Problem A. Sorcerers of the Round Table

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 mebibytes

The time has come again for the sorcerers of the round table to hold a privy council. Like so many times before, they began to squabble even before the agenda was presented: the order of the seats around the table proved controversial enough. There are n sorcerers partaking in the council, each one uniquely identified by the height of their pointed hat; the hats' heights are pairwise different integers ranging from 1 to n (the taller the hat, the more senior the sorcerer). For the sake of aesthetics, the heights of hats of two sorcerers sitting next to each other should differ by at most p.

Be warned that some sorcerers are not too fond of others – if a sorcerer a dislikes the sorcerer b, then the sorcerer b may not sit immediately to the right of the sorcerer a. We assume that the chairman (with the hat of height n) has already sat down by the table. In how many ways can the remaining sorcerers be arranged around the table?

Input

The first line of the standard input contains three integers, n, k, and p ($1 \le n \le 1\,000\,000$, $0 \le k \le 100\,000$, $0 \le p \le 3$), separated by single spaces, that specify the number of sorcerers, the number of dislikes among them, and the maximum permissible difference in the height of adjacent hats, respectively.

The k lines that follow contain ordered pairs: the i-th of these lines contains two integers, a_i and b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$), separated by a single space, specifying that the sorcerer wearing a hat of height a_i dislikes the sorcerer wearing the hat of height b_i. Each such ordered pair appears at most once in the input.

Output

The first and only line of the standard output should give an integer equal to the remainder of the division of the number of possible arrangements of the sorcerers by $10^9 + 7$.

Examples

standard input	standard output

5 2 3
1 3
5 4
	6

Problem B. Movie-goer

Input file:	*standard input*
Output file:	*standard output*
Time limit:	9 seconds
Memory limit:	256 mebibytes

Byteasar is an avid movie-goer, so he was pleased to find out that his favorite studio cinema will be running a summer movie marathon. Each of the n days of the summer, one out of m movies will be screened. A marathon pass entitles one to see any number of movies, provided that the pass holder does not skip a day, i.e., skipping one invalidates the pass. One can, however, choose their first day freely.

Based on Internet reviews, Byteasar has assigned each of the m movies a score. Now he would like to use his summer pass so as to maximize the total score of the movies he sees. What makes this a challenging task is the fact that Byteasar cannot stand watching the same movie again. Watching a movie again not only bores him, but also deprives him of all the fond memories previously associated with the movie. Thus, in fact he wants to maximize the total score of the movies he will see exactly once.

Input

In the first line of the standard input, there are two integers, n and m ($1 \leq m \leq n \leq 1\,000\,000$), separated by a single space, that specify the length of the summer movie marathon and the number of movies, respectively. For ease of notation, we number the movies from 1 to m.

In the second line, there is a sequence of n integers, $f_1, f_2, \ldots, f_n$ ($1 \leq f_i \leq m$), separated by single spaces: f_i is the number of the movie screened on the i-th day of the marathon. In the third line, there is a sequence of m integers, $w_1, w_2, \ldots, w_m$ ($1 \leq w_j \leq 1\,000\,000$), separated by single spaces: w_j is the score of the movie no. j. Note that it may happen that some of the m movies will not be screened at all during the marathon.

Output

In the first and only line of the standard output, a single number should be printed, equal to the maximial total score of the movies Byteasar can watch exactly once with his summer movie marathon pass.

Examples

standard input	standard output

9 4
2 3 1 1 4 1 2 4 1
5 3 6 6
	15

Problem C. Speed Reading Course

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

Byteasar has enrolled in a speed reading course, which has taught him many perception improving exercises. His favorite one is finding a pattern in a sequence of symbols. For this exercise, Byteasar has a computer generate a very long sequence of zeros and ones as follows. He chooses four integers n, a, b, and p such that n and a are coprime, and the computer generates a sequence $c_0, c_1, \ldots, c_{n-1}$, where $c_i = 0$ if and only if $(ai + b) \bmod n < p$. Finally, Byteasar comes up with another, shorter sequence of m symbols $w_0, w_1, \ldots, w_{m-1}$. Set up with these, his task is to find all occurrences of the shorter sequence in the one generated by the computer as quickly as possible. He has asked your help in writing a program that will verify if indeed he found all the occurrences.

Input

The first line of the standard input contains five integers, n, a, b, p, and m ($2 \leq n \leq 1\,000\,000\,000$, $1 \leq p, a, b, m < n$, $1 \leq m \leq 1\,000\,000$), separated by single spaces. The numbers a and n are coprime. In the second line, there is a word $w_0, w_1, \ldots, w_{m-1}$, consisting of m symbols, each either ‘0’ or ‘1’.

