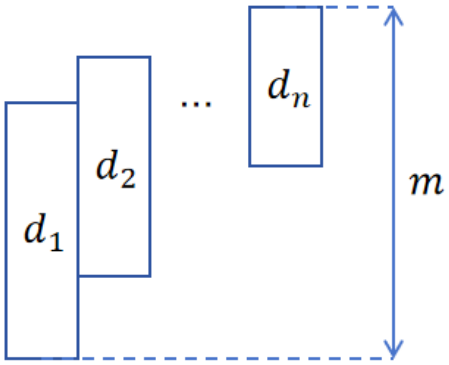
Problem A. 安全第一

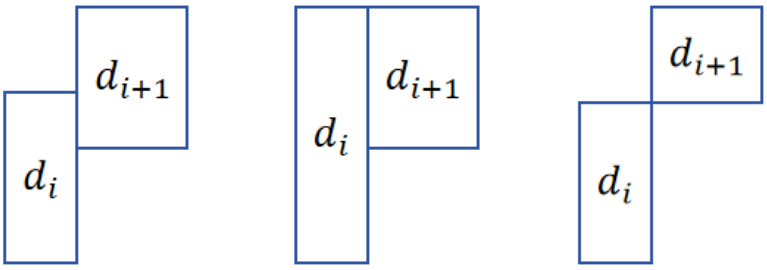
一个梯子 (Ladder) 由 n 段构成，每一段的长度都是正整数，并且长度是依次非递增的。也就是说，这 n 段的长度 $d_1, d_2, \ldots, d_n$ 要满足 $d_1 \geq d_2 \geq \cdots \geq d_n$。



Ladder

小 Q 需要一个竖立起来高度恰好等于 m 的梯子，这里梯子的高度是指它的最高点与最低点的海拔高度差。出于安全原因，这个梯子需要是稳定 (Stable) 的，这意味着要满足以下要求：

- 梯子第一段的下端必须接触地面；
- 对每个 $i = 1, 2, \ldots, n-1$，第 i 段的上端必须锁定在第 $(i+1)$ 段的上端或下端。



Not stable Stable Stable

你需要求出有多少种不同的由 n 段构成的竖立起来高度恰好等于 m 的稳定梯子。如果存在某个 k 使得两个梯子的第 k 段的长度不同，或者一个梯子的第 k 段锁定在第 $(k+1)$ 段的上端，而另一个梯子的第 k 段锁定在第 $(k+1)$ 段的下端，则这两个梯子被视为不同的。

由于答案可能很大，请输出其对 $998\,244\,353$ 取模后的结果。

Input

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

仅有的一行包含两个整数 n 和 m $(1 \leq n, m \leq 2\,000)$，表示段数和稳定梯子的所需高度。

Output

对于每组测试数据，输出一行包含一个整数，表示不同的稳定梯子的数量对 $998\,244\,353$ 取模后的结果。

Examples

standard input	standard output
3 1 3 2 3 3 3	1 4 10
1 2000 2000	204576309

Note

在下面的解释中，我们用**粗体**表示对应段的上端是和下一个段的下端锁定，并在其他情况保持原状。

在第一个样例中：

- 对于第一组测试数据，唯一可行的梯子包含一个长度为 1 的段；
- 对于第二组测试数据，所有 4 种可行梯子的段长分别为 $[\mathbf{2}, 1]$, $[3, 1]$, $[3, 2]$ 和 $[3, 3]$；
- 对于第三组测试数据，所有 10 种可行梯子的段长分别为 $[\mathbf{1}, \mathbf{1}, 1]$, $[\mathbf{2}, 1, 1]$, $[2, \mathbf{1}, 1]$, $[2, \mathbf{2}, 1]$, $[3, 1, 1]$, $[3, 2, 1]$, $[3, 2, 2]$, $[3, 3, 1]$, $[3, 3, 2]$ 和 $[3, 3, 3]$。

Problem B. 神奇的调色板

小白兔有一块神奇的调色板，上面有一个 n 行 m 列的网格。在开始调色前，小白兔会在每一行的左侧挤出一种颜料，记作 $a_1, a_2, \ldots, a_n$，并且在每一列的上方挤出一种颜料，记作 $b_1, b_2, \ldots, b_m$。

一共有 $n \times m$ 种可用的颜料，用整数 $0, 1, 2, \ldots, nm-1$ 表示不同的颜料。然后在第 i 行和第 j 列的格子中，小白兔将使用第 i 行左侧的颜料 a_i 和第 j 列上方的颜料 b_j 混合出一种颜色 $c_{i,j} = a_i b_j \bmod nm$。

小白兔希望 $n \times m$ 个格子里都是不同的颜色，你需要判断是否可行。

Input

输入的第一行包含一个整数 T $(1 \le T \le 10^4)$，表示测试数据的组数。对于每组测试数据：

仅有的一行包含两个整数 n 和 m $(1 \le n, m \le 10^6,\ 1 \le n \times m \le 10^6)$，表示行数和列数。

保证所有测试数据 $n \times m$ 的总和不超过 10^6。

Output

对于每组测试数据，如果不存在可行方案，则在一行内输出 "`No`"（不含引号）。否则，输出三行：

- 第一行包含一个字符串 "`Yes`"（不含引号）。
- 第二行包含 n 个整数 $a_1, a_2, \ldots, a_n$ $(0 \le a_i < nm)$。
- 第三行包含 m 个整数 $b_1, b_2, \ldots, b_m$ $(0 \le b_i < nm)$。

Example

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

Note

对于第一个样例，$[c_{1,1}, c_{1,2}, c_{1,3}, c_{2,1}, c_{2,2}, c_{2,3}] = [1, 3, 5, 2, 0, 4]$，这些颜色都是互不相同的。

