Problem A. Words

Input file: `standard input`
Output file: `standard output`
Memory limit: 256 Mebibytes

Let h be a function acting on strings composed of the digits `0` and `1`. The function h transforms the string w by replacing (independently and concurrently) every digit `0` with `1` and every digit `1` with the string ,,10". For example $h($,,1001"$)=$,,101110", $h($,,"$)=$,," (i.e. h assigns an empty string to the empty string). Note that h is an injection, or a one-to-one function. By h^k we denote the function h composed with itself k times. In particular, h^0 is the identity function $h^0(w)=w$.

We are interested in the strings of the form $h^k($,,0"$)$ for $k=0,1,2,3,\ldots$ This sequence begins with the following strings:

,,0", ,,1", ,,10", ,,101", ,,10110", ,,10110101".

We call the string x a *substring* of the string y if it occurs in y as a contiguous (i.e. one-block) subsequence. A sequence of integers $k_1, k_2, \ldots, k_n$ is given. Your task is to check whether a string of the form

$$h^{k_1}(\text{,,0"}) \cdot h^{k_2}(\text{,,0"}) \cdots h^{k_n}(\text{,,0"})$$

is a substring of $h^m($,,0"$)$ for some m.

Input

The first line of the standard input contains a single integer t, $1 \le t \le 13$, denoting the number of test units. The first line of each test unit's description contains one integer n, $1 \le n \le 100\,000$. The second line of each description holds n non-negative integers $k_1, k_2, \ldots, k_n$, separated by single spaces. The sum of the numbers in the second line of any test unit description does not exceed 10 000 000.

Output

Your programme should print out t lines to the standard output, one for each test unit. Each line corresponding to a test unit should contain one word: `TAK` (*yes* in Polish — if $h^{k_1}(\text{,,0"}) \cdot h^{k_2}(\text{,,0"}) \cdots h^{k_n}(\text{,,0"})$ is a substring of $h^m(\text{,,0"})$ for some m in that test unit, or `NIE` (*no* in Polish) otherwise.

Example

standard input	standard output
2 2 1 2 2 2 0	TAK NIE

Explanation of the example: The string from the first test unit is ,,110" — it is a substring of $h^4($,,0"$)=$,,10110" for example. In the second test unit there is a string ,,100", which is not a substring of $h^m($,,0"$)$ for any m.

Problem B. Ice Skates

Input file: `standard input`
Output file: `standard output`
Memory limit: 256 Mebibytes

Byteasar runs a skate club. Its members meet on a regular basis and train together, and they always use the club's ice-skates. The skate sizes are (by convention) numbered from 1 to n. Naturally, each club member has some foot size, but that is not all to it! Skaters have skate size tolerance factor d: a skater with foot size r can wear skates with sizes from r up to $r+d$. It should be noted, though, that *no skater ever wears two skates of different size simultaneously.*

To supply the club, Byteasar bought k pairs of ice-skates of each size, i.e. from 1 to n. As time passes, some people join the club, just as some established members leave it. Byteasar worries if he will have enough skates of appropriate size for every member at each training.

We assume that initially the club has no members at all. Byteasar will give you a sequence of m events of the following form: x (new) members with foot size r have joined/left the club. Right after each such event Byteasar wants to know whether he has enough skates of appropriate size for every club member. He asks you to write a programme that will check it for him.

Input

The first line of the standard input contains four integers n, m, k and d ($1 \le n \le 200\,000$, $1 \le m \le 500\,000$, $1 \le k \le 10^9$, $0 \le d < n$), separated by single spaces, that denote, respectively: maximum skate size, number of events, number of skate pairs of each size Byteasar initially bought, and the skate size tolerance factor. The following m lines contain the sequence of m events, one per line. The $(i+1)$-th line (for $1 \le i \le m$) holds two integers: r_i and x_i ($1 \le r_i \le n-d$, $-10^9 \le x_i \le 10^9$), separated by a single space. If $x_i \ge 0$, it means that x_i new members with foot size r_i each have just joined the club. And if $x_i < 0$, it means that x_i members with foot size r_i each have just left the club. You may assume the sequence is sensible, i.e. someone who never joined the club cannot leave it.

