Problem A. Matrix Game

everflame and KisuraOP are playing a game. At the start of the game, everflame provides a matrix, and KisuraOP's goal is to manipulate this matrix to achieve the highest possible score.

The matrix given by everflame is an n by m matrix $a_{i,j}$, where each position can initially be either 0 or 1. Additionally, he provides two integers A and B.

KisuraOP can perform any number of operations (including 0) on this matrix, where each operation can be one of the following two types:

- Choose an $i \in [1, n]$ and flip all elements in the i-th row.
- Choose a $j \in [1, m]$ and flip all elements in the j-th column.

Flipping an element means that if it was originally 1, it will become 0; conversely, if it was originally 0, it will become 1.

For the element $a_{i,j}$ in the i-th row and j-th column of the matrix, if $a_{i,j} = 1$, then this element will contribute $(A \cdot i + B \cdot j)$ to the score; otherwise, its contribution will be 0. The total score of the matrix will be the sum of the scores of all $n \times m$ elements.

Please help KisuraOP determine the maximum score that can be achieved after performing certain operations on this matrix.

Input

The first line contains four integers n, m, A, B ($1 \le n \le 10^6, 1 \le m \le 10, 0 \le |A|, |B| \le 10^6$).

The next n lines contain strings of length m. The character in the i-th row and j-th column represents $a_{i,j}$ ($a_{i,j} \in \{0, 1\}$).

Output

Output a single integer, which is the maximum possible score.

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

For the first example, first flipping the 1-st row and then flipping the 2-nd column can make the entire matrix 1, achieving the maximum score. The score is $(1\cdot1+1\cdot1)+(1\cdot1+1\cdot2)+(1\cdot2+1\cdot1)+(1\cdot2+1\cdot2) = 12$.

For the second example, only flipping the 2-nd column achieves the maximum score, and it can be proven that there is no better solution. The score is $1 \cdot 2 + (-5) \cdot 2 = -8$.

Problem B. Integer Generator

Given a set of integers S, your task is to generate a given integer x using no more than 70 bitwise operations on S.

Specifically, given an integer set S of size n and an integer x, each operation allows you to choose two integers a and b from S (they can be the same) and insert one of the integers a `or` b, $a \oplus b$, or a `and` b into S. You need to determine whether it is possible to make $x \in S$ with no more than 70 operations, and if so, provide a valid sequence of operations.

Here, a `or` b refers to the bitwise OR of a and b, $a \oplus b$ refers to the bitwise XOR of a and b, and a `and` b refers to the bitwise AND of a and b.

Input

The first line contains two integers n, x ($1 \le n \le 10^5, 0 \le x < 2^{30}$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 2^{30}$), representing the initial elements in S, ensuring that these integers are all distinct.

Output

If it is not possible to make $x \in S$ with no more than 70 operations, output a single integer -1.

Otherwise, output the first line with an integer k ($0 \le k \le 70$), indicating the number of operations.

Next, output k lines, each representing one of the k operations. For each operation, output three integers t, a, b ($t \in \{0, 1, 2\}$). If $t = 0$, it means this operation inserts a `or` b into S; if $t = 1$, it means it inserts $a \oplus b$ into S; if $t = 2$, it means it inserts a `and` b into S. You need to ensure that for the S before this operation, both $a \in S$ and $b \in S$ should be met.

In this problem, you do not need to minimize the number of operations; if there are multiple valid operation sequences, any one of them will be accepted.

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. Cutting Cards

There are n cards, numbered from 1 to n. A cutting card operation is performed as follows:

1. Arrange the cards in ascending order according to their index.
2. Choose a positive integer k ($1 \le k \le n$) and divide the cards into k piles. Each pile must contain at least one card, and the cards in each pile must have consecutive indexes and be arranged in ascending order from top to bottom.
3. Rearrange the piles in any order and line them up in a row.
4. Traverse the piles from left to right, and for each pile, if there are still cards in the pile, take the top card and place it at the end of the arranged cards.
5. Stop the operation when all the cards have been placed in the arranged sequence.

For example, for five cards $\{1, 2, 3, 4, 5\}$, they can be divided into $\{1\}, \{2, 3\}, \{4, 5\}$, $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}$, or $\{1, 2, 3, 4, 5\}$, but they cannot be divided into $\{1\}, \{2, 5\}, \{3, 4\}$ because this violates the consecutive numbering principle, nor can they be divided into $\{1\}, \{3, 2\}, \{4, 5\}$ because this violates the principle of arranging in ascending order.