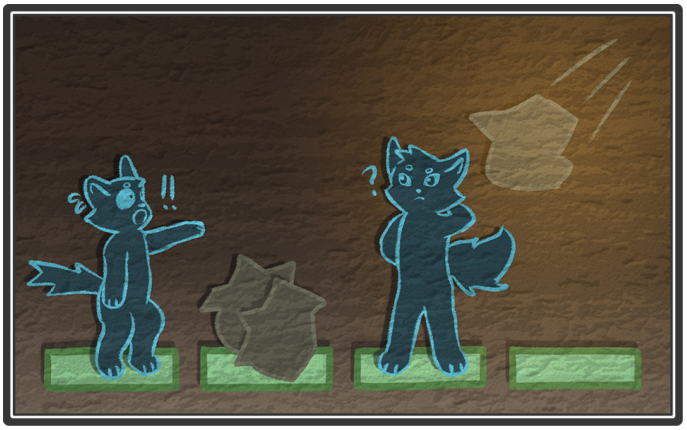
This page is intentionally left blank

Problem C. 突发事件：陨石

耳廓狐亚矾最近沉迷于《异界失控》这款围绕其独创的横版一维战场系统构建的 Roguelite 战术回合制游戏。当他在这个特殊的战场上利用关键的站位机制与合理的技能点分配，操控自己的战士小队击溃敌方、所向披靡时，他遇到了又一个突发事件（游戏设计师准备的“厚礼”）作为新章节挑战升级而引入的变数 —— 陨石。

针对本题，我们不妨让这个突发事件比原版游戏还要极端。考虑一个由从 1 到 n 按顺序编号的 n 个地块组成的线状战场。初始战场上没有任何陨石，只有若干角色被置于某些地块上。亚矾的目标是确保所有角色在 m 轮的陨石灾难中无一伤亡。

第 i 轮 $(i = 1, 2, \ldots, m)$ 开始时，致命的陨石危机将迫在眉睫：对于每个 $j = 1, 2, \ldots, n$，恰好 $a_{i,j}$（可能为零）颗陨石降落在第 j 个地块上，并与原有的陨石叠加。该阶段严禁移动角色，这意味着任何不幸站在有陨石降落的地块上的角色都将瞬间被压成烙饼。



不要让角色站在有陨石降落的地块上

在第 i 轮的陨石全部落地后，本轮将进入行动阶段，此时亚矾可以将任意存活的角色移动到相邻的地块，这种移动在本轮结束前对各角色都可以执行任意多次（或零次）。然而，现有的陨石会阻挡角色进入其所占据的地块。因此，在角色进入这样的地块之前，亚矾必须为该地块上每个陨石分别花费 1 的点数来摧毁它们。

请帮助亚矾计算在陨石灾难中确保所有角色存活所需的最少点数，或者报告达成该目标是不可能的任务。

Input

输入的第一行包含一个整数 T $(1 \le T \le 10^4)$，表示测试数据组数。对于每组测试数据：

第一行包含两个整数 n 和 m $(1 \le n, m \le 10^6, 1 \le n \times m \le 10^6)$，分别表示地块个数和轮数。

第二行包含 n 个整数 $c_1, c_2, \ldots, c_n$ $(c_i \in \{0, 1\})$。当且仅当 $c_i = 1$ 时，有一个角色初始被置于第 i 个地块上。保证游戏中至少存在一个角色。

接下来 m 行，其中第 i 行包含 n 个整数 $a_{i,1}, a_{i,2}, \ldots, a_{i,n}$ $(0 \le a_{i,j} \le 1\,000)$，表示第 i 轮开始时有 $a_{i,j}$ 块陨石降落在第 j 个地块上。

保证所有测试数据 $n \times m$ 的总和不超过 10^6。

Output

对于每组测试数据，如果不可能确保所有角色存活，在一行内输出 -1；否则，在一行内输出一个整数，表示所需的最少点数。

Example

standard input	standard output
2 3 4 1 0 1 0 1 0 2 0 0 0 0 3 4 5 0 1 1 1 1000	4 -1

Problem D. 点积游戏

本题中，1 到 n 的一个排列是下标从 1 开始的长度为 n 的整数序列，其中每个整数从 1 到 n 恰好出现一次。Alice 和 Bob 正在玩一个关于 $A = [a_1, a_2, \ldots, a_n]$ 和 $B = [b_1, b_2, \ldots, b_n]$ 的游戏，这两个序列都是 1 到 n 的排列。他们轮流进行操作，Alice 先行，无法进行任何操作的人输掉游戏。

在每一轮中，Alice 只能对排列 A 进行操作，而 Bob 只能对排列 B 进行操作。一个有效的操作包括选择可操作排列中的两个下标 i 和 j $(1 \le i, j \le n)$ 并交换它们对应的元素，要求这两个排列的点积 $\sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ 在交换后必须严格增加。也就是说，如果一个玩家无法进行任何交换以增加点积，则该玩家输掉游戏，另一个玩家获胜。

由于 Alice 和 Bob 都足够聪明，能够采用最佳策略，因此可以从一开始就确定游戏的赢家。因此，他们决定进行 n 轮游戏，并进行一些修改以使游戏不那么无聊。请帮助他们预测这 n 场游戏的赢家。

更具体地说，将给定两个初始排列 $A = [a_1, a_2, \ldots, a_n]$，$B = [b_1, b_2, \ldots, b_n]$，以及 $(n-1)$ 个修改 $[(t_1, l_1, r_1, d_1), (t_2, l_2, r_2, d_2), \ldots, (t_{n-1}, l_{n-1}, r_{n-1}, d_{n-1})]$。第一场游戏将从给定的排列 A 和 B 开始，对于 $k = 1, 2, \ldots, n-1$，第 $(k+1)$ 场游戏将从第 k 场游戏的排列，将排列 t_k 的索引区间 l_k 和 r_k 左移 d_k 次之后得到的排列开始。