Output

Your programme should print out m lines to the standard output. The i-th line (for $1 \leq i \leq m$) should either contain the word `TAK` (Polish for *yes*), or the word `NIE` (Polish for *no*), depending on whether Byteasar has the skates of appropriate size for every club member right after the i-th event.

Example

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

After all events from the input sequence took place, there are three club members who can wear skates of size 1 or 2, two members who can wear sizes 2 or 3, and three members who can wear sizes 3 or 4. With such list of members two pairs of ice-skates of sizes 1, 2, 3, and 4 each suffice:

- two members get skates of size 1;
- the skates of size 2 are given to the member who can wear size 1 or 2, and another one who can wear size 2 or 3;
- the skates of size 3 are given to the member who can wear size 2 or 3, and another one who can wear size 3 or 4;
- the remaining two members receive skates of size 4.

Problem C. Elephants

Input file:	`standard input`
Output file:	`standard output`
Memory limit:	256 Mebibytes

A parade of all elephants is to commence soon at the Byteotian zoo. The zoo employees have encouraged these enormous animals to form a single line, as the manager wills it to be the initial figure of the parade.

Unfortunately, the manager himself came to the parade and did not quite like what he saw — he had intended an entirely different order of the elephants. Therefore he enforced his ordering, claiming the animals would seem most majestic this way, and made the employees reorder the elephants accordingly.

As a pack of moving elephants can wreak havoc, the employees decided to have them rearranged by swapping one pair at a time. Luckily the animals need not stand next to each other in order to swap positions in the line. Making an elephant move, however, is not as easy as it sounds. In fact, the effort one has to put into it is proportional to the animal's mass. Hence, the effort involved in swapping a pair of elephants of respective masses m_1 and m_2 can be estimated by $m_1 + m_2$. What is the minimum effort involved in rearranging the elephants according to manager's will?

Write a programme that:

- reads from the standard input the masses of all elephants from the zoo, along with their current and desired order in the line,
- determines a sequence of elephant swaps leading from the initial to the desired order of animals in the line, such that this sequence minimises the summary effort involved in all the swaps,
- prints out the summary effort on the standard output.

Input

The first line of the standard input contains a single integer n ($2 \leq n \leq 1\,000\,000$) denoting the number of elephants in the zoo. We assume that the elephants are numbered from 1 to n to simplify things. The second line holds n integers m_i ($100 \leq m_i \leq 6\,500$ dla $1 \leq i \leq n$) separated by single spaces and denoting the masses of respective elephants (in kilogrammes).

The third line of input contains n pairwise different integers a_i ($1 \leq a_i \leq n$) separated by single spaces and denoting the numbers of successive elephants in the initial ordering. The fourth line holds n pairwise different integers b_i ($1 \leq b_i \leq n$) separated by single spaces and denoting the numbers of successive elephants in the ordering desired by the zoo manager. You may assume that the sequences (a_i) and (b_i) differ.

Output

The first and only line of the standard output should contain a single integer denoting the minimum summary effort involved in reordering the elephants from the order represented by the sequence to the one represented by (b_i).

Example

standard input	standard output
6 2400 2000 1200 2400 1600 4000 1 4 5 3 6 2 5 3 2 4 6 1	11200

One of the optimal rearrangements consists of swapping the following pairs of elephants:

- 2 and 5 — effort involved: $2\,000 + 1\,600 = 3\,600$, order achieved: `1 4 2 3 6 5`,

- 3 i 4 — effort involved: $1\,200 + 2\,400 = 3\,600$, order achieved: `1 3 2 4 6 5`,
- 1 i 5 — effort involved: $2\,400 + 1\,600 = 4\,000$, order achieved: `5 3 2 4 6 1`, which is the one desired.

Problem D. Island

Input file:	`standard input`
Output file:	`standard output`
Memory limit:	256 Mebibytes

Byteasar is the king of Byteotia, an island in The Ocean of Happiness. The island is a convex shape, and all the towns of Byteotia are located on the shore. One of these towns is Byteburg, the famous capital of Byteotia. Every pair of towns is connected by a road that goes along the line segment between the towns. Some roads that connect different pairs of towns intersect — there is a crossroad at each such intersection.

