Problem A. 矩阵游戏

everflame 和 KisuraOP 在玩游戏。游戏开始时，everflame 会给出一个矩阵，而 KisuraOP 的目标是操作这个矩阵使自己获得尽可能多的得分。

everflame 给出的是一个 n 行 m 列的矩阵 $a_{i,j}$，每个位置初始可以为 0 或 1。此外，他还额外给出了两个整数 A 和 B。

KisuraOP 能够对这个矩阵进行任意次（可以是 0 次）操作，每次操作可以是以下两种中的一种：

- 选择一个 $i \in [1, n]$，将第 i 行的所有元素取反。
- 选择一个 $j \in [1, m]$，将第 j 列的所有元素取反。

其中，一个元素如果原来是 1，取反后将变成 0；反之如果原来是 0，取反后将变成 1。

对于矩阵中第 i 行第 j 列的元素 $a_{i,j}$，如果 $a_{i,j} = 1$，那么这个元素将会贡献 $(A \cdot i + B \cdot j)$ 的得分；否则这个元素的贡献为 0。矩阵的总得分将是所有 $n \times m$ 个元素的得分之和。

请帮 KisuraOP 求出经过一定的操作后这个矩阵能取得的最大得分。

Input

第一行包含四个整数 n, m, A, B（$1 \le n \le 10^6, 1 \le m \le 10, 0 \le |A|, |B| \le 10^6$）。

接下来输入 n 行，每行一个长为 m 的字符串。第 i 行第 j 列的字符代表 $a_{i,j}$ ($a_{i,j} \in \{0, 1\}$)。

Output

输出一行一个整数，即最大可能取值。

Examples

standard input	standard output
2 2 1 1 01 10	12
3 3 1 -5 010 000 010	-8
3 3 -3 -6 011 010 100	-24

Note

对于第一个样例，先操作第 1 行再操作第 2 列可以让矩阵全为 1，取得最大得分。得分为 $(1 \cdot 1 + 1 \cdot 1) + (1 \cdot 1 + 1 \cdot 2) + (1 \cdot 2 + 1 \cdot 1) + (1 \cdot 2 + 1 \cdot 2) = 12$。

对于第二个样例，仅操作第 2 列以取得最大得分，可以证明没有更优的方案。得分为 $1 \cdot 2 + (-5) \cdot 2 = -8$。

Problem B. 整数生成器

现有一个整数集合 S，你的任务是通过不超过 70 次位运算操作，利用 S 来生成一个给定的整数 x。

具体地，给定一个大小为 n 的整数集合 S 和一个整数 x。每次操作可以选择两个 S 中的整数 a 和 b（可以相同），将 a or b，$a \oplus b$ 和 a and b 中的一个整数插入到 S 中。你需要判断是否可以通过不超过 70 次操作使得 $x \in S$，若可以，你还需要给出一种合法的操作过程。

其中，a or b 指 a 和 b 的按位或，$a \oplus b$ 指 a 和 b 的按位异或，a and b 指 a 和 b 的按位与。

Input

第一行两个整数 n, x（$1 \le n \le 10^5, 0 \le x < 2^{30}$）。

第二行 n 个整数 $a_1, a_2, \ldots, a_n$（$0 \le a_i < 2^{30}$），表示最初 S 中的元素，保证这些整数两两不同。

Output

若无法通过不超过 70 次操作使得 $x \in S$，输出一行一个整数 -1。

否则，第一行输出一个整数 k（$0 \le k \le 70$），表示操作次数。

接下来 k 行依次输出这 k 次操作。对于每次操作输出一行三个整数 t, a, b（$t \in \{0, 1, 2\}$）。若 $t = 0$，则表示这次操作将 a or b 插入 S 中，若 $t = 1$，则表示将 $a \oplus b$ 插入 S 中，若 $t = 2$，则表示将 a and b 插入 S 中。你需要保证对于此次操作时的 S，有 $a \in S$ 且 $b \in S$。

本题中，你不需要最小化操作次数，如有多个满足条件的操作方案，输出任意一个均可。

Examples

standard input	standard output
3 7 1 2 4	2 1 1 2 1 3 4
3 15 9 10 4	2 0 10 9 1 4 11
3 7 1 2 3	-1