请注意，将区间 $[p_l, p_{l+1}, \ldots, p_r]$ 左移一次将得到 $[p_{l+1}, p_{l+2}, \ldots, p_r, p_l]$。例如，将 $[1, 2, 3, 4, 5]$ 的区间 $[2, 4]$ 左移一次将得到 $[1, 3, 4, 2, 5]$，左移两次将得到 $[1, 4, 2, 3, 5]$。此外，将 $[3, 1, 4, 5, 2]$ 的区间 $[1, 5]$ 左移 4 次将得到 $[2, 3, 1, 4, 5]$。

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：

第一行包含一个整数 n $(1 \le n \le 5 \times 10^5)$，表示排列的长度。

第二行包含 n 个不同的整数 $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le n)$，表示排列 A。

第三行包含 n 个不同的整数 $b_1, b_2, \ldots, b_n$ $(1 \le b_i \le n)$，表示排列 B。

接下来的 $(n-1)$ 行中，第 i 行包含一个字符 t_i $(t_i \in \{\texttt{A}, \texttt{B}\})$ 以及三个整数 l_i，r_i $(1 \le l_i \le r_i \le n)$ 和 d_i $(1 \le d_i \le n)$。

保证所有测试数据的 n 之和不超过 5×10^5。

Output

对于每组测试数据，在一行内输出长度为 n 的字符串，如果 Alice 赢得第 i 场游戏那么第 i 个字符是 `A`，否则是 `B`（即 Bob 赢得第 i 场游戏）。

Example

standard input	standard output
5 3 1 2 3 1 2 3 A 1 1 1 B 1 1 1 3 1 2 3 2 1 3 A 1 2 1 B 2 2 1 3 1 2 3 2 1 3 A 1 3 1 B 1 2 1 3 1 2 3 3 2 1 A 2 2 1 B 2 3 1 10 1 2 3 4 5 6 7 8 9 10 4 2 3 9 6 1 5 8 7 10 A 2 9 10 B 2 7 9 A 1 10 8 B 4 6 7 B 3 10 6 A 2 5 5 A 8 9 4 B 3 9 3 A 2 7 2	BBB ABB AAB AAB AABBBBAAAA

Note

对于样例的第三组测试数据：

- 第一场游戏以 $A = [1, 2, 3]$，$B = [2, 1, 3]$ 开始；
- 第二场游戏以 $A = [2, 3, 1]$，$B = [2, 1, 3]$ 开始；
- 第三场游戏以 $A = [2, 3, 1]$，$B = [1, 2, 3]$ 开始。

Problem E. 点亮网格

小 Q 正在测试一款自制的益智游戏，该游戏涉及一个 2×2 网格，其中每个格子可以处于开启或关闭状态。当一个格子被切换时，其状态从关闭变为开启，或从开启变为关闭。游戏允许以一定的代价执行以下四种操作：

- 切换单个格子，代价为 a_0；
- 切换一行格子，代价为 a_1；
- 切换一列格子，代价为 a_2；
- 切换所有格子，代价为 a_3。

小 Q 意识到游戏已经开始，但是屏幕不知何故出现故障，导致他无法看到任何格子的当前状态。他唯一能收到的反馈是一种特殊的提示音：如果在某个操作后，所有格子都是开启状态，该提示音会被触发。

小 Q 事先知道游戏的所有 m 种可能初始网格，他想找到一个操作序列，使得无论游戏开始时的初始网格是哪种，他在按照序列的顺序进行操作后，总是能听到提示音。

请计算这样一个操作序列的最小总代价。

Input

输入的第一行包含五个整数 T $(1 \le T \le 10^4)$，a_0，a_1，a_2 和 a_3 $(1 \le a_0, a_1, a_2, a_3 \le 10^6)$，表示游戏的数量和每个操作的代价。对于每个游戏：

第一行包含一个整数 m $(1 \le m \le 16)$，表示游戏的可能初始网格的数量。

接下来是 m 个网格。对于每个网格：

- 如果不是游戏中的第一个网格，将首先有一个空行。
- 接下来的两行中，第 i 行包含一个长度为 2 的字符串 $s_{i,1}s_{i,2}$，由字符 `0` 和 `1` 组成，描述一个 2×2 的可能初始网格。设 (i, j) 为第 i 行和第 j 列的格子。如果 $s_{i,j}$ 是 `0`，那么格子 (i, j) 是关闭状态，否则格子 (i, j) 是开启状态。

保证每个游戏中的给定的网格互不相同。

Output

对于每个游戏，输出一行包含一个整数，表示最小总代价。

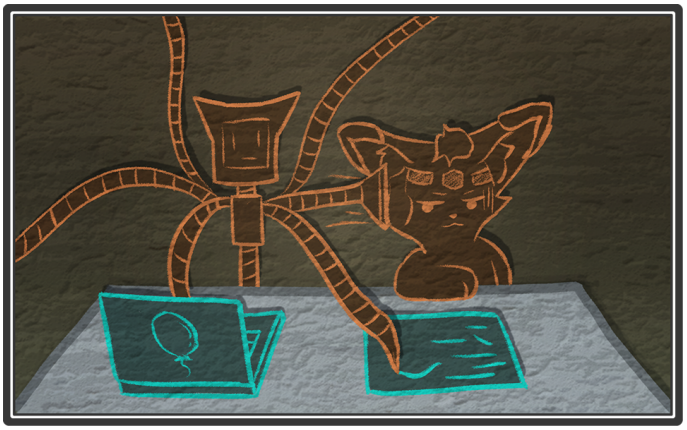
Example

standard input	standard output
2 1000 100 10 1 4 10 00 01 00 00 10 00 01 1 11 11	1121 2

Problem F. 点亮超立方体

仿生人会梦见电子羊吗？这个数十年前提出的论题，以全新的紧迫感让未来世界如坐针毡。它苦涩地提醒着未来社会，生物体已将思考和创造几乎全然拱手献给了人工智能。

2040 年的一个下午，当耳廓狐亚砜沉浸在由小 Q 设计的一款简单二维益智游戏（介绍于 **Problem E. 点亮网格**）的乐趣中时，他的解法引起了他定制 AI 系统的注意。这个 AI，尽管在那些年里已演变得近乎万能，却无法与亚砜纯粹消遣的本意共情，误将他的快乐识别为渴望更大的挑战。于是，AI 展现了自己对高维空间的超俗掌握，自发地提供了一个只有它才能真正领悟的 n 维益智游戏。



