Problem A. Balloons

Input file:	`balloons.in`
Output file:	`balloons.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

A group of children came to a toy store. Each of them would like to buy a number of balloons. The children like diversity — none of them wants to have two balloons of the same colour. Help the shop-assistant to check whether orders of all children can be completed within the current assortment of the store.

Task

Write a program that:

- reads a description of the store's assortment and the orders made by children from the input file,
- checks whether all children can be made happy,
- writes the result to the output file.

Input

The first line of input contains two integers n and m ($1 \le n \le 200\,000$, $2 \le m \le 200\,000$), separated by a single space and denoting the number of different colours of balloons that are present in the store and the number of children. The second line of input contains n integers a_i ($1 \le a_i \le 1\,000\,000$ for $1 \le i \le n$), separated by single spaces and denoting the quantities of balloons of respective colours. The third line of input contains m integers b_i ($1 \le b_i \le 1\,000\,000$ for $1 \le i \le m$), separated by single spaces and denoting the orders of respective children; $b_i = k$ means that the i-th child would like to buy k balloons, all having different colours.

Output

The first and only line of output should contain a single word `YES`, if orders of all children can be completed, or `NO` otherwise.

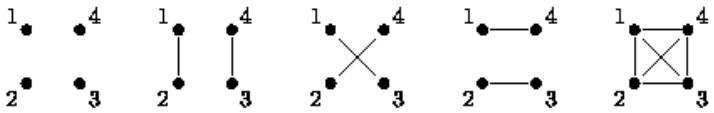
Example

balloons.in	balloons.out
4 3 3 2 1 3 1 3 4	YES
4 3 3 2 1 3 1 4 4	NO

Problem B. Cliquers

Input file:	`cliquers.in`
Output file:	`cliquers.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

An undirected graph with n vertices is called **a symmetric labeled cliquer** if each connected component of the graph contains the same number of vertices and is a clique, and the vertices of the graph are numbered with numbers from the set $\{1, \ldots, n\}$. Maurycy has drawn all symmetric labeled cliquers on a piece of paper and is going to assess beauty of each of them with a number from the set $\{1, \ldots, m\}$ (in particular, different cliquers may be assigned equal grades). In how many ways can he do this? The result should be computed modulo $10^9 - 401$. The figure below depicts all symmetric labeled cliquers for $n = 4$.



Input

The only line of the input file contains two integers n and m ($1 \le n, m \le 2 \cdot 10^9$), separated by a single space and denoting the number of vertices of each symmetric labeled cliquer and the number of grades respectively.

Output

The only line of the output file should contain the number of possible sets of grades modulo $10^9 - 401$.

Example

cliquers.in	cliquers.out
4 2	32

Problem C. Computation of a Road Network Plan

Input file:	`computation.in`
Output file:	`computation.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Byteman is going for a car trip around Byteland, but he is unfortunately unable to buy a map of the country. From his friends he learned about some properties of the bytean road network:

- There are n cities in Byteland, numbered from 1 to n.
- Each road is bidirectional and connects two different cities.
- Each pair of different cities is connected by exactly one *path*, consisting of one ore more roads, on which no city appears more than once.
- The longest path, on which no city appears more than once, consists of d roads.

Using information that he managed to collect, Byteman is going to try to reconstruct the road map of Byteland. The total number of theoretically possible road network plans satisfying the given conditions may be quite big, so any correct plan will be enough for Byteman.

Task

Write a program that:

- reads numbers n and d from the input file,
- computes any road network plan that is consistent with information collected by Byteman, or checks that no such plan exists,
- writes the result to the output file.

Input

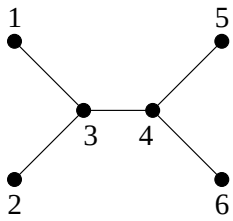
The first and only line of input contains two integers n and d ($2 \le n \le 200$, $0 \le d < n$), separated with a single space.

Output

If no road network plan exists for the values of parameters n and d from the input, the first and only line of output should contain a single word `NONE`. In the opposite case the output should consist of $n-1$ lines. Each line should contain a description of one bidirectional road — two different integers in the range from 1 to n, separated with a single space and denoting the numbers of cities connected by this road. The order of roads and numbers of cities connected by roads in the output can be arbitrary.

Example

computation.in	computation.out
6 3	1 3 2 3 3 4 4 5 4 6


1
5
3
4
2
6

Problem D. Electricity

Input file:	`electricity.in`
Output file:	`electricity.out`
Time limit:	7 seconds
Memory limit:	256 megabytes

Citizens of the Bytelon village decided to take advantage of the natural sources of energy for powering their houses, that is why they decided to build many windmills.

