第三届环球杯

第 10 赛段：西湖

2024 年 9 月 28-29 日

该问题集应包含 13 个问题（A 至 M），共 21 页。

问题 A. 意大利美食

时间限制 1 秒 内存限制 1024 兆字节

包包为您准备了一个披萨！这个披萨是一个凸多边形，它的所有边缘都有馅饼皮。不过，饼皮很脆弱，只能从顶点切开，不能切到边缘中间。不幸的是，披萨上有一块相当大的圆形菠萝，你绝对要避开它。

计算一次直切可以得到的最大一块无菠萝披萨，并输出其大小。当菠萝的任何部分都没有严格落在一块披萨上时，这块披萨就被认为是无菠萝披萨。也就是说，菠萝和披萨的交点面积为 0。

输入

有多个测试用例。输入的第一行包含一个整数 T，表示测试用例的数量。对于每个测试用例

第一行包含一个整数 n（$3 \le n \le 10^5$），表示披萨的顶点数。

第二行包含三个整数 x_c, y_c, 和 r（$-10^9 \le x_c, y_c \le 10^9, 1 \le r \le 10^9$），表示菠萝的中心坐标和半径。

对于下面的 n 行，第 i 行包含两个整数 x_i 和 y_i（$-10^9 \le x_i, y_i \le 10^9$），表示第 i 个顶点的坐标。顶点按逆时针顺序排列。没有两个点是重合的。但是可能有三个点位于同一条直线上。

保证菠萝的任何部分都不会超出披萨的边界。同时保证所有测试用例的 n 之和不超过 10^5。

输出

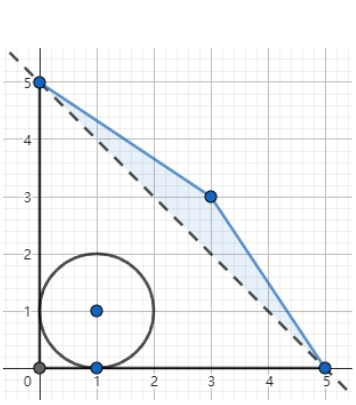
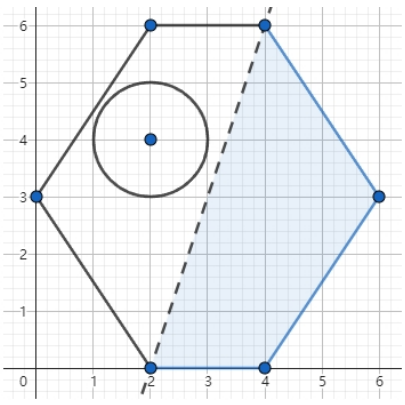
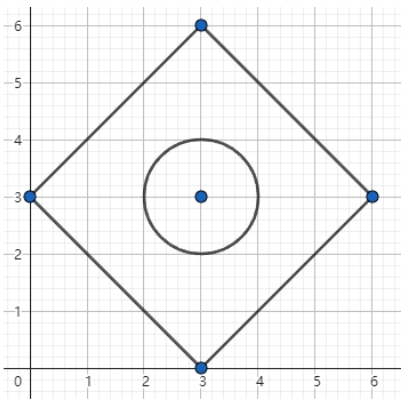
对于每个测试案例，输出一行包含一个整数的数据，表示最大一块无菠萝披萨乘以 2 的大小。如果无法得到无菠萝披萨，则输出 0。

示例

标准输入	标准输出
3 5 1 1 1 0 0 1 0 5 0 3 3 0 5 6 2 4 1 2 0 4 0 6 3 4 6 2 6 0 3 4 3 3 1 3 0 6 3 3 6 0 3	5 24 0

备注

测试用例示例如下。

问题 B. 生成字符串

时间限制 3 秒 内存限制 1024 兆字节

您 你们 给您 a 模板 字符串 $S = S\,S_{12}\,\text{-}\,\text{-}\,\text{-}\,S_n$ 的长度 n. A 生成的字符串 是由 S 的多个子串连接而成的字符串。

$T = f(k, \{l\}_i^k, \{r\}_i^k)$ 由正整数 k 和 k 对整数 (l_i, r_i) 描述，其中 $T = s[l_1 : r_1] + s[l_2 : r_2] + \text{-}\,\text{-}\,\text{-} + s[l_k : r_k]$。这里 $s[l : r]$ 表示子串 $s\,s_{ll+1}\,\text{-}\,\text{-}\,s_r$，+ 表示字符串连接。