自视甚高的定制 AI 系统

强化版的益智游戏涉及一个 n 维超立方体，它的每条边都只沿着 n 维中的某一个维度延伸，并且它 2^n 个顶点里的每个顶点都配备了一盏可以处于开启或关闭状态的灯。当一盏灯被切换时，其状态从关闭变为开启，或从开启变为关闭。

与简单版的益智游戏不同，此时有 2^n 种允许以一定的代价执行的切换操作，分别从 0 到 (2^n-1) 编号。具体地，对于操作 i $(i=0,1,\ldots,2^n-1)$：

- 执行该操作的代价等于 a_i；
- 令 $\overline{b_n b_{n-1}\ldots b_1}$ 为 i 的二进制表示，其中每个 $b_j \in \{0,1\}$ $(j=1,2,\ldots,n)$ 表示二进制表示从低到高的第 j 位，则该操作允许选择超立方体中任意合法的 k 维面（维数为 k 的子立方体），其中 k 是二进制表示中 $b_j=1$ 的数量，每个 $b_j=1$ 约束所选 k 维面必须沿着第 j 个维度延伸，每个 $b_j=0$ 约束所选 k 维面必须在第 j 个维度上无厚度。然后，切换所选 k 维面各顶点的灯。

例如，操作 0 用于切换单个顶点（0 维面）的灯，代价为 a_0；操作 $2^0, 2^1, \ldots, 2^{n-1}$ 用于切换沿着相应维度延伸的一条边（1 维面）两个端点的灯，代价分别为 $a_{2^0}, a_{2^1}, \ldots, a_{2^{n-1}}$；操作 (2^n-1) 用于切换所有灯，代价为 a_{2^n-1}。

亚砜意识到游戏已经开始，但是屏幕已被 AI 屏蔽，导致他无法看到任何灯的当前状态。他唯一能收到的反馈是一种特殊的提示音：如果在某个操作后，所有灯都是开启状态，该提示音会被触发。

因此，他需要找到一个操作序列，使得无论游戏开始时灯的初始状态如何（共 2^{2^n} 种可能性），他在按照序列的顺序进行操作后，总是能听到提示音。

作为三维生物，亚砜感到无法想象更高维度，但他相信这样一个操作序列的最小总代价仍是可计算

的。“好的，AI，现在我需要全手动控制。” 言罢，他投身解题，决心在认知之池放纵地游泳，而不是被救生衣束缚，直至忘却如何漂浮。

请计算最小总代价对 $998\,244\,353$ 取模后的结果，因为它可能特别大。

Input

输入的第一行包含一个整数 T $(1 \le T \le 10^4)$，表示测试数据组数。对于每组测试数据：

第一行包含一个整数 n $(2 \le n \le 20)$，表示超立方体的维数。

第二行包含 2^n 个整数 $a_0, a_1, \ldots, a_{2^n-1}$ $(1 \le a_i \le 10^6)$，表示每个操作的代价。

保证所有测试数据 2^n 的和不超过 2^{20}。

Output

对于每组测试数据，输出一行包含一个整数，表示最小总代价对 $998\,244\,353$ 取模后的结果。

Example

standard input	standard output
2 2 4 2 2 1 4 3 3 2 3 1 3 2 2 2 3 2 1 1 2 3 2	25 65666

Problem G. 猜测多边形

这是一道交互题。

给定一个简单多边形，但在给出多边形的 n 个顶点之前，小 Q 已经将它们打乱了。

现在你可以向小 Q 提出不超过 $(n-2)$ 个询问，每个询问包含两个整数 p 和 q（$0 \le p/q \le 1\,000, 1 \le q \le 1\,000$）。他会告诉你在 $x = p/q$ 这条线上位于多边形内部或边界上的点的总长度，可以表示为 r/s，其中 r 和 s 是两个整数（$r \ge 0, s \ge 1, \gcd(r,s) = 1$）。这里 $\gcd(r,s)$ 是 r 和 s 的最大公约数。

你需要求出多边形的面积，也可以表示为 u/v，其中 u 和 v 是两个整数（$u \ge 0, v \ge 1, \gcd(u,v) = 1$）。

多边形及其顶点的顺序在交互开始之前就已确定，并且不依赖于你的询问。也就是说，交互器不是自适应的。

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：

第一行包含一个整数 n $(3 \le n \le 1\,000)$，表示多边形的顶点数量。

接下来 n 行，每行包含两个整数 x 和 y $(0 \le x, y \le 1\,000)$，按照乱序给出每个多边形顶点的坐标 (x, y)。

多边形是简单的，即其顶点是不同的，并且多边形的两条边没有交叉或接触，除了相邻的边在它们的公共顶点处接触。此外，两个相邻的边不共线。

保证所有测试数据 n 的总和不超过 $1\,000$。

Interaction Protocol

如果你想提出一个询问，输出一行。首先输出 `?` 后接一个空格，然后输出两个整数 p 和 q $(0 \le p/q \le 1\,000, 1 \le q \le 1\,000)$。在刷新缓冲区后，你的程序应该读取两个整数 r 和 s $(r \ge 0, s \ge 1, \gcd(r,s) = 1)$，表示询问的答案。

如果你想猜测多边形的面积，输出一行。首先输出 `!` 后接一个空格，然后输出两个整数 u 和 v $(u \ge 0, v \ge 1, \gcd(u,v) = 1)$，表示多边形的面积是 u/v。在刷新缓冲区后，你的程序应该继续处理下一组测试数据，或者如果没有更多测试数据则应该立即退出。请注意，你的猜测不算入询问次数。

