Problem A. Beads

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	256 mebibytes

Byteasar once decided to start manufacturing necklaces. He subsequently bought a very long string of colourful coral beads for a bargain price. Byteasar now also has a machine that, for a given k ($k > 0$), can cut the string into pieces (or substrings) of k coral beads (i.e., the first piece consists of the beads no. $1, \ldots, k$, the second of $k+1, \ldots, 2k$, etc.). If the length of the string (measured in coral beads) is not a multiple of k, then the last piece is not used, as it has length smaller than k. From now on we denote the colours of the beads with positive integers.

Byteasar, always praising diversity, wonders how he should choose the number k in order to get as many different substrings as possible. The ends of the long string that will be cut are different: there are specific beginning and ending (rather than two interchangeable endpoints), and the machine of course starts cutting at the beginning. On the other hand, in the substrings obtained from cutting the endpoints are interchangeable, and so the substrings can be reversed. In other words, the substrings $(1, 2, 3)$ and $(3, 2, 1)$ are identical to us. Write a program that determines the optimum value of k for Byteasar.

For example, for the following string of beads:

$$(1, 1, 1, 2, 2, 2, 3, 3, 3, 1, 2, 3, 3, 1, 2, 2, 1, 3, 3, 2, 1),$$

- using $k = 1$, we would get 3 different substrings: (1), (2), (3),
- using $k = 2$, we would get 6 different substrings: $(1, 1)$, $(1, 2)$, $(2, 2)$, $(3, 3)$, $(3, 1)$, $(2, 3)$,
- using $k = 3$, we would get 5 different substrings: $(1, 1, 1)$, $(2, 2, 2)$, $(3, 3, 3)$, $(1, 2, 3)$, $(3, 1, 2)$,
- using $k = 4$, we would get 5 different substrings: $(1, 1, 1, 2)$, $(2, 2, 3, 3)$, $(3, 1, 2, 3)$, $(3, 1, 2, 2)$, $(1, 3, 3, 2)$,
- using larger values of k would give at most 3 different substrings.

Input

In the first line of the standard input there is an integer n ($1 \leq n \leq 200\,000$) denoting the length of the string to cut. In the second line there are n positive integers a_i ($1 \leq a_i \leq n$), separated by single spaces, that denote the colours of successive beads in Byteasar's string.

Output

Two integers, separated by a single space, should be printed out to the first line of the standard ouput: the (maximum) number of different substrings that can be obtained with an optimal choice of parameter k, and the number l of such optimal values of k. The second line should contain l integers separated by single spaces: the values of parameter k that yield an optimum solution; these can be given in arbitrary order.

Examples

standard input	standard output
21 1 1 1 2 2 2 3 3 3 1 2 3 3 1 2 2 1 3 3 2 1	6 1 2

Problem B. Bridges

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

San Bytecisco is a beautifully situated coastal town. It consists of n small, yet densely populated islands, numbered from 1 to n. Certain islands are connected with bridges, used for (bidirectional) road traffic. Each pair of islands can be connected with at most one bridge. The islands are connected in such a way that every island can be reached from every other by using the bridges only.

Byteasar and Bytie are going for a bike trip in San Bytecisco. The will start their ride at the island no. 1. They intend to visit every island, while passing along every bridge once and ending the trip where it began, i.e., the island no. 1. Being quite seasoned riders, they expect some serious trouble from... the wind! After all, it is very windy along the coast, and especially so on the bridges between the islands. Obviously, depending on its speed and direction, the wind makes it hard to cross the bridge *in different extent for either direction*. For simplicity we will assume for every bridge and direction of crossing, the opposing wind speed is constant.

Help Byteasar and Bytie to find a route as they desire that will in addition be the least tiresome. Byteasar and Bytie agreed on the maximum opposing wind speed as a measure of a route's tiresomeness.

Input

In the first line of the standard input there are two integers separated by a single space: n and m ($2 \le n \le 1\,000$, $1 \le m \le 2\,000$), denoting the number of islands and the number of bridges in San Bytecisco respectively. The islands are numbered from 1 to n, while the bridges from 1 to m. The following m lines specify the bridges. The line no. $(i+1)$ contains four integers a_i, b_i, l_i, p_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$, $1 \le l_i, p_i \le 1\,000$), separated by single spaces. These denote that the bridge no. i connects the islands no. a_i and b_i. The opposing wind speeds are l_i when one goes moves from a_i to b_i, and p_i if one goes from b_i to a_i.