All Bytelon households are located in a straight line, on the same side of the road (there is only one road in Bytelon), so all of the windmills were built in a straight line as well, but on the other side of the road. Some windmills were connected with some households with high-voltage lines that run over the road.

One windmill could have been connected with many households, one household — with many windmills. Similarly, some windmills or households could have been not connected at all. Each connection runs along a straight line. For any pair of windmill and household there is no more than one high-voltage line.

Bytelon government does not like this solution, however. It is believed that much more efficient is the solution where each windmill powers at most one household and each household is powered by at most one windmill. What is more, high-voltage lines should not cross the same ground point. In another words, when looking at high-voltage lines from the bird's eye view (all high-voltage lines are seen as straight lines) they should not cross. This solution is enforced by strong winds, which could have otherwise push high-voltage lines against each other causing short-circuit.

Government of Bytelon asked you to write a program, which counts the number of different ways for selecting some of the high-voltage lines, so that the set of that lines fulfill specified requirements.

Government does not like to play with big numbers; it is sufficient for your program to calculate just a remainder of the division of the computed number of possible solutions by the number specified by the Government.

Task

Write a program which:

- reads description of already built high-voltage lines from the input file,
- determines remainder of the division of the number of possible ways of selecting lines by the specified number,
- writes output to the output file.

Input

The first line of the standard input contains four integers n, m, k and r ($1 \le n, m \le 200\,000$, $1 \le k \le 1\,000\,000$, $2 \le r \le 10^9$), separated with single spaces and representing adequately: the number of households, the number of windmills, the number of high-voltage power lines and the number given by the Government. Households are numbered with integers in the range from 1 to n, windmills — with numbers from 1 to m. Households are located along the road in the order starting with a household number one to the household with number n. On the other side of the road there are windmills in order from the first to m-th.

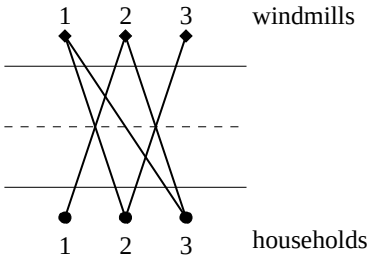
i-th of the following k lines contains two integers h_i and w_i ($1 \le h_i \le n$, $1 \le w_i \le m$ for $1 \le i \le k$), separated with a single space and representing i-th high-voltage power line connecting h_i-th household with w_i-th windmill.

Output

The first and only line should contain one integer representing remainder of division of the number of correct ways to connect households with windmills by r.

Example

electricity.in	electricity.out
3 3 5 100 1 2 2 3 3 1 2 1 3 2	8



Explanation of the example: There is only one solution with no high-voltage lines (seems like such a solution does not make sense in practice, but it is consistent with the law...), 5 possible ways to select only one line, 2 ways to select two lines, no way to select three lines.

Problem E. Enumeration of Road Network Plans

Input file:	`enumeration.in`
Output file:	`enumeration.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

Byteman is going for a car trip around Byteland, but he is unfortunately unable to buy a map of the country. His friends told him about some properties of the bytean road network:

- There are n cities in Byteland, numbered from 1 to n.
- Each road is bidirectional and connects two different cities.
- Each pair of different cities is connected by exactly one *path*, consisting of one or more roads, on which no city appears more than once.
- The longest path, on which no city appears more than once, consists of d roads.

Using information that he managed to collect, Byteman is going to try to reconstruct the road map of Byteland. Byteman would not like to work on it too much, so he would like to know the number of different road network plans satisfying the given conditions. During comparison of plans Byteman does not take exact locations of cities into consideration, but only the plan of connections; moreover, he is not interested in particular city numbers. In other words, Byteman considers two plans the same if and only if there exists a one-to-one mapping from cities of one plan to the cities of the other one, such that if cities u and v are connected by a road on the first plan then their equivalents u' and v' are also connected by a road on the second plan.

Task

Write a program that:

- reads numbers n, d and p from the input file,
- computes the number of different road network plans, that are consistent with information that Byteman managed to collect, modulo p,
- writes the result to the output file.

Input

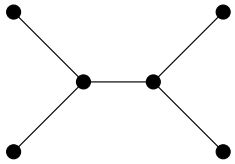
The first and only line of the input contains three integers n, d and p ($1 \le n \le 200$, $0 \le d < n$, $n < p \le 10^9$, p is a prime number), separated with single spaces.

