Problem A. Alice, Bob and Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 megabytes

Alice and Bob have recently played many complex games, so they want to play something simple to relax.

They remembered a very simple game called "Seven", which can be played with just a deck of cards. The traditional rules of Seven are a bit complex, so Alice and Bob are going to play a simplified version of Seven.

The playing cards used in the game include four suits: Spades, Hearts, Clubs, and Diamonds. Each suit has one card with each number from 1 to 13, totaling 52 cards.

The rules of the simplified game are as follows:

1. First, a deck of 52 cards is evenly split into two parts, with each part containing 26 cards, and Alice and Bob each take one of the parts;
2. The player who has the 7 of Spades places their 7 of Spades on the table;
3. Then, starting with the other player, they take turns playing. In each round, a player must choose one card from their hand to place on the table. The card that can be played must satisfy one of the following two conditions:
   - The card's point value is 7;
   - There is a card on the table that has the same suit as the card to be played, and the point value differs by exactly 1.
4. If both players have played all their cards, the game is a draw. Otherwise, if the player whose turn it is cannot play a card according to the rules, this player loses, and the other player wins.

Since the sequences of different suits in this game are independent, players often align different suits of 7 vertically and connect the sequences on either side of the 7, arranging the same suit cards in a row.

Given that Alice and Bob are both smart enough, now that Alice and Bob's initial hands are provided, please calculate the result of the game when both play optimally.

Input

Since one player's hand can be used to infer the other player's hand, the input only contains all of Alice's cards.

The input consists of a single line with 26 space-separated strings, each representing a card in Alice's hand. The cards are all distinct, and can be given in any order.

Each card is described by its suit and point value. The suit is represented by a character: `S` for Spades, `H` for Hearts, `C` for Clubs, and `D` for Diamonds. The point value is represented by a corresponding one or two-digit number, with the minimum being 1 and the maximum being 13, without leading zeros. For example, the 7 of Spades corresponds to the string `S7`.

Output

The output consists of a single line with a string that indicates the result of the game. If Alice will win this game when both players play optimally, output `Alice`; if Bob will win, output `Bob`; if it's a draw, output `Draw`.

Examples

standard input
S4 S6 S11 S13 H3 H4 H5 H6 H8 H10 H12 C2 C4 C5 C6 C8 C9 C11 C13 D1 D4 D5 D6 D8 D10 D12
standard output
Bob

standard input
S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13
standard output
Draw

standard input
H1 H2 H3 H4 H5 H6 H7 H8 H9 H10 H11 H12 C13 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13
standard output
Alice

Problem B. Big Data Permutation

Input file:	*standard input*
Output file:	*standard output*
Time limit:	15 seconds
Memory limit:	2 gigabytes

You are given a sequence $a_1, \ldots, a_n$ and a permutation $b_1, \ldots, b_n$. Process m operations of the following form:

- Modification operation: given x and y, change a_x into y.
- Query operation: given ℓ, r, and x, find the longest sub-interval $[\ell', r']$ within the interval $[\ell, r]$ (formally, $\ell \le \ell' \le r' \le r$) such that, for $\ell' \le j < r'$, we have $a_{j+1} = b_{a_j}$, and additionally, there exists an i such that $\ell' \le i \le r'$ and $a_i = x$. You only need to output the maximum length of such sub-interval (formally, $r' - \ell' + 1$); if there is none, output 0.

Input

The first line of input contains two integers n and m ($1 \le n, m \le 10^6$).

The second line contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le n$).

The third line contains n integers $b_1, \ldots, b_n$ ($1 \le b_i \le n$; all b_i are distinct).

Each of the next m lines consists of integers and has either the form "`1` x y" for a modification operation or the form "`2` ℓ r x" for a query operation ($1 \le x, y \le n$; $1 \le \ell \le r \le n$).

You may assume that there is at least one query operation.

Output

For each query operation, output one line with the corresponding maximum length.

Example

standard input	*standard output*
8 10 1 4 7 3 8 2 4 7 5 4 8 7 1 6 3 2 2 6 6 2 2 8 8 7 1 4 3 2 6 8 3 2 4 4 3 2 4 4 3 2 6 8 4 2 5 6 2 2 1 8 1 2 1 1 6	1 1 0 1 1 3 2 1 0

Problem C. Corrupted Order

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 seconds
Memory limit:	1024 megabytes

This world is in imminent danger! Order has completely fallen into chaos.