Problem C. 切牌

现有 n 张牌，编号为 1 到 n 的整数。一次切牌操作按如下步骤进行：

1. 将这些牌按编号从小到大的顺序从上到下放成一堆。
2. 选择一个正整数 k $(1 \le k \le n)$，将这些牌按从上到下的顺序依次分为 k 堆。你需要保证对于每堆牌，堆内至少有一张牌，牌的编号连续，并且按编号从小到大的顺序从上到下依次排列。
3. 将这些牌堆任意排列，并从左到右排成一排。
4. 每次按牌堆从左到右的顺序遍历牌堆，如果牌堆中还有牌，则抽出最上方的一张，放在已切好牌序列的末尾。
5. 如果所有牌都已进入已切好牌序列，则停止操作。

例如，对于五张牌 $\{1,2,3,4,5\}$，可以将其分成 $\{1\},\{2,3\},\{4,5\}$，$\{1\},\{2\},\{3\},\{4\},\{5\}$ 或 $\{1,2,3,4,5\}$，但不能将其分为 $\{1\},\{2,5\},\{3,4\}$，因为这违反了牌编号连续的原则，也不可以将其分为 $\{1\},\{3,2\},\{4,5\}$，因为这违反了编号从小到大排列的原则。

又例如，对于已经从左往右排列好的三堆牌 $\{1\},\{4,5\},\{2,3\}$，只会得到 $\{1,4,2,5,3\}$ 一种已切好牌序列，不可能得到 $\{1,2,4,5,3\}$，因为这违反了从左到右遍历的顺序，也不可能得到 $\{1,5,2,4,3\}$，因为这违反了从每堆最上方取牌的原则。

现有一个牌的目标排列，你需要计算有多少种不同的切牌操作可以得到这个目标排列。两种切牌操作不同，当且仅当牌的分堆方式不同或牌堆的排列方法不同。

这个目标排列可能会进行修改，对于每次修改，会交换目标排列中两个位置的元素。对于每次修改都需要输出答案。修改是持久化的，也就是说在此次修改之前的修改均会保留。

Input

第一行两个整数 n, Q（$2 \le n \le 10^5, 1 \le Q \le 10^5$）。

第二行 n 个整数 $a_1, a_2, \ldots, a_n$（$1 \le a_i \le n$），表示初始目标排列。

接下来 Q 行，每行两个整数 x, y（$1 \le x, y \le n, x \ne y$），表示修改中交换的两个位置。

Output

输出 $Q+1$ 行，第一行输出没有修改之前的答案，第二到第 $Q+1$ 行输出第一个修改到第 Q 个修改之后的答案。

Example

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

Note

样例中有 4 张牌，初始目标序列为 $\{1,3,2,4\}$。有如下三种切牌操作可以得到目标序列。

- $\{1,2\},\{3,4\}$
- $\{1\},\{3,4\},\{2\}$

- $\{1\},\{3\},\{2\},\{4\}$

对于第一次交换后，目标序列变为 $\{1,2,3,4\}$。有如下四种切牌操作可以得到目标序列。

- $\{1,2,3,4\}$
- $\{1\},\{2,3,4\}$
- $\{1\},\{2\},\{3,4\}$
- $\{1\},\{2\},\{3\},\{4\}$

Problem D. 生成魔咒

在字符串中隐藏各种禁忌词语来达到魔咒的效果，这一现象在魔法国屡见不鲜。于是为了更好地应对非法使用魔咒的案件发生，该国人民都需要学习魔咒学来帮助自己防御魔咒。

今天教学的内容是简便魔咒生成器，简便魔咒生成器有两个模式，**单击** 和 **长按**。

每次单击会花费 1 秒的时间，并且生成一段长度为 1 的魔咒。

每次长按需要选择一个正整数 x，然后再按 2^x 秒，这样可以生成一段长度为 10^x 的魔咒。

现在给你一些需要生成的魔咒的长度，你需要计算生成每个魔咒最少需要花多长时间。

Input

第一行输入一个整数 T（$1 \le T \le 50000$），表示需求总数。

接下来每行一个整数 r（$1 \le r \le 10^{18}$），表示每个要生成的魔咒长度。

Output

输出 T 行，每行表示需要花费的最小时间。

Example

standard input	standard output
5 23 61 62 114514 1919810	7 13 14 106 474

Problem E. 网格染色

现有一个 2 行 n 列的网格，行从上到下编号为 1 到 2，列从左到右编号为 1 到 n。有 $2n$ 种颜色，编号为 1 到 $2n$。目前一些格子已经被染色了，你需要对剩余格子进行染色（已经被染色的格子不能被重新染色），使得最后相同颜色组成的四连通块数量最少。请构造一种染色方案。