Bitratio, Byteasar's rival to the throne, had hatched a sordid plot. While Byteasar was travelling from the capital to an **adjacent** town, Bitratio's people seized Byteburg. Now Byteasar has to return to Byteburg as soon as possible in order to restore his rule. Unfortunately, some of the roads are controlled by Bitratio's guerrilla. Byteasar cannot risk the use of such roads, he can however cross them at the crossroads. Needless to say, he has to travel along the roads and hence turn only at the crossroads, for otherwise the journey would take far too long.

Byteasar's loyal servants have informed him which roads are safe. Byteasar believes your loyalty, and thus entrusts you with a task to find the shortest safe route from the town he is currently in to Byteburg.

Input

In the first line of the standard input two integers n and m are given ($3 \leq n \leq 100\,000$, $1 \leq m \leq 1\,000\,000$), separated by a single space, that denote respectively: the number of towns and the number of roads controlled by Bitratio's guerrilla. Let us number the towns from 1 to n starting from Byteburg and moving clockwise along the shore. Bytesar is currently in the town no. n. Each of the following n lines holds a pair of integers x_i and y_i ($-1\,000\,000 \leq x_i, y_i \leq 1\,000\,000$), separated by a single space, that denote the town's no i coordinates.

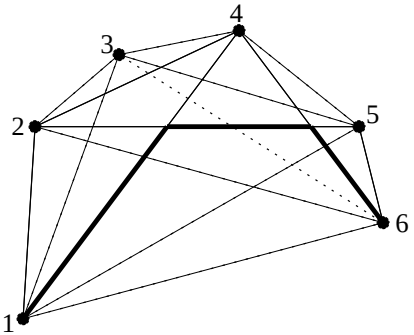
Each of the following m lines contains a pair of integers a_j and b_j ($1 \leq a_j < b_j \leq n$). Such pair means that the road connecting the towns a_j and b_j is controlled by Bitratio's guerrilla. Every such pair is unique. You can assume that in every test data set Byteasar can get to Byteburg.

Output

Your programme is to print out one floating point number to the standard output: the length of the shortest safe route leading from the town no. n to Byteburg. The absolute difference between the number returned and the correct one has to be at most 10^{-5}.

Example

standard input	standard output
6 9 -12 -10 -11 6 -4 12 6 14 16 6 18 -2 3 4 1 5 2 6 2 3 4 5 3 5 1 3 3 6 1 6	42.0000000000



The route that Byteasar should follow leaves town no. 6 in the direction of town no. 4, then turns to the road connecting towns no. 2 and 5, and finally goes along the road connecting Byteburg with the town no. 4.

Problem E. Bytie-boy's Walk

Input file:	`standard input`
Output file:	`standard output`
Memory limit:	256 Mebibytes

Bytie-boy is one of the youngest habitats of Byteburg. Let the fact that he has only just learnt to read and write speak for his age. He is, however, old enough to go school by himself. Every

morning he leaves home and comes by all his friends; the whole group goes to school together only when everyone is joined.

One day the teacher asked Bytie to prepare a list of the streets Bytie walks on his way to school, and to read it loud during the very next class. It soon turned out that the list is very long; so long in fact that it could take a vast amount of paper. Therefore it was decided that Bytie-boy would write down only the first letter from the name of every street he walks. The streets Byteburg are, with no exception, one-way, and each one connects two different crossings.

While reading the list of letters, Bytie makes a pause only at the spots at which he picked up a friend. Thus each part of his walk can be treated as a single word. Reading is still somewhat difficult for the boy: sometimes he reads from right to left instead of left-to-right. So he may read the word *milk* as *milk*, but also as *klim*. As Bytie's parents are aware of his problems, they decided to aid him by finding a route whose every fragment remains the same whether it is read left-to-right or right-to-left. For obvious reasons, the parents would also like to keep each fragment (word) as short as possible. They ask you for help.

Write a programme that

- reads the city's description from the standard input,
- determines a route from Bytie's house to the school such that each of its fragments is as short as possible and its description reads the same left-to-right and right-to-left,
- prints out the description (with fragments suitably separated) to the standard output.