Order can be abstracted as an $n \times n$ matrix, where the matrix contains a permutation of numbers from 1 to n^2. You want to save the world, so you called upon a deity to help restore order. However, the deity is not omnipotent; it can only swap two numbers in the same row or the same column of the matrix. Moreover, it does not know how to swap to restore order, and must rely on your guidance.

Fortunately, you do not necessarily need to complete the restoration in the minimum number of swaps. You only need to ensure that your number of swaps is not worse than the worst-case scenario. In other words, if your number of swaps is k, and the maximum minimum number of swaps for all permutations from 1 to n^2 is k_0, you only need to satisfy $k \leq k_0$.

Restoration refers to transforming the matrix into the following matrix:

$$\begin{matrix} 1 & 2 & 3 & \cdots & n \\ n+1 & n+2 & n+3 & \cdots & 2n \\ 2n+1 & 2n+2 & 2n+3 & \cdots & 3n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (n-1)n+1 & (n-1)n+2 & (n-1)n+3 & \cdots & n^2 \end{matrix}$$

Input

The first line of input contains a positive integer n ($1 \leq n \leq 1000$).

Each of the next n lines contains n positive integers. Together, they represent the $n \times n$ matrix. It is guaranteed that each number from 1 to n^2 appears in the matrix exactly once.

Output

The first line should contain a non-negative integer k: the number of swaps you made. This number should not be greater than the minimum number of swaps for the worst possible case of $n \times n$ matrix.

The next k lines should each contain four positive integers: x_1, y_1, x_2, y_2. They indicate that you swapped the number in row x_1, column y_1 with the number in row x_2, column y_2.

You need to ensure that $x_1 = x_2$ or $y_1 = y_2$.

If there is more than one solution, print any one of them.

Examples

standard input	*standard output*
2 4 2 3 1	3 1 1 1 2 1 2 2 2 1 1 1 2
2 2 1 3 4	3 2 1 2 2 1 1 1 2 2 1 2 2
2 3 2 1 4	2 1 1 1 1 1 1 2 1
2 1 2 3 4	0

Note

For Sample 1, it can be proven that this is one of the solutions with the minimum number of swaps, and it clearly meets the conditions.

For Sample 2, the sample output's solution is not the one with the minimum number of swaps, but we know that there exists a permutation from 1 to n^2 (the previous example) that requires at least 3 swaps, so this solution is also feasible.

For Sample 3, we allow the case where $(x_1, y_1) = (x_2, y_2)$.

For Sample 4, note that k can be equal to 0.

Problem D. Difficult Password

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	1024 megabytes

To ensure that students use a strong cluster login password, the course team has configured a password complexity policy on the cluster. At the beginning of the semester, all enrolled students will receive a randomly generated initial password. After logging into the cluster with the initial password, the cluster will require students to enter a new password, and access to the cluster will only be granted after the password has been changed. In this problem, we assume the password complexity policy is as follows:

- The password must contain at least L characters, and must include at least one digit and at least one letter.
- The password cannot contain *consecutive* A *repeated* characters. For example, `2333` contains three consecutive repeated characters.
- The password cannot contain *consecutive* B characters that form an ascending or descending sequence. An ascending sequence is defined as a *consecutive* substring of one of the three strings: `0123456789`, `ABCDEFGHIJKLMNOPQRSTUVWXYZ`, and `abcdefghijklmnopqrstuvwxyz`. A descending sequence is the reverse of a consecutive substring of one of these three strings. For example:
  - `6789` is an ascending sequence of length 4, and `FED` is a descending sequence of length 3;
  - `90`, `AZ`, and `az` are not ascending or descending sequences;
  - `GPU` is not an ascending sequence because it is not consecutive;
  - `Def` is not an ascending sequence because the letter case is inconsistent (but it contains the ascending consecutive subsequence `ef` of length 2);
  - `1112345678999` is not an ascending sequence, but its subsequence `123456789` is ascending.

Assuming the password consists only of digits and uppercase and lowercase letters, determine how many passwords of length not exceeding R satisfy the password complexity policy. This number can be very large, so find it modulo 1 000 000 007.

Input

The input consists of a single line containing four positive integers: L, R, A, B ($1 \le L \le R \le 10^9$; $2 \le A \le 6$; $2 \le B \le 26$).

Output

Output a non-negative integer representing the number of passwords that satisfy the password complexity policy and have length not exceeding R, modulo 1 000 000 007.

Examples

standard input	*standard output*
2 2 2 2	1040
12 24 3 4	718185656
100 10000000 6 6	146399052

Note

In Sample 1, since the password must contain at least one digit and at least one letter, there are $2 \cdot 10 \cdot (26 \cdot 2) = 1040$ valid passwords.