要刷新缓冲区，你可以使用：

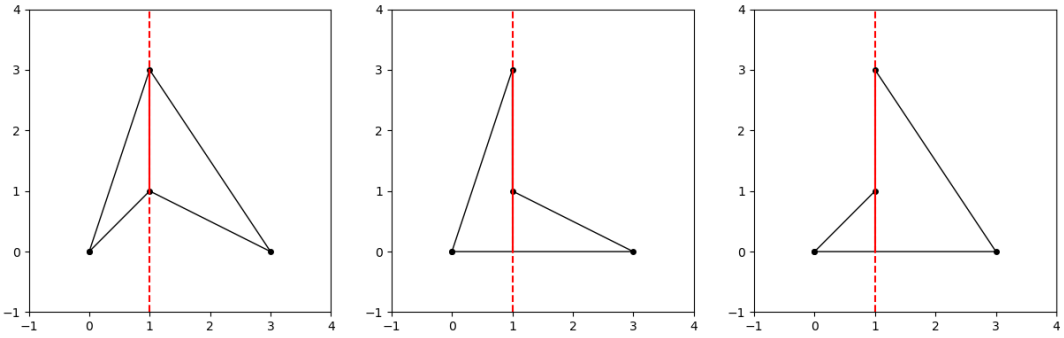
- C 和 C++ 使用 `fflush(stdout)`（如果你使用 `printf`）或 `cout.flush()`（如果你使用 `cout`）。
- Java 和 Kotlin 使用 `System.out.flush()`。
- Python 使用 `sys.stdout.flush()`。

Example

standard input	standard output
2 4 0 0 1 3 1 1 3 0 2 1 3 0 0 999 1000 1000 999 1999 999000	 ? 1 1 ! 3 1 ? 1 1 ! 1999 2

Note

对于第一个样例，有三个可能的多边形，如下图所示。只有最左边的一个满足询问 $x = 1$ 的答案为 2，因此面积为 3。其他两个多边形对询问 $x = 1$ 的答案为 3。



对于第二个样例，只有一个可能的多边形，因此面积为 $1999/2$。

请注意，样例中给出的顶点顺序可能与实际提供的不一致。

样例中的空行是为了可读性而添加的。在你的输出中，额外的空格或空行将被忽略。

Problem H. 导览图

比特大陆有 n 个城市，任何两个城市之间都有一条双向道路。恰好有 $(n-2)$ 条道路上有旅游景点，并且如果再选取一条合适的道路修建旅游景点，就可以从一个城市只通过有旅游景点的道路到其他城市。

小 Q 最初位于城市 1，手中拿着一张导览图开始了他的旅行，地图上展示了比特大陆的一些道路。假设他当前在城市 u:

- 如果他可以沿着导览图上的一条道路走到一个尚未游览过的城市 v，他将选择最小的这样的 v 并沿着道路走到城市 v。注意，城市 1 是一开始就游览过的;
- 否则，他将沿着他第一次到达城市 u 的道路返回，除非他已经在城市 1，在这种情况下，旅行立即结束。

小 Q 希望在旅行中游览所有的旅游景点，同时只经过不多于一条不同的没有旅游景点的道路。你需要帮助他计算满足他要求的不同导览图的数量。如果一张地图上的道路在另一张地图上不存在，则这两张导览图被视为不同。

由于答案可能很大，请输出其对 $998\,244\,353$ 取模后的结果。

Input

第一行包含一个整数 n $(2 \le n \le 2 \times 10^5)$，表示城市的数量。

接下来有 $(n-2)$ 行，其中第 i 行包含两个整数 u 和 v $(1 \le u, v \le n)$，表示第 i 个旅游景点位于第 u 个和第 v 个城市之间的道路上。保证如果再选取一条合适的道路修建旅游景点，就可以从一个城市只通过有旅游景点的道路到其他城市。

Output

输出一行包含一个整数，表示不同导览图的数量对 $998\,244\,353$ 取模后的结果。

Examples

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

Note

对于第一个样例，一个可行的导览图展示了 4 条道路 $(1,4)$、$(2,3)$、$(1,2)$ 和 $(1,3)$。小 Q 按照这个导览图将依次游览城市 $1 \to 2 \to 3 \to 2 \to 1 \to 4 \to 1$，在沿途旅行时会参观道路 $(1,4)$ 和 $(2,3)$ 上的旅游景点，同时仅经过一条没有景点的道路 $(1,2)$。

对于第二个样例，注意到不存在任何道路上有旅游景点，所以导览图可以包含也可以不包含唯一可能的道路 $(1,2)$。

This page is intentionally left blank

Problem I. 种树

小 Q 种下了一棵树，这棵树初始只有一个根节点 1，同时也是叶子节点。在每天上午，每个叶子节点 u 会萌发出两条树枝：

- 一条树枝连接节点 u 和节点 $2u$，其长度是 a_{2u}；
- 另一条树枝连接节点 u 和节点 $2u+1$，其长度是 a_{2u+1}。

之后节点 u 不再是叶子节点，而节点 $2u$ 和节点 $2u+1$ 成为新的叶子节点。这棵树会生长 n 天，最终长成有 $\left(2^{n+1}-1\right)$ 个节点的树，其中 2^n 个节点是叶子节点。

在这 n 天里，每天下午小 Q 可以选择一条当前已经长出来的树枝进行 “修剪”，将其长度修改为任意的正整数（可以比原来的值大或小），或者也可以不对任何树枝进行修剪。

小 Q 希望在这棵树长成之后，根节点到每个叶子节点的简单路径上的树枝长度之和两两不同。你需要帮他计算出最少要修剪多少条树枝，或者指出这是不可能做到的。

Input

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

第一行包含一个整数 n $(1 \le n \le 10)$，表示树生长的天数。

第二行包含 $\left(2^{n+1}-2\right)$ 个整数 $a_2, a_3, \ldots, a_{2^{n+1}-1}$ $(1 \le a_i \le 10^8)$，表示树枝的长度。