Input

The first line of the standard input contains two integer, n and m, separated by a single space ($2 \le n \le 400$, $1 \le m \le 60\,000$). These denote, respectively, the number of crossings in Byteburg and the number of streets connecting them. The streets descriptions follow in the next m lines. Three numbers are given in the $(i+1)$-th line: x_i, y_i, c_i ($1 \le x_i \le n$, $1 \le y_i \le n$, $x_i \ne y_i$); these denote, respectively, the start of the street, the end of the street, and the first letter of the street's name, and are separated by single spaces. The letters are lowercase, English. Any two crossings are connected by at most one street per direction (at most one in one direction and at most one in the opposite). The following line contains a single integer d ($2 \le d \le 100$) denoting the number of crossings that Bytie-boy passes through on his way to school. The next line contains d integers s_i ($1 \le s_i \le n$), also separated by single spaces. These mean that Bytie lives by the crossing no. s_1, the school is by the crossing no. s_n, and along the way Bytie picks up **successively** his friends living by the crossings no. $s_2, s_3, \ldots, s_{n-1}$. Every two successive numbers of crossings on the list are distinct, but some two numbers on the list can be the same. Moreover, if it is not possible to get from one crossing to another complying with restrictions set in the task, Bytie takes a short cut and writes nothing on his sheet.

Output

The output should consist of $d-1$ lines. There should be a number r_i ans a sequence of characters w_i in the i-th line, separated by a single space. The number r_i denotes the minimum length of the route complying with the task's conditions that connects the crossings s_i and s_{i+1}, while w_i is the sequence of first letters of the streets in this rout. If there is no route between some two crossings that satisfies aforementioned conditions, -1 should be output. It is possible there are several possible sequences w_i. In that case print any.

Example

standard input	standard output
6 7 1 2 a 1 3 x 1 4 b 2 6 l 3 5 y 4 5 z 6 5 a 3 1 5 3	3 ala -1

Problem F. Mirror trap

Input file:	standard input
Output file:	standard output
Memory limit:	256 Mebibytes

King Byteasar's palace is haunted by a ghost. The spiteful claim it is the ghost of the late Byteasar's wife (who died recently in suspicious circumstances), for it takes great pleasure in looking at itself in the mirror, just as she did. No wonder that Byteasar would gladly get rid of this ghost!

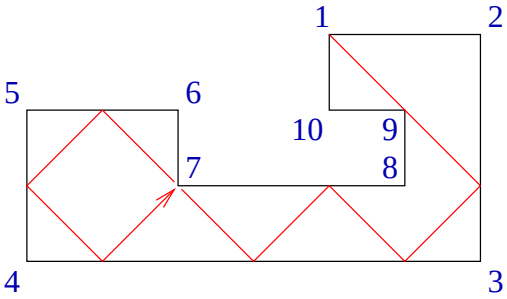
Byteasar has decided to plant a special *mirror trap* in one of the palace's chambers. It is a closed, well lit room whose every interior wall is covered with mirror. Furthermore, in its every corner a laser gun or a laser detector can be set up. As soon as the ghost crosses one of the laser beams an alarm will go off, summoning Byteasar's ghost vanquishers team (as a royal team, they prefer not to be colloquially called 'ghost busters'), who will surely dispose of the ghost in no time.

The chamber, in which the mirror trap is being installed, is in the shape of a polygon whose consecutive sides are perpendicular. Moreover, the length of every side (measured in metres) is integral. Should a laser gun be installed in some corner, its beam has to be pointed along the bisector of the angle formed by the walls meeting in this corner (in a plane parallel to the floor surface). Obviously, if the laser beam encounters a mirror, it is reflected, obeying the regular reflection law: the angle which the incident ray makes with the normal is equal to the angle which the reflected ray makes to the same normal. This angle, as you may notice, is always 45

degrees due to the chamber's architecture. A beam cast along the bisector into a corner where a detector is placed, is completely absorbed by it. On the other hand, a beam cast along the bisector into corner without a detector (though possibly with a laser gun) is reflected back (by 180 degrees).

A Maximum number of laser guns complemented with laser detectors should be placed in the room in such a way that each laser's beam is eventually absorbed by some detector, no two lasers' beams are absorbed by the same detector and each detector absorbs some laser's beam. Also, a laser gun and a detector may not be placed in the same corner.