四连通是指，如果两个颜色相同的格子有一条公共边，那么我们认为这两个格子是连通的。

Input

第一行一个整数 n（$1 \le n \le 10^5$）。

第二行 n 个整数 $a_{1,1}, a_{1,2}, \ldots, a_{1,n}$（$0 \le a_{1,i} \le 2n$），表示网格第一行的染色情况。

第三行 n 个整数 $a_{2,1}, a_{2,2}, \ldots, a_{2,n}$（$0 \le a_{2,i} \le 2n$），表示网格第二行的染色情况。

如果 $a_{i,j} = 0$，则表示第 i 行第 j 列的单元格没有被染色，否则表示第 i 行第 j 列单元格的颜色为 $a_{i,j}$。保证最初至少有一个单元格满足 $a_{i,j} \neq 0$。

Output

输出两行，每行 n 个正整数 $b_{i,j}$（$1 \le b_{i,j} \le 2n$），表示一种染色方案。输出需保证已经被染色的格子不能被重新染色，即，对于 $a_{i,j} \neq 0$ 的单元格，输出应保证 $b_{i,j} = a_{i,j}$。

如有多种满足条件的染色方案，输出任意一种均可。

Examples

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

Problem F. 排名预测

你和你的队友们刚刚打完了一场 ICPC 竞赛，五个小时的时间已经耗尽了你的精力，并且队友把你的午餐吃了，你现在只能趴在桌子上看榜。目前还没有进行到颁奖仪式，因此榜还处于封榜状态中，也就是说，现在你知道你自己队伍在比赛全程的提交的时间和是否通过，而对于其他队伍，你知道他们进行每个提交的时间，并且你知道在封榜前其他队伍的每个提交是否通过，但你不知道在封榜后其他队伍的提交是否通过。

当你查看榜时，你发现了一个你很关注的队伍，你知道了他们队封榜前的每一次提交的时间与是否通过的状态，以及封榜后每一次提交的时间，你想知道他们队的排名会不会**严格高于**你们队。为了判断他们队排名严格高于你们队的可能性，你还想知道他们最少在封榜后过多少题，排名才会严格高于你们队。

在 ICPC 竞赛中，罚时按如下规则计算。假设比赛中**通过**了 m 道题目，编号为 1 到 m。对于通过的题目 i，首次通过题目的时间记为 t_i，在通过之前对这道题进行了 c_i 次提交，则罚时 p 按如下规则计算。

$$p = \sum_{i=1}^{m} t_i + 20 \cdot c_i$$

本题中不考虑编译错误导致不计罚时的特殊因素。

对于两支队伍 A 和 B，称 A 的排名严格高于 B，当且仅当 A 通过的题目数大于 B，或 A 与 B 通过题目数相等，且 A 的罚时小于 B 的罚时。

Input

第一行一个整数 T（$1 \le T \le 100$），表示数据组数。

对于每组数据，第一行三个整数 n, a, b（$10 \le n \le 15, 1 \le a \le n, 0 \le b \le 10^5$），分别表示比赛有 n 道题，你们队在比赛结束时通过了 a 道题，罚时为 b。

第二行一个整数 s（$0 \le s \le 10^3$），表示你关注的队伍在正常比赛中进行的提交数。

接下来 s 行，每行一个整数与两个字符串 t, p, v（$0 \le t < 300$）。表示在第 t 分钟对 p 题进行了一次提交，结果为 v。保证提交按时间从小到大的顺序给出（这意味着当 t 相同时，提交顺序是输入所给的顺序），p 为前 n 个大写英文字母，且 $v \in \{$`ac`$,$`rj`$,$`pd`$\}$。`ac` 表示这次提交通过，`rj` 表示提交未通过，`pd` 表示这次提交处于封榜状态中，不知道此次提交是否通过。保证当 $t < 240$ 时，v 不是 `pd`，且当 $t \ge 240$ 时，v 一定是 `pd`。