Output

The first and only line of the standard output should contain an integer equal to the number of occurrences of the sequence $w_0, w_1, \ldots, w_{m-1}$ in the sequence $c_0, c_1, \ldots, c_{n-1}$.

Examples

standard input	standard output
9 5 6 4 3 101	3

Problem D. Squares

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

We consider decompositions of positive integers into sums of *unique* squares of positive integers, called decompositions in short from now on. For example, the number 30 has two decompositions: $1^2 + 2^2 + 5^2 = 1^2 + 2^2 + 3^2 + 4^2 = 30$, whereas the number 8 has none.

Specifically, we are interested in how large the largest square in the decomposition of a given number n has to be. In other words, we want to determine the value $k(n)$, defined as the minimum over decompositions of n of the maximum integer (not its square!) in the decomposition. We assume that $k(n) = \infty$ if n cannot be decomposed. For example, $k(1) = 1$, $k(8) = \infty$, $k(30) = 4$, $k(378) = 12$, $k(380) = 10$.

We call an integer x *overgrown* if there is an integer $y > x$ such that $k(y) < k(x)$. It follows from the previous example that 378 is overgrown.

For a given integer n, you are to determine $k(n)$ and the number of overgrown integers no larger than n.

Input

In the first and only line of the standard input, there is a single integer n ($1 \leq n \leq 10^{18}$).

Output

Your program should print two integers, separated by a single space, to the standard output: first $k(n)$ and then the number of overgrown integers in the range from 1 to n. If $k(n) = \infty$, then '-' (dash or minus sign) should be printed instead of the first number.

Examples

standard input	standard output
30	4 15
8	- 5

Problem E. Gluttons

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

At the gala dinner at the end of this year's Byteotian Sweets Amateurs Get-Together, n gluttons sat at a round-table. Soon afterwards, n cakes were set on the table. The cakes differ in size, appearance, and flavor, but for the gourmands it is the caloric value (c_i for the i-th cake) that is of utmost importance. The table has been set in such a way that there is a cake between every pair of adjacent gluttons. Each glutton can choose one of the two cakes next to them. A cake claimed by only one glutton is theirs alone. However, if a cake is claimed by two gluttons, they have to share it equally.

Each glutton wants to maximize the caloric value they receive from their chosen cake, which is halved if they only half the cake. A glutton would be dissatisfied, were they to choose wrong, i.e., if they would have more calories, had they chosen otherwise (assuming others do not change their decisions). Help the gluttons make their choices so that none of them is dissatisfied.

Input

The first line of the standard input contains a single integer, n ($2 \le n \le 1\,000\,000$), that specifies the number of gluttons (and cakes). The second line contains a sequence of n integers, $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 1\,000\,000\,000$), separated by single spaces; the caloric value of the i-th cake is given by c_i. We assume that the i-th glutton (for $1 \le i < n$) can choose either the i-th or the $(i+1)$-th cake, whereas the n-th glutton can choose either the n-th or the first one.

Output

If the gluttons cannot choose their cakes so that each and every one of them is satisfied, then the first and only line of the standard output should contain only number -1.

Otherwise, the first and only line of the standard output should hold a sequence of n integers separated by single spaces; the i-th of these integers should specify the number of the cake to be chosen by the i-th glutton.

If there is more than one correct answer, your program can pick one of them arbitrarily.

Examples

standard input	standard output
5 5 3 7 2 9	2 3 3 5 1

Problem F. Highway Modernization

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	256 mebibytes

There are n towns connected by a dense road network in Byteotia. Unfortunately, most of these roads are of poor quality. Therefore, $n - 1$ modern highways have been built over the last few years. Every city can be reached from every other city by highways alone, but this luxury has its price – one has to pay a toll for the use of each highway.

But the citizens of Byteotia consider the tolls outrageously high. To calm the public opinion down, the minister for transport has decided to modernize the highway network. However, this year's budget allows for construction of a single new highway and a demolition of another one, which may be connecting the same two towns – the modernness alone is an improvement. (The spokesperson of the ministry did not explain why an old highway has to be demolished in order for a new one to be constructed.) The minister wants to choose the two highways, old and new, so that every pair of towns remains connected by highways and so that the maximum number of highways between any pair of cities be minimized. We call this scenario *optimistic*.

On the other hand, the chancellor of the exchequer would rather that the modernization paid for itself by contributing to the budget. Thus, while also insisting on the highway network remaining connected, he wants the maximum number of highways between any pair of cities to be maximized. We call this scenario *pessimistic*.