For another example, for the already arranged three piles of cards from left to right $\{1\}, \{4, 5\}, \{2, 3\}$, the only possible arranged sequence is $\{1, 4, 2, 5, 3\}$, and it is not possible to obtain $\{1, 2, 4, 5, 3\}$ because this violates the left-to-right traversal order, nor is it possible to obtain $\{1, 5, 2, 4, 3\}$ because this violates the taking-from-top principle.

There is a target arrangement of the cards, and you need to calculate how many different cutting card operations can obtain this target arrangement. Two cutting card operations are different if and only if the way the cards are divided into piles is different or the arrangement of the piles is different.

The target arrangement may be modified, and for each modification, the elements in the two positions of the target arrangement are exchanged, and the answer needs to be output. The modifications are persistent, meaning that previous modifications will be retained.

Input

The first line contains two integers n, Q ($2 \le n \le 10^5, 1 \le Q \le 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$), representing the initial target arrangement.

The next Q lines each contain two integers x, y ($1 \le x, y \le n, x \ne y$), representing the positions to be swapped in each modification.

Output

Output $Q + 1$ lines. The first line outputs the answer before any modifications, and the second to the $Q + 1$-th lines outputs the answer after the first to the Q-th modifications.

Example

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

Note

In the example, there are 4 cards, and the initial target sequence is $\{1, 3, 2, 4\}$. There are three cutting card operations that can obtain the target sequence:

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

After the first swap, the target sequence becomes $\{1, 2, 3, 4\}$. There are four cutting card operations that can obtain the target sequence:

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

Problem D. Spell Generation

Hiding various taboo words in a string to achieve the effect of a spell is a common phenomenon in the magical world. To better cope with cases of illegal spell usage, the people of this country need to learn spellcraft to help themselves defend against spells.

Today's lesson is about a simple spell generator, which has two modes: **Click** and **Press**.

For each click, it takes 1 second to generate a spell of length 1.

For each press, you must first choose a positive integer x, and then it takes 2^x seconds to generate a spell of length 10^x.

Now you are given some lengths of spells that need to be generated, and you need to calculate the minimum time required to generate each spell.

Input

The first line contains an integer T ($1 \leq T \leq 50000$), representing the total number of requests.

Each of the following lines contains an integer r ($1 \leq r \leq 10^{18}$), representing the length of each spell to be generated.

Output

Output T lines, each line representing the minimum time required.

Example

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

Problem E. Grid Coloring

There is a grid with 2 rows and n columns, where the rows are numbered from 1 to 2 from top to bottom, and the columns are numbered from 1 to n from left to right. There are $2n$ colors numbered from 1 to $2n$. Some cells in the grid have already been colored, and you need to color the remaining cells (cells that have already been colored cannot be recolored) in such a way that the number of four-connected components of the same color is minimized. Please construct a coloring scheme.

Four-connected means that if two cells of the same color share a common edge, then these two cells are considered connected.

Input

The first line contains an integer n ($1 \le n \le 10^5$).

The second line contains n integers $a_{1,1}, a_{1,2}, \ldots, a_{1,n}$ ($0 \le a_{1,i} \le 2n$), representing the coloring of the first row of the grid.

The third line contains n integers $a_{2,1}, a_{2,2}, \ldots, a_{2,n}$ ($0 \le a_{2,i} \le 2n$), representing the coloring of the second row of the grid.

If $a_{i,j} = 0$, it means that the cell in the i-th row and j-th column is not colored, otherwise it represents the color of the cell in the i-th row and j-th column as $a_{i,j}$. It is guaranteed that at least one cell is initially colored ($a_{i,j} \neq 0$).

Output

Output two lines, each containing n positive integers $b_{i,j}$ ($1 \le b_{i,j} \le 2n$), representing a coloring scheme. The output should ensure that cells that have already been colored cannot be recolored, i.e., for cells where $a_{i,j} \neq 0$, the output should ensure that $b_{i,j} = a_{i,j}$.

If there are multiple valid coloring schemes, output any one of them.

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. Ranking Prediction

You and your teammates have just finished an ICPC competition. Five hours of competition have drained your energy, and your teammate has eaten your lunch. Now you can only lean on the desk and look at the ranking board. The award ceremony has not yet taken place, so the ranking is still frozen, meaning that you know the submission times and whether your team passed during the entire competition, but for other teams, you know the time of each submission they made, and you know whether each submission was accepted before the ranking was frozen, but you do not know whether the submissions made after the ranking was frozen were accepted.