保证 n 大于 7 的测试数据不超过 10 组。

Output

对于每组测试数据，输出一行包含一个整数，表示满足要求所需修剪的最少树枝数量，如果不可能满足要求，则输出 -1。

Example

standard input	standard output
3 2 1 2 4 3 2 1 2 1 2 3 3 2 1 2 1 2 3 3 1 1	1 2 -1

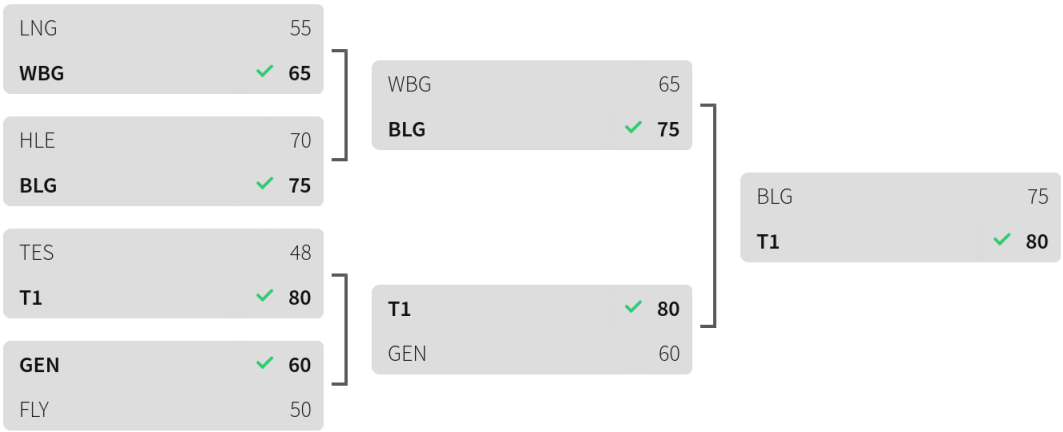
This page is intentionally left blank

Problem J. 眼见为实

2024 年 11 月 2 日，2024 英雄联盟全球总决赛在两支英雄联盟职业联赛巨头的史诗级交锋中落下帷幕。Bilibili Gaming，在 2024 年凭借双赛季国内冠军头衔称霸 LPL 赛区，无疑肩负着中国联赛时隔三年重夺召唤师奖杯的希望。横亘在征途尽头的正是 T1，其虽以 LCK 赛区四号种子的身份出征，却已然在世界赛中展现了属于他们的冠军风采。而他 —— Faker，那个被尊为 “最高的山，最长的河” 的男人 —— 仍屹立在英雄联盟最高舞台的中央，仿佛光阴也知晓对他敬畏三分。承载着四个王冠之重，这份传奇履历使 Faker 与第五冠的双向奔赴不再是缘于雄心，更像是命中注定。此刻，于伦敦 O2 竞技场宏大的穹顶之下，两支战队誓将孤注一掷：BLG 志在终结 LPL 的冠军荒，续写他们 2024 年的辉煌，而 T1 力求卫冕冠军并进一步确立他们的王朝。

一场惊心动魄的五局三胜系列赛将两支战队都逼至极限，最终 T1 以 3-2 的比分艰难战胜了 BLG，斩获了他们队史的第五个世界冠军。诚如本届世界赛的官方主题 “眼见为实” 所言，T1 在 1-2 落后的情况下实现了惊天逆转，再次声明了为什么他们在国际赛场上始终是 LPL 战队似乎不可逾越的鸿沟。

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，其中参与单败淘汰赛的八支队伍是：LNG Esports，Weibo Gaming，Hanwha Life Esports，Bilibili Gaming，Top Esports，T1，Gen.G，以及 FlyQuest。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。



图：八队单败淘汰赛的赛程表

尽管 2024 英雄联盟全球总决赛已不可挽回地成为历史，我们仍然可以创造我们自己念想的故事。在本题中，你将使用相同的八队单败淘汰赛制模拟一个假设的淘汰赛阶段。你将基于赛程表中八强从上到下的顺序（如图中最左列所示）依次获得八支战队各自的名称和实力值，保证名称和实力值均互不相同。每当两支战队对决时，实力值较高的战队必定获胜，而失败的战队将被淘汰。对于决赛相遇的两支战队，获胜的战队被认定为冠军，而另一支战队被认定为亚军。

你的任务是找出假设的淘汰赛阶段中的冠军战队和亚军战队。

Input

输入包含八行，其中第 i 行包含一个字符串 S_i $(1 \le |S_i| \le 3)$ 和一个整数 t_i $(1 \le t_i \le 100)$，表示赛程表中八强从上到下第 i 支战队的名称和实力值。

保证所有战队的名称仅包含大写英文字母或十进制数码，且名称和实力值均互不相同。

Output

输出一行“`A beats B`”（不含引号），其中 `A` 是冠军战队的名称，`B` 是亚军战队的名称。

Examples

standard input	standard output
LNG 55 WBG 65 HLE 70 BLG 75 TES 48 T1 80 GEN 60 FLY 50	T1 beats BLG
LNG 55 WBG 65 HLE 70 BLG 81 TES 48 T1 80 GEN 60 FLY 50	BLG beats T1

Note

第二个样例中，由于我们在假设的淘汰赛阶段中刻意加大了 BLG 的实力值，请注意决赛结果，令人遗憾地，与实际 2024 英雄联盟全球总决赛有所出入。

Problem K. 易碎的弹球

给定一个有 n 条边的凸多边形，里面有一个非常小的弹球，可以看作一个点，它从多边形内部或边界的某个位置开始，以正的恒定速度沿直线运动。

你可以在任何时刻激活凸多边形的一条边一瞬间。如果弹球在那个时刻恰好位于激活的边上（包括端点），它将被该边反弹。也就是说，相对于边所在的直线，入射角等于反射角。由于边是易碎的，每条边最多只能激活一次。

