的队伍比赛，则前者获胜的概率为 $\frac{x}{x+y}$。

你需要考虑所有可能的队伍种子分配方案，找出队伍 1 赢得整个锦标赛的最大概率。

Input

输入包含 8 行。第 i 行包含两个整数 a_i 和 b_i（$1 \le a_i, b_i \le 100$），表示队伍 i 的两个强度值。

Output

输出一行包含一个实数，表示队伍 1 赢得锦标赛的最大概率。

如果你的答案的绝对误差或相对误差不超过 10^{-6}，则被认为是可接受的。形式化地说，假设你的输出为 a，评测方的答案为 b，当且仅当 $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$ 时，你的输出被接受。

Examples

standard input	standard output
10 80 20 70 30 60 40 50 50 40 60 30 70 20 80 10	0.329505822460368
100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100	0.125000000000000