In the picture you can find an example of a mirror trap with the laser beam going from corner 1 to corner 7.



Write a programme that:

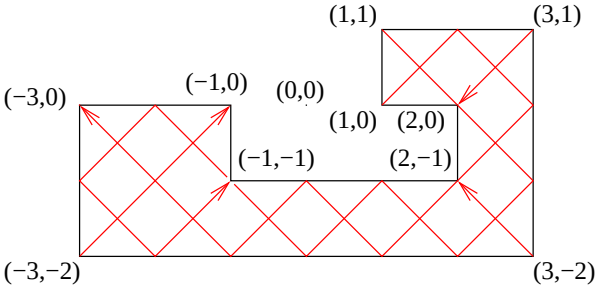
- reads from the standard input the the mirror trap's shape description,
- determines the way of setting up the maximum possible number of laser guns and detectors satisfying the aforementioned conditions,
- writes out the result to the standard output.

Input

In the first line of the standard input there is a single integer n ($4 \leq n \leq 100\,000$). It is exactly the number of walls of the mirror trap. Each of the following n lines contains two integers: x_i and y_i denoting the i^{th} corner's coordinates ($1 \leq i \leq n$, $-1\,000\,000 \leq x_i, y_i \leq 1\,000\,000$), separated by a single space. Every two successive corners are connected by a wall parallel to one of the axes. No two walls share a common point, except two successive walls, which have a common endpoint (some corner). The successive corners of the room are given in the clockwise order (ie. if one walks the room along the walls, one has the interior on the right-hand side). The summary length of all the walls does not exceed 300 000.

Output

Your programme should write a single integer m in the first line of the standard output, denoting the maximum number of laser gun-detector pairs that may be set up in the mirror trap. In each of the following m lines there should be a pair of integers a_i and b_i separated by a single space, where a_i denotes the number of the corner in which the laser gun should be placed while b_i — the number of the corner in which the corresponding laser detector should be placed, satisfying $1 \leq a_i, b_i \leq n$. If more than one optimal setup exists, pick one arbitrarily.



Example

standard input	standard output
10 1 1 3 1 3 -2 -3 -2 -3 0 -1 0 -1 -1 2 -1 2 0 1 0	5 10 5 1 7 2 9 3 8 4 6

An exemplary deployment of laser guns and detectors in the mirror trap is depicted in the figure.

Problem G. Ticket Inspector

Input file:	standard input
Output file:	standard output
Memory limit:	256 Mebibytes

Byteasar works as a ticket inspector in a Byteotian National Railways (BNR) express train that connects Byteburg with Bitwise. The third stage of the BNR reform[1] has begun. In

[1]The never ending saga of BNR reforms and the Bitwise hub was presented in the problems *Railways* from the third stage of XIV Polish OI and *Station* from the third stage of XV Polish

particular, the salaries system has already been changed. For example, to encourage Byteasar and other ticket inspectors to efficient work, their salaries now depend on the number of tickets (passengers) they inspect. Byteasar is able to control all the passengers on the train in the time between two successive stations, but he is not eager to waste his energy in doing so. Eventually he decided he would check the tickets exactly k times per ride.

Before setting out, Byteasar is given a detailed summary from which he knows exactly how many passengers will travel from each station to another. Based on that he would like to choose the moments of control in such a way that the number of passengers checked is maximal. Obviously, Byteasar is not paid extra for checking someone multiple times — that would be pointless, and would only disturb the passengers. Write a programme that will determine for Byteasar when he should check the tickets in order to maximise his revenue.

Input

In the first line of the standard input two positive integers n and k ($1 \leq k < n \leq 600$, $k \leq 50$) are given. These are separated by a single space and denote, respectively, the number of stations en route and the number of controls Byteasar intends to make. The stations are numbered from 1 to n in the order of appearance on the route.

In the next $n-1$ lines the summary on passengers is given. The $(i+1)$-th line contains information on the passengers who enter the train on the station i — it is a sequence of $n-i$ nonnegative integers $x_{i,i+1}, x_{i,i+2}, \ldots, x_{i,n}$ separated by single spaces. The number $x_{i,j}$ (for $1 \leq i < j \leq n$) denotes the number of passengers who enter the train on station i and leave it on station j. The total number of passengers (i.e. the sum of all $x_{i,j}$) does not exceed 2 000 000 000.