When you check the ranking, you notice a team you are very concerned about. You know the time and status of each submission they made before the ranking was frozen, as well as the times of each submission made after the ranking was frozen. You want to know whether their team's ranking will be **strictly higher** than your team's. To determine the possibility of their ranking being strictly higher than yours, you also want to know the minimum number of problems they need to solve after the ranking is frozen.

In ICPC competitions, the penalty time is calculated according to the following rules. Suppose a team **solved** m problems, numbered from 1 to m. For each solved problem i, let the time of the first accepted submission be t_i, and the number of submissions before solving this problem be c_i. The penalty time p is calculated as follows:

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

In this problem, special factors such as compile errors that do not count towards penalty time are not considered.

For two teams A and B, team A is said to have a strictly higher ranking than team B if and only if the number of problems solved by A is greater than that solved by B, or if the number of problems solved by A and B is equal, and the penalty time of A is less than that of B.

Input

The first line contains an integer T ($1 \le T \le 100$), indicating the number of test cases.

For each test case, the first line contains three integers n, a, b ($10 \le n \le 15, 1 \le a \le n, 0 \le b \le 10^5$), representing that there are n problems in the competition, your team solved a problems by the end of the competition, and the penalty time is b.

The second line contains an integer s ($0 \le s \le 10^3$), indicating the number of submissions made by the team you are concerned about during the normal competition.

The next s lines each contain an integer followed by two strings t, p, v ($0 \le t < 300$). This indicates that at minute t, a submission was made for problem p, and the result was v. It is guaranteed that submissions are given in chronological order (this means that when t is the same, the submit order is the order given by the input), p is one of the first n uppercase letters, and $v \in \{\texttt{ac}, \texttt{rj}, \texttt{pd}\}$. `ac` means this submission was accepted, `rj` means the submission was rejected, and `pd` means this submission is in a frozen ranking state, and it is unknown whether this submission was accepted. It is guaranteed that when $t < 240$, v is not `pd`, and when $t \ge 240$, v is guaranteed to be `pd`.

Output

For each test case, output a single integer on a new line. If it is impossible for the team you are concerned about to have a final ranking strictly higher than your team, output -1; otherwise, output the minimum number of problems they need to solve after the ranking is frozen for their final ranking to be strictly higher than yours.

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. Monetary System

Given a sorted ascending array A of length n, where the i-th element is A_i with $A_1 = 1$.

Construct a currency system using this array. For any positive integer x, define the currency note count function as $f(x, n)$, where n represents the array's length. This function represents the number of banknotes required to pay x yuan under this currency system, following the greedy principle of always using the largest possible denominations first before smaller ones. For any positive integer x and any positive integer $y \in [1, n]$, $f(x, y)$ satisfies:

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

You need to process q queries. For each query, given an integer m, determine how many positive integers x satisfy $f(x, n) = m$.

Input

The first line contains two integers n and q ($1 \le n \le 10^5, 1 \le q \le 10^6$), representing the length of the array A and the number of queries.

The second line contains n positive integers $A_1, A_2, \ldots, A_n$ ($1 = A_1 < A_2 < \ldots < A_n \le 10^6$), the elements of array A.

The third line contains q integers $m_1, m_2, \ldots, m_q$ ($1 \le m_i \le 10^9$), where m_i is the value for the i-th query.

Output

Output a line containing q space-separated integers. The i-th integer is the answer to the i-th query.

Example

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

Note

For the sample, six denominations of notes are given in the monetary system: 1-yuan, 5-yuan, 10-yuan, 20-yuan, 50-yuan, and 100-yuan. When making a 6-yuan payment using notes, you must use two notes: one 1-yuan note and one 5-yuan note, i.e., $f(6, 6) = 2$. Although you could theoretically use six 1-yuan notes to make a 6-yuan payment, this method does not satisfy the principle of taking as many large denomination notes as possible and therefore does not satisfy the definition of the function in this problem.

Problem H. Loose Subsequences

Given a string S of length n consisting only of lowercase letters, the following conventions are established.

- Subsequence: A sequence formed by extracting several elements (not necessarily consecutive) from S without changing their relative positions is called a subsequence.
- A k-loose subsequence: If any two adjacent characters in a subsequence of S are at least k positions apart in the original string S, then this subsequence is called a k-loose subsequence of S. Specifically, for a subsequence $T = \overline{S_{pos_1} S_{pos_2} \cdots S_{pos_m}}$ of S of length m, T is a k-loose subsequence of S if and only if $\forall i \in [1, m-1]$, $pos_{i+1} - pos_i > k$.