Output

对于每组数据，输出一行一个整数。如果你关注的队伍的最终排名不可能严格高于你们队，输出 -1，否则输出他们最少封榜后通过多少题，最终排名会严格高于你们队。

Example

standard input	standard output
1 11 6 900 13 11 C ac 34 J ac 52 D rj 61 D ac 193 A rj 207 A rj 220 G ac 245 A pd 247 A pd 262 H pd 299 A pd 299 C pd 299 K pd	2

Problem G. 货币系统

给定一个长度为 n 的升序数组 A，第 i 项的值为 A_i，其中 $A_1 = 1$。

用这个数组构建一个货币系统，对于任意正整数 x，定义换钱张数函数为 $f(x, n)$，其中 n 为数组长度。这个函数表示使用这一套货币系统支付 x 元时，如果要满足尽量先拿大面额纸币再拿小面额纸币的原则，需要拿出多少张纸币。对于任意正整数 x 和一个正整数 $y \in [1, n]$，$f(x, y)$ 满足：

$$f(x,y) = \begin{cases} \lfloor \frac{x}{A_y} \rfloor + f(x \bmod A_y, y-1) & y > 1 \\ x & y = 1 \end{cases}$$

你需要处理 q 组询问，每一组询问会给出一个整数 m，请回答有多少个正整数 x 满足 $f(x, n) = m$。

Input

输入第一行有两个正整数 n 和 q（$1 \le n \le 10^5, 1 \le q \le 10^6$），表示数组 A 的长度和询问次数。

第二行有 n 个正整数 $A_1, A_2, \ldots, A_n$（$1 = A_1 < A_2 < \ldots < A_n \le 10^6$），表示数组 A。

第三行有 q 个整数 $m_1, m_2, \ldots, m_q$（$1 \le m_i \le 10^9$），其中 m_i 表示第 i 次询问的整数。

Output

输出一行 q 个用空格隔开的整数，第 i 个整数表示第 i 次询问的答案。

Example

standard input	standard output
6 2 1 5 10 20 50 100 1 2	6 18

Note

对于样例，给出的货币系统中纸币面额有 1 元、5 元、10 元、20 元、50 元和 100 元六种。按照规则，当你要支付 6 元时只能拿出两张纸币，即一张 1 元和一张 5 元，也就是 $f(6, 6) = 2$。虽然支付 6 元也可以使用六张 1 元，但是这种方案并不满足尽量拿大面额纸币的原则，也不满足本题函数的定义。

Problem H. 松散子序列

给定一个长度为 n 且仅由小写字母组成的字符串 S，并做出如下约定。

- 子序列：从 S 中将若干元素（不一定连续）提取出来而不改变相对位置形成的序列称为子序列。
- k 松散子序列：若 S 的一个子序列中任意两个相邻字符在原串 S 中至少间隔 k 个位置，则这个子序列称为 S 的一个 k 松散子序列。具体地，对于 S 的一个长为 m 的子序列 $T = \overline{S_{pos_1}S_{pos_2}\cdots S_{pos_m}}$，$T$ 是 S 的一个 k 松散序列当且仅当 $\forall i \in [1, m-1]$，$pos_{i+1} - pos_i > k$。

现给定一个非负整数 k，你需要求出 S 本质不同的非空 k 松散子序列数目，对 998244353 取模的结果。

称 S 的两个子序列 A 与 B 本质不同，当且仅当 $|A| \neq |B|$ 或存在下标 i 满足 $A_i \neq B_i$。

Input

第一行一个整数 T（$1 \leq T \leq 10^6$），代表数据组数。

对于每一组数据，第一行两个整数 n, k（$1 \leq n \leq 10^6, 0 \leq k \leq n$），意义如题目描述。

第二行一个长为 n 的字符串 S，保证仅由小写字母组成。

保证对于所有数据，$\sum n \leq 10^6$。

Output

对于每组数据，输出一行一个整数，代表 S 本质不同的非空 k 松散子序列数目，对 998244353 取模。

Example

standard input	standard output
3 4 1 aabb 5 2 abcab 7 3 abcdece	3 6 10

Note

对于第一组数据，子序列 a，b，ab 符合要求。

对于第二组数据，子序列 a，b，c，aa，ab，bb 符合要求。

Problem I. 队伍取名

取名字总是很难的，而打 XCPC 的时候给队伍取名更是很难的。

Kagarii 最近在为队伍取队名的事情绞尽脑汁，如何将队名起得卓尔不群，独树一帜，大气磅礴，内涵丰富，意味深长，独具匠心，耐人寻味是一件很难的事情。更别说队伍里有三个人，需要满足三个人的口味更是难上加难。