The rumors of both scenarios made their way to the press. A journalist named Byteasar is to write a piece on this subject. Being a rare diligent journalist, Byteasar wants to describe the most optimistic and the most pessimistic scenario of modernization. Help him by writing a program that will provide him with the data for his article.

Input

In the first line of the standard input, there is a single integer n ($3 \le n \le 500\,000$) that specifies the number of towns in Bytoetia. These are numbered from 1 to n. Next, $n - 1$ lines describing the highways follow. The i-th of these lines contains two integers, a_i and b_i ($1 \le a_i, b_i \le n$, $a_i \neq b_i$), separated by a single space, indicating that there is a highway linking the towns no. a_i and b_i.

Output

In the first line of the standard output, five integers, k, x_1, y_1, x_2, and y_2, should be written. These are to describe the optimistic scenario: the maximum number of highways between a pair of cities is k when the highway linking the towns x_1 and y_1 is demolished and a new highway linking the towns x_2 and y_2 is constructed. The second line should describe the pessimistic scenario in the same format. The towns that specify the endpoints of a highway (either to be demolished or constructed) can be given in either order. If more than one solution exists, your program can choose one of them arbitrarily.

Examples

standard input	standard output

6
1 2
2 3
2 4
4 5
6 5
	3 4 2 2 5
	5 2 1 1 6

Problem G. Car Washes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	256 mebibytes

Byteasar plans to place a bid in a tender for operating n car washes along the main express way in Byteotia. Before he does though, he would like to estimate the revenue he can attain.

To this end, he has commissioned a market research. The research results conclude that m potential customers are going to drive along the way. In particular, the i-th of them is going to drive along the segment between the washes a_i and b_i (inclusively), and would be interested in having their car washed if the price were at most c_i bythalers. Byteasar intends to set the price in each car wash independently. Assuming that each customer is a rational (and tidy!) agent, i.e., chooses the cheapest wash along their way, or none if all the washes exceed their budget, Byteasar wants to set the prices so as to maximize his total revenue.

Input

The first line of the standard input contains two integers, n and m ($1 \le n \le 50$, $1 \le m \le 4000$), separated by a single space, that specify the numbers of car washes and customers respectively. The washes are numbered from 1 to n. The m lines that follow describe the customers: the i-th of these contains three integers, a_i, b_i, and c_i ($1 \le a_i \le b_i \le n$, $1 \le c_i \le 500\,000$), separated by single spaces, which indicate that the i-th customer is driving along the segment between the washes a_i and b_i, and has a budget of c_i bythalers for car washing.

Output

The first line of the standard output should contain a single integer s that equals the maximum total revenue of Byteasar, expressed in bythalers. The second line should contain a price list that attains the revenue s (under the rational agents assumption), namely, a sequence of n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le 500\,000$), separated by single spaces, where p_i is the price in the i-th car wash. If more than one correct answer exists, your program can pick one out of those arbitrarily.

Examples

standard input	standard output

7 5
1 4 7
3 7 13
5 6 20
6 7 1
1 2 5
	43
	5 5 13 13 20 20 13

Problem H. Visits

Input file:	*standard input*
Output file:	*standard output*
Time limit:	15 seconds
Memory limit:	256 mebibytes

Byteasar is a remarkable man – over the last 21 years he was a postman, a banker, an ice-skater, and even a king! No wonder he has many friends. Or had, as his chronic profession and workplace changes unfortunately made him lose touch with many of the old friends... The time has come to change that! Bytasar is to start a grand tour of Byteotia to renew the old friendships.

There are n towns in Byteotia, connected via a network of $n - 1$ bidirectional roads. Our protagonist wants to visit each and every town, and has already fixed the order of these visits. From each town to the next on his itinerary, he will drive a car rented from BMW (Byteotian Motor Wagons). Each rental is completely free, but the cars have to be refueled – a car with a tank of capacity k has to be tanked in the very beginning and then refueled after every k traversed roads. BMW is well aware of Byteasar's plan and the fact that he intends to complete his journey as soon as possible. Therefore, they have cunningly chosen the tank capacities of their rentals so that in every town, Byteasar has to (re-)fuel two cars: the one he arrives in and the one he leaves in (with the exception of the initial and final towns, where only one car has to be tanked).

Knowing Byteasar's itinerary, the fuel prices and the tank capacities, determine how much each segment of the route will cost.

Input

In the first line of the standard input, there is a single integer n ($2 \le n \le 50\,000$) that specifies the number of towns in Byteotia. The towns are numbered from 1 to n. In the next line, there is a sequence of n integers $c_1, \ldots, c_n$ ($1 \le c_i \le 10\,000$), separated by single spaces, that specify the fuel prices in all the towns of Byteotia: the number c_i is the cost of filling the tank of any car in the town no. i (this is a "buffet tanking", i.e. "tank all you can for only c_i").