Now, given a non-negative integer k, you need to calculate the number of distinct non-empty k-loose subsequences of S, and output the result modulo 998244353.

Two subsequences A and B of S are considered different if and only if $|A| \neq |B|$ or there exists an index i such that $A_i \neq B_i$.

Input

The first line contains an integer T ($1 \leq T \leq 10^6$), representing the number of test cases.

For each test case, the first line contains two integers n, k ($1 \leq n \leq 10^6, 0 \leq k \leq n$), as described in the problem statement.

The second line contains a string S of length n, guaranteed to consist only of lowercase letters.

It is guaranteed that for all test cases, $\sum n \leq 10^6$.

Output

For each test case, output a single integer on a new line, representing the number of distinct non-empty k-loose subsequences of S, modulo 998244353.

Example

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

Note

For the first test case, the subsequences `a`, `b`, `ab` meet the requirements.

For the second test case, the subsequences `a`, `b`, `c`, `aa`, `ab`, `bb` meet the requirements.

Problem I. Team Naming

Naming is always difficult, and coming up with a name for a team competing in XCPC is even harder.

Kagarii has been racking his brain over how to come up with a unique, distinctive, grand, profound, and cleverly crafted team name that resonates with all three members of the team. It is even more challenging to satisfy the tastes of all three members.

In the end, desperate Kagarii thought of a very trite method: to select one character from each team member's name to form a three-character team name. However, he suddenly realized that the characters selected from the names of the three members happen to form the name of one of the team members!

For example, in Kagarii's team, the names of the three members are: "Da Da Juan", "Xiao Zhong Da", and "Da Zhong Xiao". These three can form the team name "Da Zhong Xiao" (the first character of "Da Da Juan", the second character of "Xiao Zhong Da", and the third character of "Da Zhong Xiao"), which happens to be the name of one of the members.

Clearly, this makes naming the team much easier, so Kagarii plans to promote this naming method to the entire training team. He hopes to see more such team names.

Specifically, the training team has n members, where the name of the i-th member consists of three numbers $S_{i,1}, S_{i,2}, S_{i,3}$ (since there are too many Chinese characters, they are all encoded as integers), and it is guaranteed that all members have different names. Kagarii wants to know how many ways there are to select a team of three members such that selecting one character from each person's name at **different positions**, and combining these characters according to the positions of the characters in the **original names**, exactly forms the name of one of the team members. Of course, if the formed team names are different, they are considered different solutions.

Formally, the task is to count the number of quadruples (i, j, k, id), where $1 \le i < j < k \le n$ and $id \in \{i, j, k\}$, such that there exists a permutation p of $\{i, j, k\}$ satisfying $\forall x \in \{1, 2, 3\}, S_{id,x} = S_{p_x,x}$.

Input

The first line contains a positive integer n ($3 \le n \le 10^5$), representing the number of members.

The next n lines each contain three positive integers $S_{i,1}, S_{i,2}, S_{i,3}$ ($1 \le S_{i,j} \le 10^6$), representing each person's name. It is guaranteed that for any $i \ne j$, there exists an x such that $S_{i,x} \ne S_{j,x}$.

Output

Output a single line containing an integer, representing the number of valid quadruples.

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

For the sample 1, a quadruple satisfying the condition is $(2, 3, 5, 2)$, and the corresponding permutation p

is $\{3, 5, 2\}$.

For the sample 2, quadruples satisfying the condition are $(1, 2, 3, 1), (1, 2, 3, 2), (1, 2, 3, 3)$.

Problem J. Puzzle Competition

The annual CCPC (*Cipher & Code Penetrating Competition*) is approaching, and this year's competition has a unique way of releasing the problems.

This competition has a total of n problems, numbered from 1 to n. However, unlike previous years, this year's problems are not all unlocked at the beginning of the competition; instead, they are unlocked gradually. Specifically, the organizing committee has determined a directed graph with n nodes based on the relationships between the problems, where the nodes are numbered from 1 to n and node i represents the i-th problem. Initially, each problem has an energy level of 0, and each problem has a parameter a_i, which indicates that if the energy level of this problem is greater than or equal to a_i, the problem will be immediately unlocked. Once unlocked, all directed edges originating from that node will **simultaneously** transfer 1 point of energy to the corresponding destination problem, which takes w_i seconds.