Output

Your programme should print out (in a single line) an increasing sequence of k integers from the interval from 1 to $n-1$ separated by single spaces to the standard output. These numbers should be the numbers of stations upon leaving which Byteasar should control the tickets.

Example

standard input	standard output
7 2 2 1 8 2 1 0 3 5 1 0 1 3 1 2 2 3 5 6 3 2 1	2 5

OI. Their knowledge, however, is not required at all in order to solve this problem.

Byteasar controls 42 tickets.

Problem H. Pebbles

Input file:	standard input
Output file:	standard output
Memory limit:	256 Mebibytes

Johny and Margaret are playing "pebbles". Initially there is a certain number of pebbles on a table, grouped in n piles. The piles are next to each other, forming a single row. The arrangement of stones satisfies an additional property that each pile consists of at least as many pebbles as the one to the left (with the obvious exception of the leftmost pile). The players alternately remove any number of pebbles from a single pile of their choice. They have to take care, though, not to make any pile smaller than the one left to it. In other words, the piles have to satisfy the initial property after the move as well. When one of the players cannot make a move (i.e. before his move there are no more pebbles on the table), he loses. Johny always starts, to compensate for Margaret's mastery in this game.

In fact Margaret is so good that she always makes the best move, and wins the game whenever she has a chance. Therefore Johny asks your help — he would like to know if he stands a chance of beating Margaret with a particular initial arrangement. Write a programme that determines answers to Johny's inquiries.

Input

In the first line of the standard input there is a single integer u ($1 \leq u \leq 10$) denoting the number of initial pebble arrangements to analyse. The following $2u$ lines contain descriptions of these arrangements; each one takes exactly two lines.

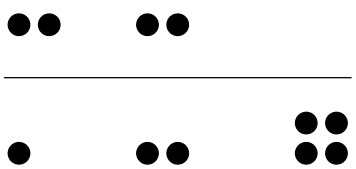
The first line of each description contains a single integer n, $1 \leq n \leq 1\,000$ — the number of piles. The second line of description holds n non-negative integers a_i separated by single spaces and denoting the numbers of pebbles in successive piles, left to right. These numbers satisfy the following inequality $a_1 \leq a_2 \leq \cdots \leq a_n$. The total number of pebbles in any arrangement does not exceed 10 000.

Output

Precisely u lines should be printed out on the standard output. The i-th of these lines (for $1 \leq i \leq u$) should hold the word `TAK` (*yes* in Polish), if Johny can win starting with the i-th initial arrangement given in the input, or the word `NIE` (*no* in Polish), if Johny is bound to lose that game, assuming optimal play of Margaret.

Example

standard input	standard output
2 2 2 2 3 1 2 4	NIE TAK



Problem I. Fire extinguishers

Input file:	standard input
Output file:	standard output
Memory limit:	256 Mebibytes

Byteasar has had a new palace built. It consists of n chambers and $n-1$ corridors connecting them. Each corridor connects exactly two chambers. The rooms are numbered from 1 to n. There is only a single entrance to the palace, which leads to chamber no. 1. For each chamber there is exactly one route leading to it from the entrance, without turning back on the way. In other words, the chambers and the corridors form a *tree* — a connected acyclic graph.

The fire marshal who is to approve the building demands placing fire extinguishers inside. The following are his exact requirements:

- The fire extinguishers should be placed in (some) chambers, and one chamber may store any number of extinguishers.
- Each chamber has to be assigned one fire extinguisher, though it may be stored in another chamber.
- Each fire extinguisher can be assigned to at most s different chambers.
- For each room its assigned extinguisher is within the range of k corridors.

Byteasar has a week spot for lavish palaces, so it is no surprise he has very little money now, right after completion of another splendid palace. Therefore he is interested in the minimum number of fire extinguishers sufficient for satisfying fire marshal's demands.