Problem E. Edges and Divisors

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 megabytes

We play a game on a directed acyclic graph $G = (V, E)$. We start at vertex $s_1 \in V$. The rules of the game are as follows:

- The game consists of rounds numbered from 1.
- In the i-th round, the player selects a path p_i starting from s_i and containing at least one edge such that the sum of the weights of all edges belonging to this path is an exact multiple of $(i + 1)$. If the player cannot select such a path, the player has failed, and will not score any points. Otherwise, the round ends successfully, and the endpoint of p_i is recorded as s_{i+1}.
- After a successful round, the player can either end the game or continue with the next round. If the player chooses to end the game, the selected i paths $p_1, \ldots, p_i$ are called doubling paths, and the score is calculated.

If the player has not failed, then when ending the game, for the selected doubling paths $p_1, \ldots, p_k$, the score of the game is defined as $\sum_{i=1}^{k} a_i \left|p_i\right| / k$, where $|p_i|$ represents the number of edges in path p_i, and a_i is the weight given in the input. Clearly, as the graph is acyclic, at most $(n - 1)$ paths can be selected, so the input only provides the weights $a_1, \ldots, a_{n-1}$.

Given the graph and the starting vertex, calculate the maximum achievable score in the game.

Input

The first line of input contains three integers, n, m, and s_1: the number of vertices, the number of edges, and the index of the starting vertex ($2 \le n \le 100$; $1 \le m \le \frac{n(n-1)}{2}$; $1 \le s_1 \le n$).

The second line contains $(n - 1)$ integers $a_1, \ldots, a_{n-1}$: the weights used to calculate the score ($1 \le a_1 \le a_2 \le \ldots \le a_{n-1} \le 10^9$).

Each of the next m lines contains three integers, u_i, v_i, and w_i, describing a directed edge from u_i to v_i with weight w_i ($1 \le u_i, v_i \le n$; $u_i \ne v_i$; $1 \le w_i \le 10^9$). It is guaranteed that the graph is acyclic, the graph is connected if we treat edges as undirected, and there are no multiple edges.

Output

Output a single line containing two integers separated by a space.

If at least one path can be selected, there exists an optimal selection scheme that maximizes the score, and the optimal score can be represented as p/q where p and q are coprime integers and $q > 0$. In this case, output p and q. Otherwise (if it is impossible to select any paths), output "`-1 -1`".

Examples

standard input	*standard output*
5 5 1 1 11 21 1211 1 2 4 1 3 11 2 5 9 3 4 12 4 5 13	6 1
9 16 1 1 10 100 1000 10000 100000 1000000 10000000 1 2 2 1 3 3 2 3 5 2 5 7 3 4 11 3 5 13 3 6 17 3 7 19 4 7 23 5 6 29 5 8 31 6 7 37 6 8 41 6 9 43 7 9 47 8 9 53	221 3

Note

The selected doubling paths are $p_1 = ((1, 2))$ and $p_2 = ((2, 5))$.

Problem F. Find And Modify

Input file:	*standard input*
Output file:	*standard output*
Time limit:	12 seconds
Memory limit:	1024 mebibytes

You are given a permutation $a_1, \ldots, a_n$. You need to maintain a sequence $b_1, \ldots, b_n$ initialized by zeroes. Process m operations of the following form:

- Modification operation: given ℓ and r, for each pair (i, j) such that $\ell \le i \le j \le r$ and $a_i \le a_j$, increment b_j by 1;
- Query operation: given x, return b_x.

Input

The first line of input contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$).

The second line contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le n$; all a_i are distinct).

Each of the next m lines consists of integers and has either the form "`1` ℓ r" for a modification operation or the form "`2` x" for a query operation ($1 \le \ell \le r \le n$; $1 \le x \le n$).

You may assume that the input contains at least one query operation.

Output

For each query operation, output one line containing an integer that represents the answer.

Example

standard input	*standard output*
8 10 5 4 8 7 1 6 3 2 1 2 5 2 8 1 2 8 1 7 8 2 4 2 1 2 6 2 4 1 8 8 2 4	0 4 0 3 4 4

Problem G. Generate Optimal Key

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 megabytes

A *pattern* is defined as a binary string of length L. Given n patterns, we want to find a *key* which is also a binary string of length L.

For a given pattern and a given key, the *error value* is defined as the number of positions where the key differs from the pattern. For example, if the pattern is `101`, and the key is `000`, then the first and third positions are different, so the error value is 2.