你的任务是维护一个字符串多集合 A，支持以下三种类型的操作：

- $+\,k\,l_1\,r_1\,l_2\,r_2\,\text{-}\,\text{-}\,\text{-}\,l_k\,r_k$：将 $f(k, \{l\}_i^k, \{r\}_i^k)$ 插入多集合 A。
- $-\,t$：从多集合 A 中擦除*第* t 次操作中插入的字符串。
  第 t *次*操作是插入操作，此时插入的字符串不会被擦除。
- $?\,k\,l_1\,r_1\,l_2\,r_2\,\text{-}\,\text{-}\,\text{-}\,l_k\,r_k\,m\,u_1\,v_1\,u_2\,v_2\,\text{-}\,\text{-}\,\text{-}\,u_m\,v_m$：回答多集合 A 中字符串的个数以字符串 $f(k, \{l\}_i^k, \{r\}_i^k)$ 开头，以 $f(m, \{u\}_i^m, \{v\}_i^m)$ 结尾。

输入

每个测试文件中只有一个测试用例。

第一行包含两个整数 n 和 q（$1 \le n, q \le 10^5$），表示 S 的长度和运算次数。

第二行包含一个字符串 $s\,s_{12}\,\text{-}\,\text{-}\,s_n$，由小写英文字母组成，表示模板字符串。

对于以下 q 行，*第* i 行包含上述格式的操作。保证 $1 \le l_i \le r_i \le n$，$1 \le u_i \le v_i \le n$。还保证在所有类型的操作中，k 的总和是 +，加上所有"..... "类型运算中的 k 之和，加上所有"..... "类型运算中的 m 之和，不会超过 3×10^5。

输出

对于每个 ? 类型的操作，输出一行包含一个整数的答案。

示例

标准输入	标准输出
8 7 abcaabbc + 3 1 3 2 4 3 8	2 1

+ 2 1 4 1 8 + 1 2 4 ?1 5 6 1 7 8 - 3 + 1 2 5 ?1 2 3 1 5 5	

问题 C. 排列

时间限制 1 秒 内存限制 1024 兆字节

这是一个互动问题。

n 有一个隐藏的排列组合。回想一下，n 的排列是一个序列，其中的每个整数都来自 1 到 n（包括这两个数字）正好出现一次。小猪想通过一些查询来解开这个排列组合。

每个查询必须由一个序列（不一定是排列）和 n 个整数组成，整数范围从 1 到 n（均包括在内）。每个查询都会得到一个整数 x，表示小猪查询序列中相应元素与隐藏排列相匹配的位置数。例如，如果隐藏排列是 $\{1, 3, 4, 2, 5\}$，而小猪的询问序列是 $\{2, 3, 5, 2, 5\}$，那么他将得到 3 作为答案。

小猪最近很忙，他把这个问题交给了你。找出不超过 6666 的排列组合
询问。

输入

每个测试文件中只有一个测试用例。

输入的第一行包含一个整数 n（$1 \le n \le 10^3$），表示隐藏排列的长度。

互动协议

查询时，输出一行。首先输出 0，然后用空格隔开，接着打印一连串从 1 到 n 的整数，用空格隔开。刷新输出后，程序将读取一个表示查询答案的整数 x。

如果您想猜测排列组合，请输出一行。首先输出 1，然后用空格隔开，接着打印 n 的排列组合，用空格隔开。刷新输出后，程序应立即退出。

请注意，每个测试用例的答案都是预先确定的。也就是说，交互器不是自适应的。还请注意，您的猜测不算是查询。

要清除输出，可以使用

- 在 C 和 C++ 中，fflush(stdout)（如果使用 printf）或 cout.flush()（如果使用 cout）。
- Java 中的 System.out.flush()。
- 在 Python 中使用 stdout.flush()。