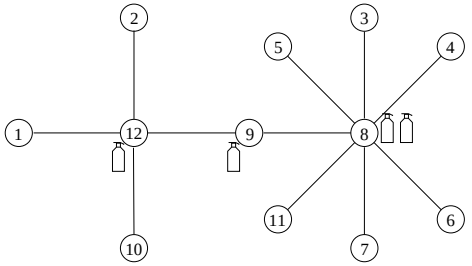
Input

The first line of the standard input contains three integers n, s and k separated by single spaces, $1 \le n \le 100\,000$, $1 \le s \le n$, $1 \le k \le 20$. Each of the following $n-1$ lines holds two integers separated by a single space. Line no. $i+1$ contains the numbers $1 \le x_i < y_i \le n$ denoting the corridor connecting chambers no. x_i and y_i.

Output

The first and only line of the standard output is to hold one integer — the minimum number of fire extinguishers that have to be installed in palace.

Example



standard input	standard output
12 3 1 1 12 3 8 7 8 8 9 2 12 10 12 9 12 4 8 5 8 8 11 6 8	4

Problem J. Trains

Input file:	standard input
Output file:	standard output
Memory limit:	256 Mebibytes

The Trains of Colour Parade begins tomorrow in Byteotia. Intense preparations are already in progress at the station's auxiliary tracks. There are n parallel tracks at the station, numbered from 1 to n. The train no. i occupies the i^{th} track. Every train consists of l cars and each car is painted with one of 26 colours (denoted by non-capital letters of the English alphabet). We

say that two trains *look the same*, if their corresponding cars are painted the same colour.

Throughout the parade a crane will switch places of certain pairs of cars. The real parade, however, will take place tomorrow. Today the train dispatcher, Byteasar, watched the general rehearsal closely. He even wrote down the sequence of car swaps.

Byteasar particularly dislikes many trains looking the same. For each train p he would like to calculate the maximum number of trains that at some moment look the same as the train p at the very same moment.

Write a programme that:

- reads descriptions of the trains occupying tracks and the sequence of car swaps,
- for each train determines the maximum number of trains that look the same as it at certain moment,
- writes out the result.

Input

The first line of the input contains three integers n, l and m ($2 \leq n \leq 1\,000$, $1 \leq l \leq 100$, $0 \leq m \leq 100\,000$), denoting respectively the number of trains, their common length and the number of car swaps. The following n lines contain descriptions of the trains on successive tracks. The k^{th} line consists of l small letters of the English alphabet denoting the colours of successive cars of the k^{th} train. Then m lines describing the car swaps follow, in the order of the swaps. The r^{th} line contains four integers p_1, w_1, p_2, w_2 ($1 \leq p_1, p_2 \leq n$, $1 \leq w_1, w_2 \leq l$, $p_1 \neq p_2$ or $w_1 \neq w_2$) denoting the r^{th} car swap—the car no. w_1 of the train no. p_1 is swapped with the car no. w_2 of the train no. p_2.

Output

Your programme should write out exactly n lines. The k^{th} line should contain one integer—the number of trains looking the same as the train no. k at certain moment.

Example

standard input	standard output
5 6 7 ababbd abbbbd aaabad caabbd cabaad 2 3 5 4 5 3 5 5 3 5 2 2 1 2 4 3 2 2 5 1 1 1 3 3 4 1 5 6	3 3 3 2 3

The figure presents the successive car swaps:

```
tor 1:  ababbd    ababbd    ababbd    ababbd    aaabbd    aaabbd    aaabbd    aaabbd
tor 2:  abbbbd    ababbd    ababbd    aaabbd    aaabbd    acabbd    acabbd    acabbd
tor 3:  aaabad -> aaabad -> aaabad -> aaabbd -> aaabbd -> aaabbd -> aaabbd -> aaabbd
tor 4:  caabbd    caabbd    caabbd    caabbd    cabbbd    cabbbd    cabbbd    dabbbd
tor 5:  cabaad    cabbad    caabbd    caabbd    caabbd    aaabbd    aaabbd    aaabbc

          (0)       (1)       (2)       (3)       (4)       (5)       (6)       (7)
```

The number of trains looking the same as either of the trains no. 1, 2 or 3 was maximal at time `(4)` (when all three looked the same). The number of trains looking the same as the train no. 5 was maximal at time `(5)` and `(6)`. The number of trains looking the same as the train no. 4 was maximal at time `(2)`.