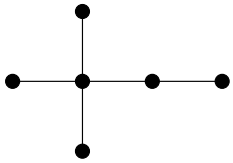
Output

The first and only line of output should contain a single integer — the remainder of the division by p of the number of different plans that are consistent with conditions known to Byteman.

Example

enumeration.in	enumeration.out
6 3 13	2





Problem F. Idempotent Functions

Input file:	`idempotent.in`
Output file:	`idempotent.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

You are given a positive integer n. By A we mean a set $\{1, 2, 3, \ldots, n\}$. Function $f : A \to A$ is called a *permutation* if it is injective (for distinct arguments returns distinct values). Function $f : A \to A$ is called *idempotent* if for every $i \in A$ holds $f(f(i)) = f(i)$.

You are given a function $f : A \to A$. How many pairs of functions (g, h) satisfy following conditions:

- $g : A \to A$ is a permutation,
- $h : A \to A$ is idempotent,
- for every $i \in A$ holds $f(i) = h(g(i))$?

Task

Write a program which:

- reads from the input file the number n and description of function f,
- determines the number of pairs of functions (g, h) satisfying the requirement described above,
- writes result to the output file.

Input

In the first line of the input there is one integer n ($1 \le n \le 200\,000$). Second line contains description of function f: $f(i)$ ($1 \le f(i) \le n$) for $i = 1, \ldots, n$, separated with single spaces.

Output

In the first line of the output there should be a single integer: the remainder of the division by $1\,000\,000\,007$ of the number of different pairs of functions (g, h) satisfying the requirement.

Example

idempotent.in	idempotent.out
8 7 4 5 1 7 4 4 1	288

Problem G. Journey

Input file:	journey.in
Output file:	journey.out
Time limit:	7 seconds
Memory limit:	256 megabytes

In Byteland there are n cities numbered from 1 to n. These cities are connected by a network of m bidirectional roads. It is known that each pair of cities is connected by at most one road.

Byteman admitted recently that he likes some cities more than others. More precisely, he spends his time especially well in cities with numbers from 1 to k, so during every journey he visits each of them at least once.

Byteman's journey is a sequence of n cities, such that each pair of consecutive cities is connected by a road. The journey can start and end in any city. Your task is to compute the number of distinct journeys around Byteland that Byteman can make. Because this number might be quite large, it will be enough to find it modulo $10^9 + 9$.

Task

Write a program that:

- reads a description of the network of connections in Byteland from the input file,
- computes remainder of the division of the number of distinct journeys that can be made by $10^9 + 9$,
- writes the result to the output file.

Input

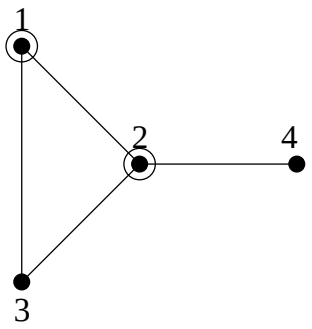
The first line of input contains four integers n, m, k and d ($1 \le n \le 20$, $1 \le k \le \min(n, 7)$, $1 \le d \le 1\,000\,000\,000$), separated by single spaces. The following m lines contain descriptions of connections between cities of Byteland. A description of a road consists of two numbers a_i, b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$), separated by a single space and denoting the numbers of cities connected by the i-th road.

Output

The output should contain one integer, denoting the number of distinct journeys, that Byteman can make, modulo $10^9 + 9$.

Example

journey.in	journey.out
4 4 2 3 1 2 2 3 3 1 2 4	10



All possible Byteman's journeys are:

- $1 \to 2 \to 1$
- $1 \to 2 \to 3$
- $1 \to 2 \to 4$
- $1 \to 3 \to 2$
- $2 \to 1 \to 2$
- $2 \to 1 \to 3$
- $2 \to 3 \to 1$
- $3 \to 1 \to 2$
- $3 \to 2 \to 1$
- $4 \to 2 \to 1$

Problem H. Mosaicism

Input file:	`mosaicism.in`
Output file:	`mosaicism.out`
Time limit:	7 seconds
Memory limit:	256 megabytes

Molecular biologists have been studying genomes (i.e. sequences of genes) of particular organisms, trying to infer some conclusions about evolution of species and operation of cells and tissues. In their research they compare the structure and functions of respective genes and they identify genes very similar to each other (called **homologous genes**).