The $n-1$ lines that follow describe the road network of Byteotia. Each of those lines contains two integers a and b ($1 \le a, b \le n$), separated by a single space, which indicate that there is a two-way road connecting the towns no. a and b.

Following those, there is a line with a sequence of n integers $t_1, \ldots, t_n$, separated by single spaces, that specify the order in which Byteasar is going to visit the towns (each of the numbers from 1 to n appears exactly once in this sequence). Finally, the last line of the input contains a sequence of $n - 1$ integers $k_1, \ldots, k_{n-1}$, separated by single spaces, which specify the tank capacities of the rental cars: the number k_i indicates that on the ride from town no. t_i to the town no. t_{i+1}, Byteasar has to refuel his car after every k_i roads traversed. You may assume that k_i always divides the distance between those towns.

Output

Your program should print $n - 1$ lines to the standard output, each containing a single integer. The number in the i-th line should be the total cost of refueling on the way from the town no. t_i to the town no. t_{i+1}.

Examples

standard input	standard output

5
1 2 3 4 5
1 2
2 3
3 4
3 5
4 1 5 2 3
1 3 1 1
	10
	6
	10
	5

Problem I. Seal

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Bytie found a strange document among his mail today. It was a notification that he had inherited a gigantic amount of money after his uncle Byteasar. The document has multiple impression of the Seal of the Kingdom of Byteotia. Just in case though, Bytie would like to make sure that this is not a scam. To this end, he wants to determine if these are impressions of the legitimate seal.

Bytie knows very well how the Seal of the Kingdom of Byteotia looks. However, these is so much ink on the document he received that it is hard to tell if an overzealous clerk made multiple impressions or if it is a feeble attempt to fool Bytie. Help Bytie by writing a program that, given the impressions on the document and the matrix of the Seal of the Kingdom of Byteotia, determines if the impressions on the document are legitimate.

The Seal of the Kingdom of Byteotia has complex security measures that prevent all of the following:

- rotating the seal impression,
- making a seal impression whose part does not appear on the document, and
- inking any point of the document with more than one seal impression.

Input

In the first line of the standard input, there is a single integer q ($1 \le q \le 10$), specifying the number of data sets. The lines that follow describe successive data sets.

In the first line of a single data set description, there are four integers, n, m, a, and b ($1 \le n, m, a, b \le 1000$), separated by single spaces.

The following n lines describe the impressions on the document. Each of these lines contains m characters, each of which is either '`.`' (dot) or '`x`'. The dot signifies that there is no ink at the respective position of the document, whereas '`x`' signifies a trace of ink.

Next, a sample document with a single impression of the Kingdom of Byteotia seal is described, in the same format as the one used for the document received by Bytie, in a lines of b characters, '`.`' or '`x`' each. You may assume that both impressions, from Bytie's document and the legitimate seal matrix, contain a trace of ink.

Output

Your program should print exactly q lines to the standard output. The i-th of these lines should provide the answer for the i-th data set.

If the document received by Bytie could have been imprinted with the legitimate seal, the answer for the data set should be a single word "`TAK`" (Polish for *yes*). If, on the other hand, the document is a forgery, the answer should be the word "`NIE`" (Polish for *no*).

Examples

standard input	standard output
2 3 4 4 2 xx.. .xx. xx.. x. .x x. .. 2 2 2 2 xx xx .x x.	TAK NIE

Problem J. Desert

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

The route from Bythad do Bytara leads among the sands of the Great Byteotian Desert. Making the trip is trying, especially since there are only s wells along the route. Knowing that his country's prosperity depends on communications routes, the ruler of Byteotia has decided to have new wells dug along this particular route. The distance from Bythad to Bytara is $n + 1$ bytemiles, and at each integer multiple of a bytemile from Bythad, there already is a well or a new one can be dug. However, the lower the water level is, the harder and costlier it is to dig a well.

Thus, the rules has commissioned the royal geologist Byteasar with surveying the options. Byteasar has m measurements obtained through the satellite network. Unfortunately, the information gathered by the satellites does not provide direct information about the water levels. Each measurement is for a contiguous fragment of the route and tells only that in certain points of this fragment, the water level is lower than in the remaining points. Moreover, it is known that the water level in every point is between 1 and 10^9 bytemeters below surface.

Help Byteasar by determining what the water level might be in every point along the route. It could turn out that the satellite data are contradictory.

Input

The first line of the standard input contains three integers, n, s, and m ($1 \le s \le n \le 100\,000$, $1 \le m \le 200\,000$), separated by single spaces, which specify the number of potential well slots, number of wells along the route and the number of satellite measurements.