示例

标准输入	标准输出
5	

3 4 2	0 3 1 3 2 2 0 3 1 5 2 2 0 3 5 4 4 4 1 3 1 5 2 4

备注

请注意，如果收到 "超过时间限制 " 的判决，有可能是您的查询无效或查询次数超过限制。

问题 D. 收集硬币

时间限制 1 秒 内存限制 1024 兆字节

一排有 10 个[9] 牢房，从左到右编号为 1 至 10^9。两个机器人在小区内巡逻。每个机器人的最大速度为每秒 v 个单元（v 为整数），表示如果一个机器人当前在 p 单元那么只要 $1 \le p' \le n$ 且 $1 \le p' \le n$，它就可以在下一秒移动到任何一个单元 p'。
$|p' - p| \le v$.
单元格中会出现 n 枚硬币。第 i 枚硬币将在第 t_i 秒出现在第 c_i 个单元格中。如果此时有机器人在同一单元格中，它就可以拾起这枚硬币。否则，硬币会立即消失。

更正式地说，在每一秒钟内，以下步骤将依次发生：

- 每个机器人可以移动到不超过 v 个单元的位置（留在当前单元也可以）。
- 单元格中会出现硬币。
- 如果至少有一个机器人与一枚硬币在同一小格内，则该硬币会被收集起来。
- 所有未收集的硬币都会消失。

你的任务是在第 1 秒之前确定两个机器人的初始位置，并明智地移动它们，以尽可能小的 v 收集所有硬币。

输入

有多个测试用例。输入的第一行包含一个整数 T，表示测试用例的数量。对于每个测试用例

第一行包含一个整数 n（$1 \le n \le 10^6$），表示单元格中出现的硬币数量。

对于下面的 n 行，第 i 行包含两个整数 t_i 和 c_i（$1 \le t_i, c_i \le 10^9$），表示第 i 枚硬币出现的时间和第 i 枚硬币出现的位置。对于所有 $1 \le i < n$，保证 $t_i \le t_{i+1}$。对于所有 $i \neq j$，也保证 $t_i \neq t_j$ 或 $c_i \neq c_j$。

保证所有测试用例的 n 之和不超过 10^6。

输出

对于每个测试案例，输出一行包含一个整数的数据，表示机器人可能达到的最小最大速度。如果不可能收集到所有硬币，则输出-1。

示例

标准输入	标准输出

3 5 1 1 3 7 3 4 4 3 5 10 1 10 100 3 10 100 10 1000 10 10000	2 0 -1

问题 E. 普通朋友

时间限制 8 秒 内存限制 1024 兆字节

段树是小 Q 非常喜欢的一种数据结构。它们结构简单、时间复杂度高、功能强大，所以小Q曾经花了很长时间研究段树的一些特性。

最近，小 Q 又开始研究线段树了，但与以往不同的是：她把重点放在了更广义的线段树上。在普通的分段树中，对于一个区间 $[l, r]$，我们取 $\text{mid} = \frac{l+r}{2}$，然后把这个区间分成 $[l, \text{mid}]$ 和 $[\text{mid}+1, r]$。在广义分段树中，mid 并不要求正好是区间的中点，但 mid 仍必须满足 $l \le \text{mid} < r$。

小 Q 不知道如何实现平衡 BST，所以她想用段树来实现平衡 BST 的所有操作。其中最有名的一个不能用段树实现的操作是反转区间，但小 Q 不服气，想用段树实现它。

具体来说，当小 Q 扭转一个区间时，她会先把这个区间划分为几个最大区间，由分段树中的节点代表；假设从小到大，它们分别由分段树中的 k 个节点 $a_1, a_2, \ldots, a_k$ 代表。然后，小 Q 将剥离这 k 个子树，并以相反的顺序将它们重新连接到段树上，也就是说，原来位于 $_1$ 位置的子树将被替换为 $_k$ 的子树，原来位于 $_2$ 位置的子树将被替换为 $_{k-1}$ 的子树，以此类推；原来位于 $_k$ 位置的子树将被替换为 $_1$ 的子树。最后，小 Q 将交换这 k 个子树中所有节点的左右子节点。我们不难发现，经过这样的操作后，这棵树仍然是一棵广义段树，元素从左到右的顺序正好是颠倒区间后的结果。

给定这样一棵广义段树，小 Q 需要执行 m 次操作；每次给定一个前缀 $[1, x]$，她都要将其反转。同时，小 Q 想知道 k 的值，即在反向操作过程中，当前广义段树划分的区间数。