Recently, while examining viruses attacking bacteria (called **bacteriophages** or simply **phages**), they observed a very intriguing phenomenon. If sequences of genes of some phages are arranged one below another and homologous genes are coloured with the same colour, then a peculiar mosaic is obtained:

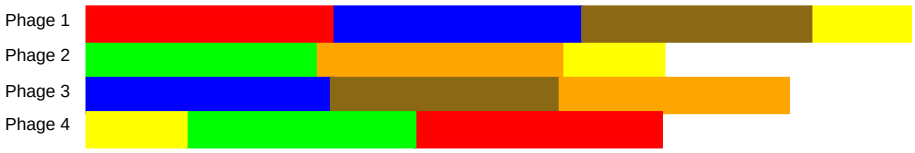


Figure 1: *"The Four Phages"*. Colours of genes of phage 1: red, blue, brown, yellow; phage 2: green, orange, yellow; phage 3: blue, brown, orange; phage 4: yellow, green, red.

To describe the phenomenon that they discovered, scientists invented a measure that they called **the coefficient of mosaicism**. This coefficient can be computed for a phage only in juxtaposition with some other phages — it is then equal to the total number of points computed in the following way. For every two genes A, B from the i-th phage and for every two different phages j, k such that:

- gene A has some homologous gene in the j-th phage and does not have homologous genes in k-th phage,
- gene B has some homologous gene in the k-th phage and does not have homologous genes in the j-th phage,

the i-th phage ($i \neq j$ and $i \neq k$) receives 1 point. Each quadruple A, B, j, k is counted exactly once in the summation, i.e. quadruples A, B, j, k and B, A, k, j are considered the same.

In the situation illustrated in the above figure, phage number 3 has the coefficient of mosaicism equal to 2. That is because the blue gene has a homologous gene in phage 1 and does not have any in phage 2; on the other hand, the orange gene has a homologous gene in phage 2 and does not have any in phage 1. A similar situation holds for genes: brown and orange. Phage number 1 has the coefficient of mosaicism equal to 6 — it receives points for pairs of genes: red–blue, red–brown, 2 times yellow–blue and 2 times yellow–brown.

Manual computation of the coefficients is quite difficult, so the biologists decided to ask you to write a program that would compute the coefficients of mosaicism of all given phages.

Task

Write a program that:

- reads the descriptions of pairs of homologous genes in a set of phages from the input file,
- computes the coefficients of mosaicism for all phages,

- writes the result to the output file.

Input

The first line of input contains one integer n ($3 \leq n \leq 300$) denoting the number of phages under consideration.

The i-th of the following n lines contains a description of the sequence of genes of the i-th phage. Each such description starts with a single integer l_i ($1 \leq l_i \leq 300$) denoting the number of genes in the i-th phage's sequence, which is followed by l_i integers a_{ij} ($1 \leq a_{ij} \leq 100\,000$) describing its successive genes. All these numbers are separated with single spaces.

Two genes are considered homologous if and only if they are represented by equal numbers in the description from the input. The scientists proved that no two genes of a phage can be homologous[1], so no number can appear an a description of a single phage twice.

Output

The output should consist of n lines. i-th of them should contain a single integer equal to the coefficient of mosaicism of the i-th phage.

Example

mosaicism.in	mosaicism.out
4 4 1 5 2 4 3 3 6 4 3 5 2 6 3 4 3 1	6 3 2 1

The figure from the task description illustrates the example test case.

[1] In reality, appearance of two homologous genes in a single phage is quite improbable.

Problem I. Reconstruction of Byteland

Input file:	`reconstruction.in`
Output file:	`reconstruction.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Everyone get to work! Byteland needs to be reconstructed after a devastating war!

You are responsible for assigning postal codes to bytean cities. Each city should receive one postal code — a positive integer not greater than 10^9. Different cities should be assigned different postal codes.

The Bytean Postal Service is organized in a quite strange way; a letter can be sent from city A to city B only if the postal codes of these two cities have a common divisor greater than 1. Obviously, one of your objectives is that there should be a possibility of sending letters directly from every city to every other city after the postal codes are assigned.

Additionally, the new anti-corruption law requires that for each set of cities, containing more than a half of bytean cities, postal codes assigned to cities from this set must not have any common divisor greater than one.

Task

Write a program that:

- reads the number of cities of Byteland from the input file,
- assigns a postal code to each bytean city,
- writes the assignment to the output file.

Input

The first and only line of input contains one integer n ($4 \le n \le 100$) denoting the number of cities of Byteland.

Output

The output should consist of exactly n lines. The i-th line should contain one positive integer not greater than 10^9 — the postal code of the i-th bytean city. You can assume that for each possible input there exists a valid assignment of postal codes to the cities. If there are multiple correct solutions, your program should output any one of them.