最后，走投无路的 Kagarii 想到了一个非常老套的办法：从每个队员的名字里选一个字出来，组成三个字的队名。但他突然发现，三个队员的各选一个字出来，恰好是队伍里某个队员的名字！

比如 Kagarii 的队伍中三个人名字分别为：大大卷，小中大，大中小，那么这三个人可以取队名叫做「大中小」（「大大卷」的第一个字，「小中大」的第二个字，「大中小」的第三个字），恰好是一个队员的名字。

显然这个时候取队名就很方便了，所以 Kagarii 准备把这个取名方式推广到整个集训队。他希望见到更多这样的队名。

具体的，集训队有 n 名成员，其中第 i 名成员的名字是由三个数字 $S_{i,1}, S_{i,2}, S_{i,3}$ 组成（由于汉字的数量太多了，所以全部编码成整数表示），并且保证所有成员的名字都不相同。Kagarii 想知道有多少个选出三人组队的方式，使得从每个人名字里选一个**不同位置**的字，按照在**原本名字里的位置**将这三个选出的字组合起来，恰好是某个队员的名字，当然如果组成的队名不同，也看做不同的方案。

形式化的，即询问四元组 (i, j, k, id)（$1 \le i < j < k \le n$，$id \in \{i, j, k\}$）的个数，满足存在一个 $\{i, j, k\}$ 的排列 p，使得 $\forall x \in \{1, 2, 3\}, S_{id,x} = S_{p_x,x}$ 成立。

Input

第一行一个正整数 n（$3 \le n \le 10^5$），表示人数。

接下来 n 行，每行三个正整数 $S_{i,1}, S_{i,2}, S_{i,3}$（$1 \le S_{i,j} \le 10^6$），表示每个人的名字。保证对于任意 $i \ne j$，存在 x 满足 $S_{i,x} \ne S_{j,x}$。

Output

输出一行一个整数，表示合法的四元组个数。

Examples

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

Note

对于样例 1，一个满足条件的四元组为 $(2, 3, 5, 2)$，对应的排列 p 为 $\{3, 5, 2\}$。

对于样例 2，满足条件的四元组为 $(1, 2, 3, 1), (1, 2, 3, 2), (1, 2, 3, 3)$。

Problem J. 解谜比赛

一年一赛季的 CCPC（*Cipher & Code Penetrating Competition*）即将来临，今年的比赛采取了与众不同的方式来进行题目的发放。

本次比赛共有 n 道题目，编号从 1 到 n。但与往年不同的是，今年的题目并不是比赛一开始就全部解锁的，而是逐步解锁的。具体来说，组委会根据题目之间的关系确定了一张具有 n 个点的有向图，节点编号从 1 到 n，点 i 代表第 i 道题。最初每道题具有的能量都为 0，并且每道题目都有一个参数 a_i，表示如果这道题具有的能量大于等于 a_i，这道题就会被立刻解锁。当其解锁后，从该点出发的所有有向边将会**同时**向该边对应的终点题目传输 1 点能量，传输需要花费 w_i 的时间。

但组委会发现有些题目可能永远无法被解锁，这是组委会不想看到的，因此组委会设计了 k 个强制刷新器来帮助选手推进进度。每个强制刷新器管控一些题目，并且会在第 t_i 秒被启动，其启动后会立刻将其管控的题目的参数 a_i 修改为 0。

现在对于每个题目，你想知道其最早的解锁时间，或者判断其永远无法被解锁。

Input

第一行三个整数 n, m, k（$3 \le n \le 10^5, 0 \le m \le 10^6, 0 \le k \le 10^5$），分别表示题目的数量，有向边的数量和强制刷新器的数量。

第二行 n 个整数 a_i（$0 \le a_i \le 10^5$），表示每道题目的参数。保证存在至少一个 i 满足 $a_i = 0$。

接下来 k 行，第 j 行两个整数 t_j, sc_j（$0 \le t_j \le 10^9, 1 \le sc_j \le n$）表示第 j 个强制刷新器在第 t_j 秒启动，并管控 sc_j 道题目，然后接着 sc_j 个整数 $id_1, \ldots, id_{sc_j}$（$1 \le id_l \le n$），表示这个强制刷新器管控的题目编号。保证一个强制刷新器管控的题目两两不同。