We want to find a key such that the sum of the error values for this key and the given n patterns is minimized. Additionally, there are m distinct forbidden keys, and the key we find cannot be one of the forbidden keys.

Input

The first line of input contains three integers, n, m, and L ($1 \le n \le 1000$; $1 \le m \le \min(1000, 2^L - 1)$; $1 \le L \le 1000$).

Each of the following n lines represents a pattern and contains a binary string of length L.

Each of the following m lines represents a forbidden key and contains a binary string of length L. All forbidden keys are distinct.

Output

Output a single integer: the minimum sum of error values for a non-forbidden key and the n given patterns if we select the key optimally.

Examples

standard input	*standard output*
4 1 4 0000 1000 1100 1010 1000	5
2 4 4 0000 0000 0000 1000 0100 0010	2

Problem H. Huge Sequences

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 megabytes

Given three sequences $a_1, \ldots, a_n$, $b_1, \ldots, b_n$, and $c_1, \ldots, c_n$, define the *value* of the interval $[\ell, r]$ as the product of three factors:

- the bitwise AND of $(a_\ell, \ldots, a_r)$,
- the bitwise OR of $(b_\ell, \ldots, b_r)$, and
- the greatest common divisor of $(c_\ell, \ldots, c_r)$.

There are m queries. Each query provides an interval $[\ell, r]$, and asks for the sum of the values of all intervals $[\ell', r']$ such that $\ell \le \ell' \le r' \le r$. As the answer may be large, find it modulo 2^{32}.

Input

The first line of input contains two integers n and m ($1 \le n \le 10^6$; $1 \le m \le 5 \cdot 10^6$).

The second line contains n integers $a_1, \ldots, a_n$.

The third line contains n integers $b_1, \ldots, b_n$.

The fourth line contains n integers $c_1, \ldots, c_n$.

The constraints are: $1 \le a_i, b_i, c_i \le n$.

Each of the following m lines contains two integers ℓ and r and represents a query ($1 \le \ell \le r \le n$).

Output

Output m lines, each containing one integer: the corresponding answer modulo 2^{32}.

Examples

standard input	*standard output*
5 3 3 3 1 1 1 2 1 3 2 2 4 5 3 4 4 1 2 2 5 4 5	48 63 24
1 1 1 1 1 1 1	1

Problem I. Integration of Lines and Poker

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 megabytes

You are given a board of size $n \times m$, with the top left corner at $(1, 1)$ and the bottom right corner at (n, m). There are k different colors of pieces, numbered from 1 to k. Initially, each cell contains one piece.

There are q operations in total. Each operation involves swapping two adjacent (up, down, left, right) pieces. After this, every continuous sequence of at least 3 pieces of the same color in the same row or column will be eliminated.

After elimination, all pieces will fall down due to gravity, creating empty spaces at the top of the column. Once all pieces have fallen, if there are still pieces that can be eliminated, a chain reaction will occur, continuing to eliminate until no more eliminations are possible. A single elimination followed by a fall is called "one round of elimination", and we can define the "number of rounds" of elimination triggered by one operation.

Some pieces have special properties that trigger special effects when eliminated. There are a total of 6 types of special properties:

1. Eliminating will remove all pieces in the same row;
2. Eliminating will remove all pieces in the same column;
3. Eliminating will remove all pieces in the same row and column;
4. Eliminating will remove all pieces in a 3×3 square centered on it;
5. Eliminating will remove all pieces in a 5×5 square centered on it;
6. Eliminating will remove all pieces of the same color.

Triggering a piece's special effect may also trigger other pieces' special effects, but these are all triggered within the same round of elimination, as a chain reaction, before falling due to gravity.

In the game, each operation must be valid, meaning the two positions involved in the operation must be adjacent and not empty, and the operation must lead to a piece elimination. If an operation is not valid, it is skipped. The game ends after all q operations are completed.

The *main color* of a valid operation is defined as the color that is directly eliminated by the swap (not including those triggered by special effects or falling). It is easy to see that a valid operation has 1 or 2 main colors.

In the game, players aim to score as many points as possible through their operations. The scoring rules consist of 5 types: elimination score + chain score + combination score + pattern score + endgame score.