Example

reconstruction.in	reconstruction.out
5	714 2090 4485 29029 215441

Problem J. Safe

Input file:	`safe.in`
Output file:	`safe.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

Byteman keeps all of his savings in an old safe. The lock of this safe consists of n identical wheels, each of them has the same word consisting of m letters written on it. Wheels can be rotated independently of each other (this way each wheel can be put in m different positions). Safe gets opened, when for all m positions letters written on all wheels for that position are the same.

Recently one of the Byteman's acquaintance told him, that it is a good idea to keep money in a bank. Byteman decided to open his safe as soon as possible and move all of this money to the high-interest bank account. Assuming that rotation of any wheel by $\frac{360}{m}$ degrees to the left or to the right can be performed within one second, calculate, how long would it take (at minimum) to open Byteman's safe.

Task

Help Byteman!

Input

The first line of the input file contains two integers n and m ($2 \le n, m \le 1\,000\,000$), separated with a single space and denoting the number of wheels in the lock and the length of the word written on each wheel. The second line of the input contains one word $s_1 s_2 \dots s_m$ of length m, consisting of large English letters. In the third line there are n integers $o_1, o_2, \dots, o_n$ ($0 \le o_i < m$), separated with single spaces. $o_i = k$ means that the word from the i-th wheel is rotated by k positions to the left (relatively to some defined initial position), which means it is set in position $s_{k+1} s_{k+2} \dots s_m s_1 s_2 \dots s_k$. For instance, if $o_i = 0$ then the word on i-th wheel is not rotated at all.

Output

The only line of the output file should contain one integer, the minimal time required to open Byteman's safe.

Example

safe.in	safe.out
4 6 SLOWIK 2 0 3 5	6

For the example lock, words written on each wheel look as follows:

```
OWIKSL
SLOWIK
WIKSLO
KSLOWI
```

For example, rotating the first wheel by one position to the left gives `WIKSLO`. When we rotate the same wheel to the right instead, we get `LOWIKS`.

Problem K. Studies

Input file:	`studies.in`
Output file:	`studies.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

Byteland University (BU) offers Computer Science studies consisting of n levels. Each level is an equivalence of semester, however the number of levels can be less or greater than the number of semesters at the classical CS studies program. Every student during his studies can submit different applications to the dean of the faculty. Those applications have a great impact on the student's career. In particular, applications can result in the change of level of the studies. Submitting an application can have financial impact on the student's budget — as well profitable (scholarships) as negative (charges for attending additional classes etc.).

Byteman is a lazy, but very smart student of BU. For the past years he has been collecting data about different applications that has been submitted by other students of BU. For each application, Byteman knows how the application influenced the level of studies as well as the budget of the student. Byteman doesn't care much about fast graduation. The only thing he cares about is maximizing his income.

A guarantee of infinite income is the following situation: student can start his studies from some level and then submit a sequence of applications in such a way that at the end he **returns to the initial level of studies**, in the same time earning some positive amount of bytedollars. Byteman is a very patient person — he can submit many applications. Gush... He can even submit the same application many times, as long as this action leads to some positive income. Byteman assumes, that the dean behaves deterministically, which means that the answer to the same application is always the same.

Task

Write a program which:

- reads a description of all possible applications from the input file,
- determines all levels of studies at the BU, which can lead to infinite income,
- writes result to the output file.

Input

The first line of the input contains two integers n and m ($2 \le n \le 300$, $1 \le m \le n \cdot (n-1)$), separated with a single space and representing the number of levels at the BU and the number of applications to analyze. Following m lines contain description of applications. Each line contains three integers a_i, b_i and c_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$, $-10^9 \le c_i \le 10^9$), separated with single spaces and meaning: the level of studies student was at, when he submitted an application, the level of studies student got transferred to after his application had been considered by the dean and costs associated with this application (positive value represents profit of the student, negative — loss). None of the pairs (a_i, b_i) occurs more than once, but both (a_i, b_i) and (b_i, a_i) can occur in the input.

Output

The first line of the output should contain one integer k, representing the number of different levels of studies, from which Byteman can start his studies to obtain infinite income. The second line should contain k integers in the range $\{1, \ldots, n\}$, separated with single spaces and representing all levels of studies Byteman is interested in. Numbers should be listed in increasing order.

Example

studies.in	studies.out
8 8 1 2 3 1 3 -3 5 4 4 6 5 -1 6 7 -1 7 8 5 8 6 2 4 8 -3	5 4 5 6 7 8