当然，小 Q 很轻松地就解决了这个问题，这让她更有信心不学习如何写平衡 BST。那么，你能解决这个问题吗？

输入

第一行包含两个正整数 n 和 m（$2 \le n, m \le 3 \times 10^5$），分别代表树的大小和操作次数。

下一行包含 $n - 1$ 个正整数，以深度优先搜索（DFS）的顺序给出，为每个区间提供分割点中值。

接下来的 m 行都包含一个正整数 x，表示将前缀 $[1, x]$ 反过来的操作。

输出

输出 m 行，每行包含一个正整数，代表每次操作过程中的 k 值。

示例

标准输入	标准输出
3 3 2 1 2 3 2	1 1 2

问题 F. 三角形

时间限制 1 秒 内存限制 1024 兆字节

给定由小写英文字母组成的 n 个字符串 $S_1, S_2, \cdots, S_n$，我们说两个字符串 S_a, S_b 和 S_c 构成三角形，如果满足以下所有约束条件：

- $S_a + S_b > S_c$ 或 $S_b + S_a > S_c$。
- $S_a + S_c > S_b$ 或 $S_c + S_a > S_b$。
- $S_b + S_c > S_a$ 或 $S_c + S_b > S_a$。

这里的 + 是字符串连接操作，字符串按词典顺序进行比较。例如

因为 ba、cb 和 cbaa 构成三角形：

- cb + ba = cbba > cbaa。
- cbaa + ba = cbaaba > cb。
- cb + cbaa = cbcbaa > ba。

数出 $1 \le a < b < c \le n$ 且 S_a, S_b, S_c 构成三角形的整数元组 (a, b, c) 的个数。

输入

有多个测试用例。输入的第一行包含一个整数 T，表示测试用例的数量。对于每个测试用例

第一行包含一个整数 n（$1 \le n \le 3 \times 10^5$），表示字符串的数量。

对于下面的 n 行，第 i 行包含一个字符串 S_i ($1 \le |S_i| \le 3 \times 10^5$)，由小写英文字母组成。

保证单个测试用例中的字符串总长度不超过 3×10^5，所有测试用例的字符串总长度不超过 10^6。

输出

对于每个测试用例，输出一行，其中包含一个整数，表示有效元组的数量。

示例

标准输入	标准输出

3 6 cbaa cb cb cbaa ba ba 3 SDC PC SD CPC 1 ccpc	16 0 0

问题 G. 阻止城堡 2

时间限制 1 秒 内存限制 1024 兆字节

在一个有 10^9 行和 10^9 列的棋盘上有 n *个*城堡和 m 个障碍。每个城堡或障碍物正好占据一个单元格，所有被占据的单元格都是不同的。如果两个城堡位于同一行或同一列，并且它们之间没有障碍物或其他城堡，它们就可以互相攻击。更正式地说，让（i, j）成为*第* i 行和*第* j 列上的单元格。如果以下条件之一为真，位于 (i_1, j_1) 和 (i_2, j_2) 的两个城堡可以互相攻击：

- $i_1 = i_2$，并且对于所有 $\min(j_1, j_2) < j < \max(j_1, j_2)$ 的情况，在 (i_1, j) 处没有障碍物或城堡。
- $j_1 = j_2$，并且对于所有 $\min(i_1, i_2) < i < \max(i_1, i_2)$，在 (i, j_1) 处没有障碍物或城堡。

您必须清除棋盘上的 k *个*障碍，但又不想让太多的城堡互相攻击。在清除棋盘上的 k *个*障碍物后，尽量减少可以互相攻击的城堡对的数量。

输入

有多个测试用例。输入的第一行包含一个整数 T，表示测试用例的数量。对于每个测试用例

第一行包含三个整数 n、m 和 k（$1 \le n$，$m \le 10^5$，$1 \le k \le m$），分别表示城堡数量、障碍物数量以及您需要清除的障碍物数量。

对于下面的 n 行，第 i 行包含两个整数 r_i 和 c_i（$1 \le r_i, c_i \le 10^9$），表示*第* i *个*城堡位于 r_i -*th* row 和 c_i -th column 上。

对于下面的 m 行，*第* i 行包含两个整数 r^r 和 c^r（$1 \le r^r_i, c^r_i \le 10^9$），表示*第* i *个*障碍物位于第 r^r_i -行和第 c^r_i -列。