接下来 m 行，每行三个整数 u, v, w（$1 \le u, v \le n, 0 \le w \le 10^9, u \ne v$），表示从题目 u 向题目 v 有一条有向边，其需要 w 的时间传输 1 点能量。

数据保证 $\sum sc_i \le 10^6$。

Output

输出一行共 n 个整数，表示解锁这个题目所需要的最短时间，如果这道题目永远无法被解锁，则输出 -1。

Examples

standard input	standard output
6 9 0 0 2 1 1 1 4 1 2 1 2 3 1 3 4 1 4 5 1 5 2 1 2 6 1 3 6 1 4 6 1 5 6 1	0 -1 -1 -1 -1 -1
6 9 1 0 2 1 1 1 4 100 2 3 5 1 2 1 2 3 1 3 4 1 4 5 1 5 2 1 2 6 1 3 6 1 4 6 1 5 6 1	0 101 100 101 100 102
4 3 0 1 0 1 1 3 1 10 1 2 100 2 4 1000	-1 0 -1 1000

Note

注意本题数据输入量过大，请采用较快的输入方法进行读入

Problem K. 打字机

lh8k 是一个喜欢拾荒的人。有一天他发现了一台打字机。这台打字机上有一个按钮和两个纸槽。经过一番折腾，他发现了这个打字机的工作规律

1. 开始时，你需要向两个纸槽中塞入两张纸条，其中上纸槽中塞入一张已经有文字的纸条作为模板，下纸槽塞入一张空白纸条。打字机会从上纸槽中的纸条读取内容写到下纸槽中的纸条中。为了方便，我们记上纸槽中纸条的内容为字符串 T，下纸槽中的为 S，初始时 $S = \varepsilon$ 为空。

2. 打字机在上下两个纸槽中均有一个指针。我们记上纸槽和下纸槽中的指针指向的位置分别为 p 和 q，初始时这两个指针都指向两个纸条开始的地方，即 $p = q = 1$。

3. 每次按下按钮，打字机会从上纸槽的指针处读取文字，并将该文字写入到下纸槽指针所指位置。（即 $S[q] := T[p]$），在打印完毕后，两个纸槽中的指针均会发生"移动"*。下纸槽中的指针始终向前移动（$q := q + 1$）；上纸槽中的指针开始时也是向前移动，但是若当前指针指向的位置为 T 的末尾的话，则会反向移动，一直移动到 T 的开头后又反向，循环往复。

比方说，若 lh8k 在上纸槽中插入一个写有 $T = \texttt{abcd}$ 的字符串，并且按下 20 次按钮，那么他就会在下纸槽中获得一张打印有 $S = \texttt{abcdcbabcdcbabcdcbab}$ 字符串的纸条。此外若 $|T| = 1$，则打印出的 S 只有一种字符，即 S 由 $T[1]$ 重复若干次组成。

lh8k 现在想要通过这个打字机打印出一个字符串 S，他想 T 的长度越短越好。他想知道 T 的长度最短可以是多少？请注意，从打印开始到结束不能更换纸条，也不能随意更改纸条和指针位置。打印 S 时必须从 T 的开头正向开始。

不过对于上述这个问题 lh8k 觉得只求一次答案不够过瘾，因此他想对 S 的每个前缀 S'，都求一遍最短的 T 长度是多少。

由于串总长度过长，因此对于每个串只需要输出 $\bigoplus_{i=1}^{|S|} i \times ans_i$ 的值，其中 ans_i 表示的是长度为 i 的前缀对应的最短的 T 的长度，$\bigoplus$ 表示按位异或。

Input

第一行一个整数 t（$1 \le t \le 10^3$），表示测试数据组数。

对于每组数据，一行一个仅由小写英文字母组成的字符串 S（$1 \le |S| \le 10^6$），表示 lh8k 想要通过打字机打印出的字符串。

数据保证 $\sum |S| \le 10^6$。

Output

对于每组数据，输出一行一个整数，表示 $\bigoplus_{i=1}^{|S|} i \times ans_i$ 的值，其中 ans_i 的含义见题面。

Example

standard input	standard output
5 a aa ababa abcdcbabcdcbabcdcbab popipopi	1 3 1 92 51