需要注意的是，如果弹球同时位于两条边上，你不能同时激活这两条边。然而，你可以快速连续激活这两条边。在这种情况下，弹球将被反弹两次，弹球的移动方向也会改变两次，而其位置和速度保持不变。你也可以选择只激活其中一条边或不激活任何边。

由于弹球也是易碎的，它最多只能承受 k 次反弹。你需要对于每个 $k = 0, 1, 2, \ldots, n$ 求出弹球在凸多边形内可以行进的最长路程。

Input

第一行包含一个整数 n $(3 \le n \le 6)$，表示凸多边形的顶点数量。

接下来的 n 行，其中第 i 行包含两个整数 x_i 和 y_i $(-100 \le x_i, y_i \le 100)$，表示凸多边形第 i 个顶点的坐标。

保证这 n 个顶点是按逆时针顺序给出的，并且任意三个顶点不共线。

Output

输出 $(n+1)$ 行，其中第 i 行包含一个实数，表示弹球在 $i-1$ 次反弹内可以行进的最长路程。

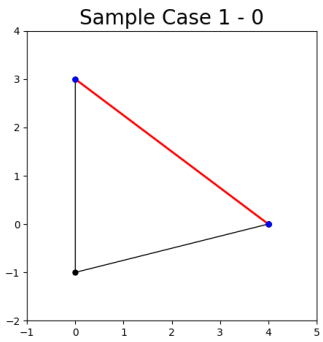
如果你的答案的绝对误差或相对误差不超过 10^{-6}，则将被视为正确。形式地说，假设你的输出为 a，标准答案为 b，当且仅当 $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$ 时，你的输出会被接受。

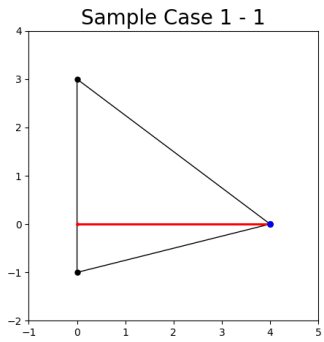
Examples

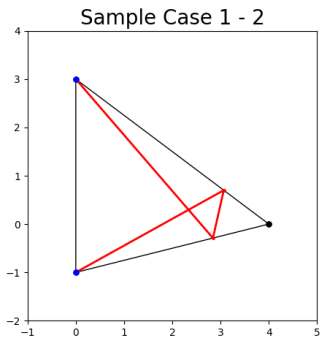
standard input	standard output
3 4 0 0 3 0 -1	5.000000000000000000 8.000000000000000000 8.868185038797563409 12.210024810881955830
3 4 0 0 3 0 2	5.000000000000000000 5.366563145999495272 6.111919138499425171 6.782203304416628317

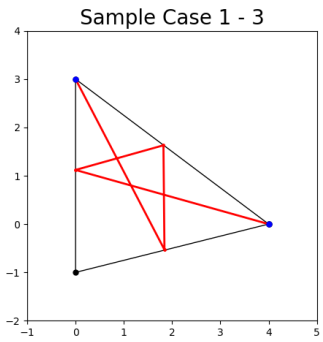
Note

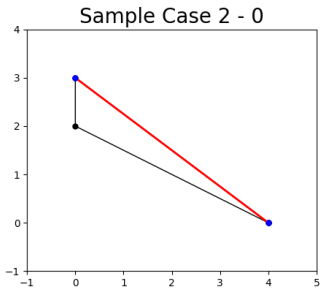
样例见下图。

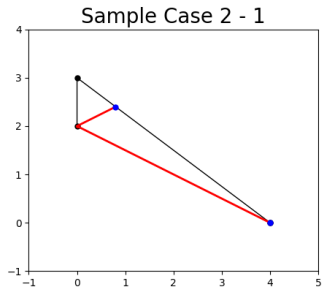

Sample Case 1 - 0

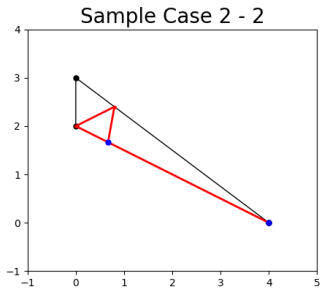

Sample Case 1 - 1

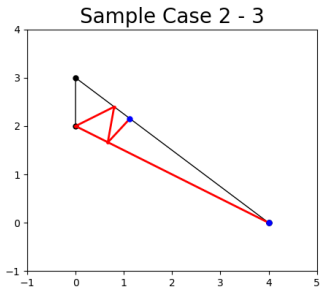

Sample Case 1 - 2


Sample Case 1 - 3


Sample Case 2 - 0


Sample Case 2 - 1

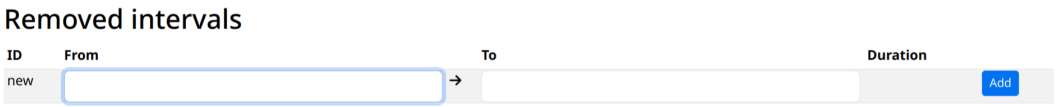

Sample Case 2 - 2


Sample Case 2 - 3

Problem L. 大奖赛

大奖赛中有着非常多的题目和非常长的比赛时间。队伍 1 和队伍 2 参加了这场比赛，共有 n 次提交。每次提交可以得到正确或错误的结果。如果一个队伍的提交是正确的，就认为对应的问题被该队伍解出。

比赛结束后，队伍根据解决的问题数量从多到少依次排名。解决相同数量问题的队伍按解题总时间从少到多依次排名。总时间是每个解决问题所消耗时间的总和。解决问题所消耗的时间是从比赛开始到第一次正确提交的时间，加上每个之前错误提交该问题的 p 分钟罚时。未解决的问题不计入总时间。在平局的情况下，队伍 1 排在队伍 2 之前。



大奖赛的裁判想测试一个名为 “移除区间” 的功能，该功能允许选择两个整数 L 和 R $(0 \le L < R)$，并从比赛中移除时间区间 $[L, R]$ 分钟，使得：