可以保证被占用的单元格是不同的。此外，还能保证所有测试用例的 n 和 m 之和*都*不会超过 10^5。

输出

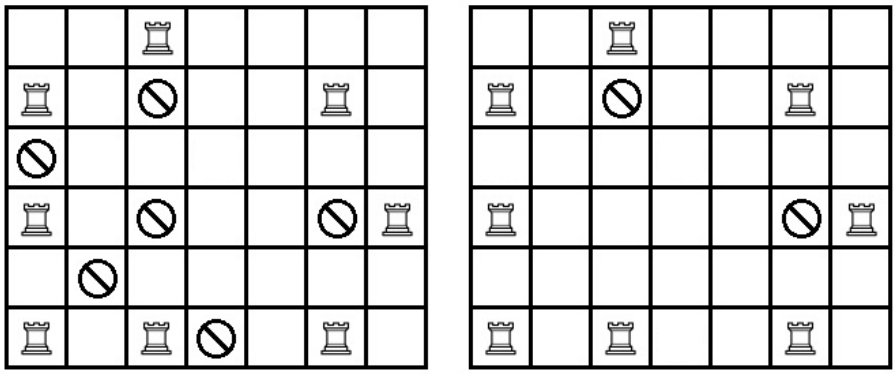
对于每个测试案例，首先输出一行，其中包含一个整数，表示在移除整整 k *个*障碍物后可以互相攻击的城堡对的最小数量。然后再输出一行，其中包含 k *个*不同的整数 $b_1, b_2, \ldots, b_k$（$1 \le b_i \le m$），中间用空格隔开，表示要清除的障碍物的索引。如果有多个有效答案，您可以输出其中任何一个。

示例

标准输入	标准输出
3 8 6 4 1 3 2 1 2 6 4 1 4 7 6 1 6 3 6 6 2 3 3 1 4 3 4 6 5 2 6 4 3 2 1 10 12 10 10 10 11 1 4 1 5 1 3 2 1 1 2 1 2 2 2 3	4 6 3 2 5 2 1 0 1 2

备注

在第一个测试案例中，左边的图片显示的是原始棋盘，右边的图片显示的是移除 4 个障碍后的棋盘。清除障碍后，可以互相攻击的棋子对为：第 2 和第 4 个棋子、第 4 和第 6 个棋子、第 6 和第 7 个棋 子 、第 7 和第 8 个棋子。



在第三个测试案例中，由于只有一个城堡，所以没有一对城堡可以互相攻击。

问题 H. 路径交叉

时间限制 5 秒 内存限制 1024 兆字节

有一棵有 n 个顶点和 $(n-1)$ 条边的树，其中第 i 条边连接顶点 u_i 和 v_i，权重为 w_i。

您的任务是处理 q 个查询。第 i 个查询可以用三个整数 a_i, b_i 和 k_i 来描述。这个查询会暂时将 a_i -th 边的权重改为 b_i。然后，您应该选择 $2k_i$ 个不同的顶点 $s_1, s_2, \ldots, s_{k_i}, e_1, e_2, \ldots, e_{k_i}$ 并考虑树上的 k_i 条简单路径，其中第 p 条路径从顶点 s_p 开始，在顶点 e_p 结束。如果一条边包含在所有 k_i 路径中，我们就说这条边是好边。最大化好边的总权重。

请再次注意，每次查询的权重变化都是暂时的。每次查询后，都应重新更改权重。

输入

每个测试文件中只有一个测试用例。

第一行包含两个整数 n 和 q（$2 \le n \le 5 \times 10^5$，$1 \le q \le 5 \times 10^5$），表示顶点数和查询次数。

对于下面的 $(n-1)$ 行，第 i 行包含三个整数 u_i, v_i 和 w_i（$1 \le u_i, v_i \le n$）、$1 \le w_i \le 10^9$）表示第 i 条边连接顶点 u_i 和 v_i，权重为 w_i。

对于以下 q 行，第 i 行包含三个整数 a_i、b_i 和 k_i（$1 \le a_i \le n-1$，$1 \le b_i \le 10^9$，$1 \le k_i \le \lfloor \frac{n}{2} \rfloor$），表示第 i 次查询。

输出

对每个查询输出一行，其中包含一个表示答案的整数。

示例

标准输入	标准输出
7 3 1 2 20 2 3 10 2 4 40 4 6 10 1 5 30 5 7 10 2 100 1 5 50 2 2 100 3	160 110 20