*其实发生移动的是纸条

Problem L. 路线选择

赶集，也叫赶墟，趁墟，是一种历史悠久的民间传统贸易活动，指在特定日期和固定地点进行的商品交易与社交集会，具有鲜明的民俗特征和地域文化差异。

今天，你所在的城市正在进行一场端午大集，你打算前往其中的 k 个摊位。具体地，现场被规划为一个 $n \times m$ 大小的网格，共有 n 行格点，每行有 m 个格点，每行之间的间距与每列之间的间距均为 1。我们用 $(0,0)$ 表示左上角的格点坐标，右下角的格点坐标为 $(n-1, m-1)$。大集的入口为 $(0,0)$，出口为 $(n-1, m-1)$，你需要从入口出发，按任意顺序经过 k 个摊位，最终到达出口。摊位沿网格的边设置，可以看成网格边上的点。形式化地，每个摊位坐标为 (x, y)，其中 x 或 y 为整数。

你需要在网格上的边移动，但现场盛况空前，你想尽快逛完。由于人流密度不同，在每条边上移动的速度也不同，你想知道从入口出发，逛完这 k 个摊位之后到达出口所需要的最短时间。逛摊位所花时间忽略不计。

Input

第一行三个整数 n, m, k（$2 \le n \le 50, 2 \le m \le 4, 1 \le k \le 10^5$）。

接下来 n 行，第 i 行 $m-1$ 个整数 $v^h_{i,0}, v^h_{i,1}, \ldots, v^h_{i,m-2}$（$1 \le v^h_{i,j} \le 10^5$），其中 $v^h_{i,j}$，表示在从 $(i-1, j)$ 到 $(i-1, j+1)$ 这条水平边上移动的速度。

接下来 $n-1$ 行，第 i 行 m 个整数 $v^v_{i,0}, v^v_{i,1}, \ldots, v^v_{i,m-1}$（$1 \le v^v_{i,j} \le 10^5$），其中 $v^v_{i,j}$，表示在从 $(i-1, j)$ 到 (i, j) 这条竖直边上移动的速度。

接下来 k 行，每行两个实数 x, y（$0 \le x \le n-1, 0 \le y \le m-1$），表示打算前往的摊位坐标。保证 x 和 y 中至少有一个是整数，并且小数点位数不超过 3 位。保证这 k 个摊位的位置互不相同。

Output

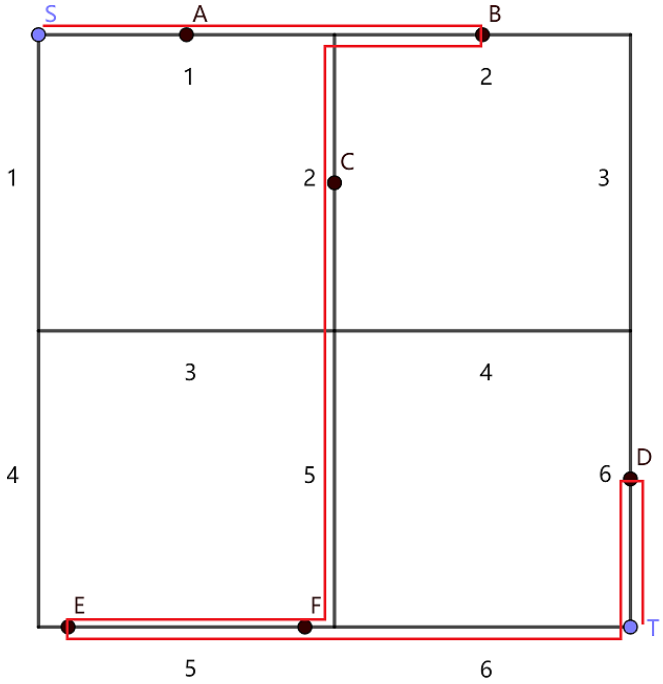
输出一行一个实数，表示从入口出发，逛完这 k 个摊位到达出口所需的最短时间。如果你的输出与答案之间的绝对误差或相对误差不超过 10^{-6}，则认为你的输出正确。

Example

standard input	standard output
3 3 6 1 2 3 4 5 6 1 2 3 4 5 6 0 0.5 0 1.5 0.5 1 1.5 2 2 0.1 2 0.9	2.893333333

Note

对于样例，用 S 表示入口，T 表示出口，按在样例中出现的先后顺序给摊位编号为 A 到 F。一种花费时间最短的可行线路如下图所示。



边的旁边标注的数字为在这条边上移动的速度。