However, the organizing committee has discovered that some problems may never be unlocked, which is something they want to avoid. Therefore, they designed k forced refreshers to help participants progress. Each forced refresher controls some problems and will be activated at time t_i seconds, immediately modifying the parameter a_i of the problems it controls to 0.

Now, for each problem, you want to know its earliest unlock time or determine if it can never be unlocked.

Input

The first line contains three integers n, m, k ($3 \le n \le 10^5, 0 \le m \le 10^6, 0 \le k \le 10^5$), representing the number of problems, the number of directed edges, and the number of forced refreshers.

The second line contains n integers a_i ($0 \le a_i \le 10^5$), representing the parameters of each problem. It is guaranteed that there is at least one i such that $a_i = 0$.

The next k lines contain the j-th line with two integers t_j, sc_j ($0 \le t_j \le 10^9, 1 \le sc_j \le n$), indicating that the j-th forced refresher is activated at t_j seconds and controls sc_j problems, followed by sc_j integers $id_1, \ldots, id_{sc_j}$ ($1 \le id_l \le n$), representing the problem numbers controlled by this forced refresher. It is guaranteed that the problems controlled by a forced refresher are all distinct.

The next m lines each contain three integers u, v, w ($1 \le u, v \le n, 0 \le w \le 10^9, u \ne v$), indicating that there is a directed edge from problem u to problem v, which takes w seconds to transfer 1 point of energy.

It is guaranteed that $\sum sc_i \le 10^6$.

Output

Output a single line containing n integers, representing the shortest time required to unlock each problem. If a problem can never be unlocked, output -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

Note: The input size for this problem is very large; please use a faster input method for reading.

Problem K. Typewriter

lh8k is a person who enjoys scavenging. One day, he discovered a typewriter. This typewriter has a button and two paper slots. After some experimentation, he figured out how this typewriter works.

1. At the beginning, you need to insert two pieces of paper into the two slots, with one piece of paper in the upper slot containing text as a template, and a blank piece of paper in the lower slot. The typewriter will read the content from the paper in the upper slot and write it onto the paper in the lower slot. For convenience, we denote the content of the paper in the upper slot as the string T, and the content in the lower slot as S, initially $S = \varepsilon$ (empty).
2. The typewriter has a pointer in both the upper and lower slots. We denote the positions of the pointers in the upper and lower slots as p and q, respectively, initially both pointing to the start of the two pieces of paper, i.e., $p = q = 1$.
3. Each time the button is pressed, the typewriter reads the text from the position of the pointer in the upper slot and writes that text into the position pointed to by the pointer in the lower slot (i.e., $S[q] := T[p]$). After printing, both pointers in the slots will "move" *. The pointer in the lower slot always moves forward ($q := q + 1$); the pointer in the upper slot also moves forward initially, but if the current pointer position is at the end of T, it will move backward, continuing to move until it reaches the beginning of T, then it will move forward again, repeating this cycle.

For example, if lh8k inserts a string $T =$ `abcd` into the upper slot and presses the button 20 times, he will obtain a printed paper in the lower slot with the string $S =$ `abcdcbabcdcbabcdcbab`. Furthermore, if $|T| = 1$, the printed S will consist of only one character, i.e., S consists of $T[1]$ repeated several times.

lh8k now wants to print a string S using this typewriter, and he wants the length of T to be as short as possible. He wants to know what the minimum possible length of T can be. Note that the paper cannot be changed from the start to the end of the printing process, nor can the positions of the paper or pointers be changed arbitrarily. Printing S must start from the beginning of T in a forward direction.

However, for this problem, lh8k feels that just finding one answer is not enough, so he wants to find the shortest length of T for each prefix S' of the string S.

Due to the overall length of the strings being too long, for each string, you only need to output the value of $\bigoplus_{i=1}^{|S|} i \times ans_i$, where ans_i represents the shortest length of T corresponding to the prefix of length i, and $\bigoplus$ denotes the bitwise XOR operation.

Input

The first line contains an integer t ($1 \le t \le 10^3$), indicating the number of test cases.

For each test case, there is one line containing a string S composed only of lowercase English letters ($1 \le |S| \le 10^6$), representing the string that lh8k wants to print using the typewriter.

The data guarantees that $\sum |S| \le 10^6$.