备注

对于第一个查询，选择 $s_1 = 3$ 和 $e_1 = 7$。

对于第二个查询，选择 $s_1 = 4$、$s_2 = 6$、$e_1 = 7$ 和 $e_2 = 5$。

对于第三个查询，选择 $s_1 = 3$、$s_2 = 4$、$s_3 = 6$、$e_1 = 5$、$e_2 = 1$ 和 $e_3 = 7$。

问题 I.找到自己

时间限制	1.5 秒 内存限
制	1024 兆字节

量子怪兽是一种奇特的生物，它以波函数的形式同时存在于不同的世界线中。只要有一条可能的世界线没有发现量子怪兽，你就永远无法捕捉到它。

现在，在一个有 n 个节点和 m 条边的连通无向图中隐藏着一个量子怪兽。你希望捕获它。整个过程如下循环进行：

1. 量子怪沿着图中的一条边移动到相邻节点。
2. 你在图中选择一些节点观察一次。您可以知道量子怪兽是否在您观察到的节点集中。
3. 如果您掌握的历史信息足以唯一确定量子怪的位置，您就成功捕获了它，循环结束。否则，返回步骤 1。

你想知道是否有可能捕获这个量子怪兽。换句话说，要确定是否存在一种捕捉策略，无论量子怪的初始位置和移动计划如何，你都能在有限的步数内唯一地确定它的位置。

输入

第一行包含一个整数 T（$1 \le T$），表示测试用例的数量。

对于每个测试用例，第一行包含两个整数 n 和 m（$2 \le n \le 10^6$，$1 \le m \le 10^6$），表示图中节点和边的数量。

接下来的 m 行分别包含两个整数 u 和 v（$1 \le u, v \le n$），表示图中存在一条连接 u 和 v 的无向边。

保证所有图都是连通的无向图，没有自循环或多条边，所有测试用例的 m 之和不超过 10^6。

输出

对于每个测试用例，如果存在捕获策略，则输出包含字符串 "YES "的一行；如果不存在，则输出包含字符串 "NO "的一行。

示例

标准输入	标准输出
3 3 3 1 2 2 3 3 1 4 4 1 2 2 3 3 4 4 1 6 6 1 2 2 3 3 4 4 5 5 6 6 1	没有 是否

问题 J. Sheriruth

时间限制 5 秒 内存限制 1024 兆字节

给你一个简单有向图 $G = (V, E)$，其中 $V = \{0, 1, 2, .$如果 a、b、$c \in V$ 满足以下所有限制条件：

- $b \neq c$
- $(a \to b) \in E$
- $(a \to c) \in E$
- $(b \to c) \notin E$

然后，我们会将边 $b \to c$ 加入边集 E。

不断进行这样的操作，直到做不出来为止。我们可以证明最终的图 $G^r = (V, E^r)$ 是唯一的。

然后，我们会问你 q 个问题。对于每个问题，我们都会给出节点 u、$v \in V$。你需要回答在 G^r 上有多少条从 u 到 v 的路径，并且在这些路径上我们不会与任何节点相遇超过一次。

由于答案可能太大，您只需告诉我们答案的模 S。

输入

输入的第一行包含四个整数 n、m、q 和 S（$1 \le n \le 5 \times 10^5$，$0 \le m \le 10^6$，$1 \le q \le 10^6$，$1 \le S \le 2^{30}$）。

每行包含两个整数 u 和 v（$0 \le u, v \le n - 1, u \neq v$），表示 E 中有一条边 $u \to v$。

下面的 q 行描述了所有问题。每行包含两个整数 u 和 v（$0 \le u, v \le n-1$）。

输出

输出 q 行。其中第 i 行包含一个整数，表示答案模数 S。

示例

标准输入	标准输出
11 9 30 998244353 0 1 0 2 3 4 5 4 6 5 7 8 8 9 9 8 10 9 0 1 0 2 1 0 1 2 2 0 2 1 3 4 3 5 3 6 4 3 4 5 4 6 5 3 5 4 5 6 6 3 6 4 6 5 7 8 7 9 7 10 8 7 8 9 8 10 9 7 9 8 9 10 10 7 10 8 10 9	2 2 0 1 0 1 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 0 0 1 0 0 1 1