- **Elimination score:** For each valid operation, the elimination score for the i-th round of elimination is i times the sum of the color numbers of all pieces eliminated in that round.
- **Chain score:** If the total number of elimination rounds for a valid operation is x, the chain score is $80 \cdot (x - 1)^2$.
- **Combination score:** In a certain round of elimination, consider only the eliminations caused by "at least 3 consecutive pieces of the same color in the same row or column" (disregard eliminations caused by special effects). If a certain eliminated same-color block connected by side has size x, the combination score is $50(x - 3)^2$. Some examples: 4 same-color pieces in a line give a combination score of 50; 5 same-color pieces forming a line, cross, or T-shape give a score of 200; a 2×3 square of same-color pieces (which may appear after a fall) gives a score of 450.

- **Pattern score:** Every 5 valid operations, a pattern score is calculated based on the main colors of the previous 5 valid operations (if an operation has multiple main colors, take the one that can yield the maximum score according to the following rules):
  - High card: All 5 colors are different, score is 50 + the highest color number among all cards;
  - One pair: 2 pieces of the same color + 3 pieces of different colors, score is 100 + the pair's color number $\times 2$;
  - Two pairs: 2 same-color pairs + 1 other color, score is 200 + the larger color number of the pairs $\times 2$ + the smaller color number of the pairs;
  - Three of a kind: 3 pieces of the same color + 2 different colors, score is 300 + the three of a kind's color number $\times 3$;
  - Full house: 3 pieces of one color + 2 pieces of another color, score is 500 + the color number of the three of a kind $\times 3$ + the color number of the pair;
  - Four of a kind: 4 pieces of the same color + 1 other color, score is 750 + the four of a kind's color number $\times 5$;
  - Five of a kind: All 5 pieces are the same color, score is 1000 + the five of a kind's color number $\times 10$.
- **Endgame score:** If all q operations are valid, 1000 bonus points are awarded at the end. If the board is completely cleared at the end of the game, 10 000 bonus points are awarded.

Given the initial state of a game and each operation performed by the player, you need to calculate the player's total score.

Input

The first line of input contains four integers: n, m, k, and q ($2 \le n, m \le 50$; $m + n > 4$; $2 \le k \le 100$; $1 \le q \le 1000$).

The next n lines, each containing m integers $a_{i,j}$, represent the initial state of the pieces' colors from top to bottom and from left to right ($1 \le a_{i,j} \le k$).

The following n lines, each containing m integers $b_{i,j}$, represent the initial state of the pieces' special effects, as described in the problem statement. Specifically, $b_{i,j} = 0$ indicates no special effect ($0 \le b_{i,j} \le 6$).

Each of the next q lines contains four positive integers: $x_{i,1}$, $y_{i,1}$, $x_{i,2}$, and $y_{i,2}$. They indicate the swap of pieces at coordinates $(x_{i,1}, y_{i,1})$ and $(x_{i,2}, y_{i,2})$ ($1 \le x_{i,1}, x_{i,2} \le n$; $1 \le y_{i,1}, y_{i,2} \le m$).

It is guaranteed that the initial state has no direct elimination situations.

Output

Output a single line with an integer representing the total score at the end of the game.

Examples

standard input	*standard output*
8 8 5 5 1 1 5 1 5 4 5 3 2 1 2 2 5 4 3 2 1 4 1 4 2 1 5 4 2 1 5 5 2 1 4 4 2 3 5 2 3 4 2 2 4 2 4 3 3 2 4 5 1 3 4 3 5 2 4 3 3 4 2 5 2 1 1 2 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 3 1 0 0 0 0 3 0 0 0 0 0 0 0 0 0 1 4 3 2 4 2 5 4 5 5 7 2 7 3 8 5 8 6 6 7 6 8	11692

standard input	*standard output*
8 8 5 8 1 1 5 1 5 4 5 3 2 1 2 2 5 4 3 2 1 4 1 4 2 1 5 4 2 1 5 5 2 1 4 4 2 3 5 2 3 4 2 2 4 2 4 3 3 2 4 5 1 3 4 3 5 2 4 3 3 4 2 5 2 1 1 2 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 3 1 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 1 1 2 2 3 2 4 2 3 2 3 3 4 2 4 3 5 4 5 5 7 2 7 3 8 5 8 6 6 7 6 8	684

standard input	*standard output*
5 5 2 1 1 1 2 1 1 1 1 2 1 1 2 2 1 2 2 1 1 2 1 1 1 1 2 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 4 3	3023

Note

The sums of the first three types of scores after each operation are: 315, 417, 429, 435, 482. After the 5-th operation, the pattern score is calculated, with the optimal pattern being (1 2 4 2 4), yielding a score of $200 + 4 \cdot 2 + 2 \cdot 1 = 210$. At the end of the game, we obtain both types of endgame bonuses, resulting in the total score of 11 692.