Output

For each test case, output a single integer on a new line, representing the value of $\bigoplus_{i=1}^{|S|} i \times ans_i$, where the meaning of ans_i is as described in the problem statement.

*Actually, it is the paper that moves.

Example

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

Problem L. Route Selection

Market fair is a long-standing traditional folk trading activity that refers to the trading of goods and social gatherings held on specific dates and at fixed locations, characterized by distinct folk customs and regional cultural differences.

Today, in your city, a Dragon Boat Festival fair is taking place, and you plan to visit k stalls. Specifically, the venue is organized as an $n \times m$ grid, consisting of n rows of grid points, with m grid points in each row, and the spacing between rows and columns is 1. We denote the coordinates of the top-left grid point as $(0, 0)$ and the bottom-right grid point as $(n-1, m-1)$. The entrance of the fair is at $(0, 0)$, and the exit is at $(n-1, m-1)$. You need to start from the entrance, visit the k stalls in any order, and finally reach the exit. The stalls are set along the edges of the grid and can be considered as points on the edges of the grid. Formally, the coordinates of each stall are (x, y), where either x or y is an integer.

You need to move along the edges of the grid, but the fair is bustling, and you want to finish your visit as quickly as possible. Due to varying crowd densities, the speed of movement along each edge differs, and you want to know the shortest time required to start from the entrance, visit all k stalls, and reach the exit. The time spent at the stalls is negligible.

Input

The first line contains three integers n, m, k ($2 \le n \le 50, 2 \le m \le 4, 1 \le k \le 10^5$).

The next n lines contain the i-th line with $m-1$ integers $v^h_{i,0}, v^h_{i,1}, \ldots, v^h_{i,m-2}$ ($1 \le v^h_{i,j} \le 10^5$), where $v^h_{i,j}$ represents the speed of movement along the horizontal edge from $(i-1, j)$ to $(i-1, j+1)$.

The next $n-1$ lines contain the i-th line with m integers $v^v_{i,0}, v^v_{i,1}, \ldots, v^v_{i,m-1}$ ($1 \le v^v_{i,j} \le 10^5$), where $v^v_{i,j}$ represents the speed of movement along the vertical edge from $(i-1, j)$ to (i, j).

The next k lines each contain two real numbers x, y ($0 \le x \le n-1, 0 \le y \le m-1$), representing the coordinates of the stalls you intend to visit. It is guaranteed that at least one of x or y is an integer, and the decimal places do not exceed 3. It is also guaranteed that the positions of these k stalls are distinct.

Output

Output a single line with a real number, representing the shortest time required to start from the entrance, visit all k stalls, and reach the exit. If the absolute or relative error of your output compared to the answer does not exceed 10^{-6}, your output will be considered correct.

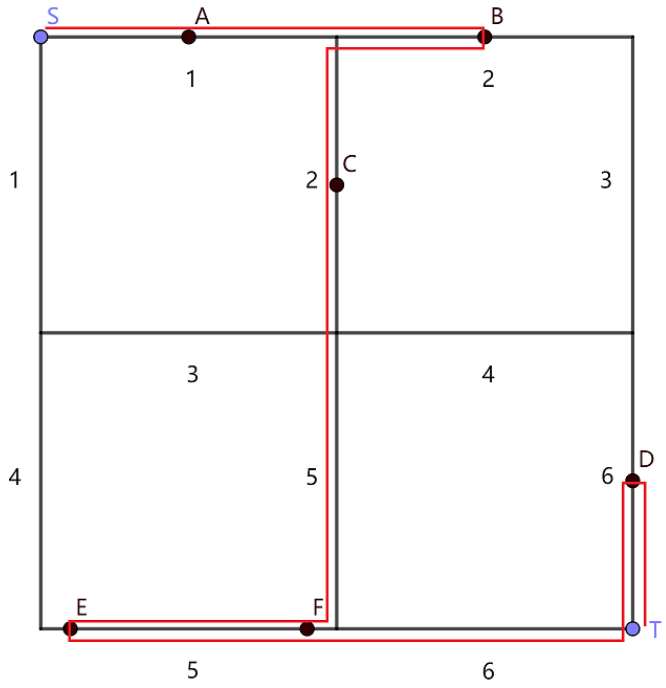
Example

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

For the sample, let S represent the entrance and T represent the exit. The stalls are numbered from A

to F in the order they appear in the sample. One of the feasible routes with the shortest travel time is shown in the figure below.



The numbers next to the edges indicate the speed of movement along that edge.