问题 K. 正多边形

时间限制 1 秒 内存限制 1024 兆字节

有一个有 n 个顶点的凸多边形。顶点按逆时针顺序从 1 编号到 n（包括 n），顶点 i 的值为 $f(i)$。

如果顶点的值按逆时针顺序排列构成了一个重合顶点，我们就说这个顶点子集是重合的。更正式地说，假设子集包含 k 个顶点 $v_0, v_1, \ldots, v_{k-1}$，按逆时针顺序排列。应该存在一个整数 d，使得 $0 \le d < k$，并且对于所有 $0 \le i < k$，我们有 $f(v_{(d+i) \bmod k}) = f(v_{(d-1-i) \bmod k})$。

在所有宫调子集中，找出凸面最大的子集。

输入

有多个测试用例。输入的第一行包含一个整数 T，表示测试用例的数量。对于每个测试用例

第一行包含一个整数 n（$3 \le n \le 500$），表示凸多边形的顶点数。第二行包含 n 个整数 $f(1), f(2), \ldots, f(n)$（$1 \le f(i) \le 10^9$），其中 $f(i)$ 是凸多边形的值。
第 i 个顶点。

对于下面的 n 行，第 i 行包含两个整数 x_i 和 y_i （$-10^9 \le x_i, y_i \le 10^9$），表示第 i 个顶点的坐标。顶点按逆时针顺序排列。凸多边形的大小保证为正，且没有两个顶点重合。但是可能有三个顶点位于同一条直线上。

保证所有测试用例的 n 之和不超过 10^3。

输出

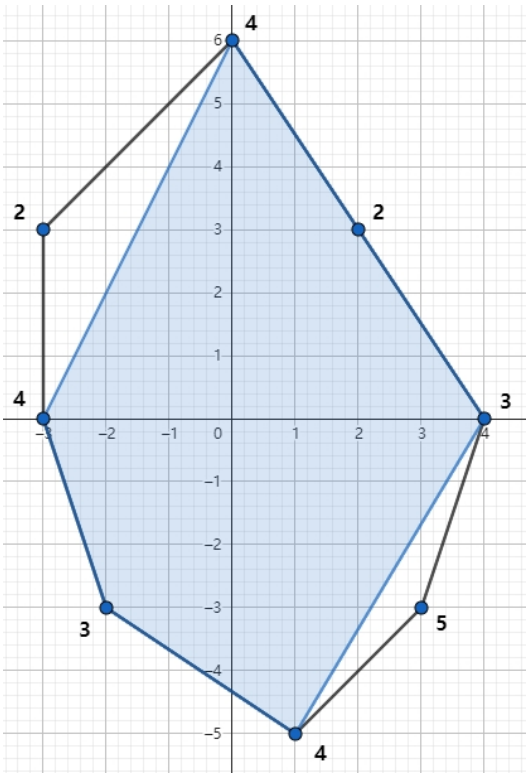
对于每个测试用例，输出一行包含一个整数的数值，表示一个 palindromic 子集的最大凸壳的大小乘以 2。

示例

标准输入	标准输出
3 8 2 4 2 4 3 4 5 3 2 3 0 6 -3 3 -3 0 -2 -3 1 -5 3 -3 4 0 3 1 2 3 0 0 1 0 0 1 3 1 1 1 0 0 1 0 0 1	84 0 1

备注

第一个测试案例示例如下。选择顶点 2、4、5、6、8，并考虑 $d = 1$，那么值序列 $\{4, 3, 4, 3, 4\}$ 是一个回文。



问题 L.宇宙旅行

时间限制　　2 秒 内存限制　　1024 兆字节

包包是一个宇宙旅行者，穿梭于无数个平行宇宙之间。每个宇宙都有一个从 0 开始的整数编号。

每个宇宙都有 n 个神奇苹果。虽然这些宇宙有许多相似之处，但它们之间仍有细微差别。第 i 个苹果在第 j 个宇宙中的魔力是 $i \oplus j$。这里，⊕表示位排他性或运算。

小宝非常优柔寡断，所以他准备了 q 个旅行计划。每个旅行计划都可以用三个整数 l、r 和 k 来描述，即小宝将前往从 l 到 r（包括 l 和 r）的每个宇宙，并收集每个宇宙中 n 个苹果中魔力最小的 k 个苹果。