The s lines that follow describe the wells: the i-th one contains two integers, p_i and d_i ($1 \le p_i \le n$, $1 \le d_i \le 1\,000\,000\,000$), which indicate that the i-th well is located p_i bytemiles from Bythad and is d_i bytometers deep (i.e., that the water level in the well is d_i bytometers below ground surface). The wells are given in an increasing order of p_i.

The next m lines describe the satellite measurements: the i-th one contains three integers l_i, r_i, and k_i ($1 \le l_i < r_i \le n$, $1 \le k_i \le r_i - l_i$), followed by a sequence of k_i integers, $x_1, x_2, \ldots, x_{k_i}$ ($l_i \le x_1 < x_2 < \ldots < x_{k_i} \le r_i$). These specify that a measurement was taken on the segment from l_i to r_i (including these endpoints), and it indicated that the water level in each of the points $x_1, \ldots, x_{k_i}$ is *strictly lower* than the water level in each of the remaining (integer) points of the interval, i.e. $\{l_i, \ldots, r_i\} \setminus \{x_1, \ldots, x_{k_i}\}$. The sum of all the k_i's does not exceed 200 000.

Output

If the measurements are contradictory, the first line of the standard output should contain a single word "`NIE`" (Polish for *no*). Otherwise, the first line of the output should contain the word "`TAK`" (Polish for *yes*), whereas the second line should contain a sequence of n integers, each in the range from 1 to 1 000 000 000, specifying the depths below the surface of the water levels at successive points along the route from Bythad. If there are multiple solutions, your program should pick one arbitrarily.

Examples

standard input	standard output

5 2 2
2 7
5 3
1 4 2 2 3
4 5 1 4
	TAK
	6 7 1000000000 6 3

3 2 1
2 3
3 5
1 3 1 2
	NIE

2 1 1
1 1000000000
1 2 1 2
	NIE

Problem K. Direction signs

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

After a long year of intense programming, Byteasar is about to leave for well earned vacation. As he drove to his holiday destination, he saw many *direction signs* along the way, giving the distances (in kilometers) to the numerous towns of Byteland. The distances displayed on the signs are integer numbers, even though the exact distance from such sign to the town in question may not be an integer. Therefore, the distances on the signs are *rounded down* (to the largest integer no larger than the true distance).

After the trip, Byteasar realized that the information on the signs he saw looked suspicious. Pondering this some more, Byteasar concluded that, together, all the distances on the signs are contradictory. He believes that this is due to random people being hired to do the road works, due to the shortage of competent crews. Byteasar would like to find out just how many of the signs surely give wrong distances. To this end, he has decided to find a maximum set of signs such that the distances given on them are consistent. This task is too hard for Byteasar, so he is asking you for help. Fortunately, Byteasar has an excellent memory, and he is able to recall all the signs he saw. He was not, however, looking at the meter as he was passing the signs, so he cannot tell when exactly, or even in what order, he saw them.

We assume that Byteland is a line, and the towns are small enough so that they can be identified with points on said line. We also assume that, during his voyage, Byteasar *has not passed through any town.* A set of signs is consistent if there exists a placement of these signs and all the towns on the line such that the distances displayed on the signs are the true distances rounded down. Naturally, neither the towns nor the signs have to be located at integer points. No two towns and no two signs can be located in the same point. Byteasar swears that there is a set of at least 20% of the signs that are consistent. Since he worked as a road works supervisor a long time ago, in particular on the very road he took on this trip, you can treat his conviction (stemming from an estimation of the fraction of road workers who are competent) as a guarantee about the input data.

Input

In the first line of the standard input, there are two integers, n and m ($1 \leq n \leq 1000$, $1 \leq m \leq 200$), separated by a single space, that specify the number of signs seen by Byteasar and the number of towns in Byteland respectively. Each of the n lines that follow describes a single sign; the i-th such line contains a sequence of m integers, $d_{i,1}, d_{i,2}, \ldots, d_{i,m}$ ($1 \leq d_{i,j} \leq 10^6$), separated by single spaces, such that $d_{i,j}$ is the distance to the town no. j (in kilometers) as displayed on the i-th sign, i.e., rounded down.

Output

In the first line of the standard output, a single integer t should be printed. This should be the maximum number of signs that are consistent. In the second line, there should be t integers that specify the numbers of those signs. They should be given in the order Byteasar could have seen them along the way. If more than one solution exists, your program can choose one of them arbitrarily.

Examples

standard input	standard output

3 2
2 2
2 3
3 2
	2
	2 1