- 所有提交时间小于 L 分钟的提交不受影响；
- 所有提交在区间 $[L, R]$（单位：分钟）内的提交时间更改为从比赛开始后 L 分钟；
- 所有提交大于 R 分钟的提交时间减少 $(R - L)$ 分钟。

特别要注意的是，上述更改只会影响某些提交的提交时间，但不会影响提交的相对顺序。

你需要找到最短的时间区间 $[L, R]$（单位：分钟），使得在从比赛中移除该区间后，两个团队的排名会发生变化，或者指出这样的区间不存在。如果存在多个这样的区间，你需要找到 L 最小的那个。

Input

输入的第一行包含一个整数 T $(1 \le T \le 4 \times 10^5)$，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 $(1 \le n \le 4 \times 10^5)$ 和 p $(1 \le p \le 10^{12})$，表示提交数量和罚时分钟。

接下来 n 行按提交顺序描述比赛期间的提交。每行包含四个整数 a $(a \in \{1, 2\})$，b $(1 \le b \le 10^9)$，c $(1 \le c \le 10^{12})$ 和 d $(d \in \{0, 1\})$，表示团队 a 在比赛开始 c 分钟时在问题 b 做出提交，并得到了结果 d。如果 $d = 1$，则提交是正确的并解决了该问题；否则，提交是错误的。保证提交时间是单调不减的。

保证所有测试数据 n 的总和不超过 4×10^5。

Output

对于每组测试数据，输出一行。如果存在可行的时间区间，输出两个整数 L 和 R，表示 L 最小的最短区间。否则，输出 -1。

Example

standard input	standard output
2 6 20 1 1 60 0 2 2 60 0 2 2 120 1 1 2 180 1 1 2 180 0 2 2 300 1 2 20 1 1 300 1 2 2 300 1	120 160 -1

Problem M. 寂灭，而新生

谬翻阅着他的沉思录，寻得一张纸条被遗忘于泛黄的书页间。此纸条，缘于远方天狐座的灵韵，记载着一句似是自我激励的真言："梦若星辰，遥不可及，渺不足道，却足以将至暗点亮。"

只是时过境迁，谬终究得以瞥见梦隐秘的一面。经年累月的成长催使他更加老练，或许也将伤痕刻满了他的容颜，他难逃承认星辰与梦想，在平等支配万物的宿命论之下，皆注定将归于寂灭。追随梦想的命途万念俱灰，怀疑现实的愁云笼罩心头，但谬拒绝向悲观妥协，他选择等待，至今仍在等待，怀揣着不渝的信念。他相信破碎之梦，恰如轮回盛典中的星辰余烬，必将重塑并再度燃起烈焰。

他继续回顾着沉思录中的点滴，头顶正是宇宙浩瀚无垠。的确，相较于有限，无限总能蕴藏着更多的憧憬与奥秘，无论在时空的领域，在梦想的星云，甚至 —— 在数学世界里。他的指尖应时对景地停留在了诉说希尔伯特旅馆悖论的一页，仿佛触及了这个无限集合经典范例曾牵动他思绪的瞬间。字里行间，谬构想了一个升级版的无限旅馆 —— 天界希尔伯特旅馆 —— 并亲自扮演其馆长，留下一个有趣的问题供像你这样的程序员来解决。

希尔伯特的无限旅馆

作为潜匿深空的奇观，天界希尔伯特旅馆拥有无限多个房间。每个房间都被分配了一个独特的整数（可以是正数、负数或零）作为其房号，且所有的房号构成 $\mathbb{Z}$，即全体整数集。

旅馆有 n 个楼层，房间根据其房号除以 n 后的非负余数分布在不同楼层。具体地，房间 x 和房间 y 处于同一楼层当且仅当它们的房号除以 n 后余数相同，用同余方程表示即 $x \equiv y \pmod{n}$。

和其他典型的希尔伯特旅馆一样，题述版本也通过特定的搬迁指令来腾空已占用的房间。谬设计了 m 条搬迁指令。每条指令表示为 (a, b)，能引导所有与房间 a 处于同一楼层的旅客同时搬入新的房间，新房号通过原房号加上 b 来确定。必要时，谬可以选择 m 条指令中的一条将其广播到相应的楼层。你可能会想到当旅客被要求搬入已占用的房间时会发生什么。在这种情形下，正在搬迁的旅客仍会搬入，永远陪伴着原住户。

然而，由于天界希尔伯特旅馆的无限性，谬很难追踪一位旅客经过多次指令后的潜在位置，导致管理效率低下。因此，谬提出了 q 组询问，每组询问考察了一位初始入住房间 x 的旅客。假设搬迁指令可以以任意顺序执行任意次数，具体次数和顺序均不确定，对于每组询问，你的任务是判断所考察的旅客所有可能到达房间的房号集合是否为无限集。形式化地，令 S_k 为该旅客经过至多 k 条指令后可能到达房间的房号集合，你需要判断对于所有整数 t，是否都存在 k 使得 $|S_k| > t$ 成立，其中 $|S_k|$ 表示 S_k 的大小。

Input

第一行包含三个整数 n，m 和 q $(1 \le n, m, q \le 5 \times 10^5)$，分别表示楼层个数、搬迁指令个数和询问组数。

接下来 m 行，其中每行包含两个整数 a 和 b $(-10^9 \le a, b \le 10^9)$，描述一个搬迁指令 (a, b)。

接下来 q 行，其中每行包含一个整数 x $(-10^9 \le x \le 10^9)$，表示一组考察初始入住房间 x 的旅客的询问。

Output

对于每组询问，如果所考察的旅客所有可能到达房间的房号集合为无限集，在一行内输出 "`Yes`"（不含引号），否则在一行内输出 "`No`"（不含引号）。

Examples

standard input	standard output
3 2 3 1 1 -1 3 1 2 3	Yes Yes No
3 2 3 1 1 -1 0 1 2 3	No No No