在每个旅行计划中，请帮助包包计算他收集到的苹果的魔力总和。请注意，旅行计划并没有真正从每个宇宙中取出苹果。也就是说，每个查询都是独立的。

输入

每个测试文件中只有一个测试用例。

第一行包含两个整数 n 和 q（$1 \le n$，$q \le 10^5$），表示每个宇宙中的苹果数量和计划数量。

第二行包含 n 个整数 $a_1, a_2, \cdots, a_n$（$0 \le a_i < 2^{60}$）。

对于以下 q 行，第 i 行包含三个整数 l_i、r_i 和 k_i（$0 \le l_i \le r_i < 2^{60}$，$1 \le k_i \le n$），表示第 i 个行进计划。

输出

对每个行进计划输出一行，其中包含一个整数，表示该计划的答案。由于答案可能很大，因此输出它的模数 998244353。

示例

标准输入	标准输出
8 3 2 0 2 4 0 5 2 6 1 1 6 2 7 5 0 1048575 4	4 23 720895450

问题 M. 寻找埃尔多拉多

时间限制 1 秒 内存限制 1024 兆字节

一个王国有 n 座城市，连接城市的双向铁路有 m 条。第 i *条*铁路由 c_i -th 铁路公司运营，铁路长度为 l_i。

您想从城市 1 开始环游全国。您购买了 k *张*火车票。*第* i *张*车票可以用两个整数 a_i 和 b_i 表示，也就是说，如果您使用这张车票，您可以一次性乘坐几条铁路，只要这些铁路都是由 a_i 公司运营的，并且总长度不超过 b_i。使用车票时，也允许只停留在当前城市。每次只能使用一张车票，每张车票只能使用一次。

由于您觉得确定票据的使用顺序是个负担，所以决定只按当前顺序使用票据。更正式地说，你要执行 k *次*操作。在*第* i *次*操作中，你可以选择留在当前城市 u；或者选择另一个城市 v，这样就存在一条从城市 u 到城市 v 的路径，路径上的所有铁路都由 a 公司运营$_i$，且铁路总长度不超过 b_i，最后移动到城市 v。

对于每个城市，确定在使用所有 k *张*机票后是否有可能到达该城市。

输入

有多个测试用例。输入的第一行包含一个整数 T，表示测试用例的数量。对于每个测试用例

第一行包含三个整数 n、m 和 k（$2 \le n \le 5 \times 10^5$，$1 \le m \le 5 \times 10^5$，$1 \le k \le 5 \times 10^5$），表示城市数量、铁路数量和车票数量。

对于以下 m 条线路，第 i *条*线路包含四个整数 u_i, v_i, c_i, 和 l_i（$1 \le u_i, v_i \le n, u_i \neq v_i, 1 \le c_i \le m, 1 \le l_i \le 10^9$），表示*第* i *条*铁路连接城市 u_i 和 v_i。它由 c 公司运营$_i$，长度为 l_i。请注意，可能有多条铁路连接同一对城市。

对于下面的 k 条线路，第 i *条*线路包含两个整数 a_i 和 b_i（$1 \le a_i \le m$，$1 \le b_i \le 10^9$），表示如果使用*第* i 张车票，可以一次性通过一些铁路，如果这些铁路都由 a 公司运营$_i$，且总长度不超过 b_i。

保证所有测试用例的 n、m 和 k 之和不超过 5×10^5。

输出

对于每个测试用例，输出一行包含长度为 n 的字符串 $s\, s_{12}\, -\, -\, s_n$，其中每个字符要么为 0，要么为 1。如果用这 k *张*车票可以从城市 1 前往城市 i，则 $s_i = 1$；否则 $s_i = 0$。

示例

标准输入	标准输出
2 5 6 4 1 2 1 30 2 3 1 50 2 5 5 50 3 4 6 10 2 4 5 30 2 5 1 40 1 70 6 100 5 40 1 30 3 1 1 2 3 1 10 1 100	11011 100

备注

对于第一个示例测试用例：

- 要到达第 4 个城市，您可以使用第 1 张车票从第 1 个城市移动到第 2 个城市，然后在使用第 2 张车票时留在第 2 个城市，然后使用第 3 张车票从第 2 个城市移动到第 4 个城市，然后在使用第 4 张车票时留在第 4 个城市。
- 要到达第 5 个城市，您可以使用第 1 张车票从第 1 个城市移动到第 5 个城市，方法是通过 1 号和 6 号铁路，然后在使用以下车票时停留在 5 号城市。
- 由于不能更改机票的使用顺序，您无法到达第 3 个城市。