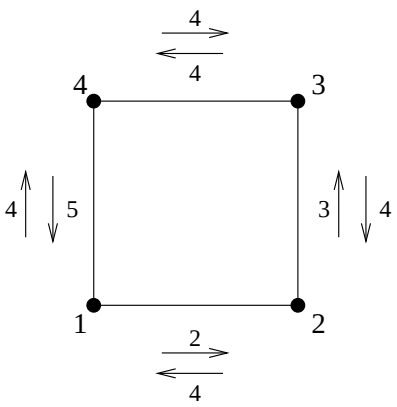
Output

If there is no route satisfying the requirements of the daring two riders, the first and only line of the standard output should hold the word `NIE` (*no* in Polish). Otherwise, the output should have two lines, specifying the least tiresome route over San Bytecisco. The first line should hold the maximum opposing wind speed for that route, i.e., the number we wish to minimize. The second line should hold m integers, separated by single spaces, giving the numbers of successive bridges one crosses on the least tiresome route.

Should there be more than one least tiresome route, your program can pick one arbitrarily.

Examples

standard input	standard output
4 4 1 2 2 4 2 3 3 4 3 4 4 4 4 1 5 4	4 4 3 2 1



Explanation of the example: The optimum route for Byteasar and Bytie is the following one: $1 \xrightarrow{4} 4 \xrightarrow{4} 3 \xrightarrow{4} 2 \xrightarrow{4} 1$. The maximum opposing wind speed on that route is 4.

Problem C. Frog

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

On the bed of one particularly long and straight Byteotian brook there lie n rocks jutting above the water level. Their distances from the brook's spring are $p_1 < p_2 < \ldots < p_n$ respectively. A small frog sitting on one of these is about to begin its leaping training. Each time the frog leaps to the rock that is the k-th closest to the one it is sitting on. Specifically, if the frog is sitting on the rock at position p_i, then it will leap onto such p_j that:

$$|\{p_a \ : \ |p_a - p_i| < |p_j - p_i|\}| \leq k \qquad \text{and} \qquad |\{p_a \ : \ |p_a - p_i| \leq |p_j - p_i|\}| > k.$$

If p_j is not unique, then the frog chooses among them the rock that is closest to the spring. On which rock the frog will be sitting after m leaps depending on the rock is started from?

Input

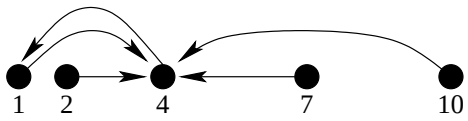
The first line of the standard input holds three integers, n, k and m ($1 \leq k < n \leq 1\,000\,000$, $1 \leq m \leq 10^{18}$), separated by single spaces, that denote respectively: the number of rocks, the parameter k, and the number of intended leaps. The second line holds n integers p_j ($1 \leq p_1 < p_2 < \ldots < p_n \leq 10^{18}$), separated by single spaces, that denote the positions of successive rocks on the bed of the brook.

Output

Your program should print a single line on the standard output, with n integers $r_1, r_2, \ldots, r_n$ from the interval $[1, n]$ in it, separated by single spaces. The number r_i denotes the number of the rock that the frog ends on after making m leaps starting from the rock no. i (in the input order).

Examples

standard input	standard output
5 2 4 1 2 4 7 10	1 1 3 1 1



The figure presents where the frog leaps to (in a single leap) from each and every rock.

Problem D. Godzilla

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

The favourite activity of citizens of Byteburg is watching a popular soap opera *The Cold and the Pitiful*. This TV series is transmitted via cable network to each house in Byteburg. The TV cable network consists of n nodes and m **unidirectional** connections. For every node in the network, there is at least one house connected to it. The TV series is transmitted to some of the nodes (called the *transmission* nodes). You can watch the TV program in a given house if there exists a connection (not necessarily direct) from a transmission node to the node to which the house is connected. To minimize the costs, the number of transmission nodes should be as small as possible.

Unfortunately, one day a nasty monster Godzilla has come to Byteburg. Strangely enough, the monster enjoys eating TV cable infrastructure. Each day it eats one connection of the network. The owner of the TV network does not want the number of abonents to decrease, so he has to update the set of transmission nodes in such a way that every citizen can still watch *The Cold and the Pitiful* every day. Help him check if he does it optimally.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 100\,000$), which denote the number of nodes and the number of connections in the cable network. The following m lines contain descriptions of the connections. Each of the lines contains two integers a and b ($1 \le a, b \le n, a \ne b$), which denote a unidirectional connection from the node a to the node b. There exists at most one direct connection between any pair of nodes in a given direction. The next line contains an integer k ($1 \le k \le m$) representing the number of connections that were attacked by Godzilla. The following k lines contain the numbers of the eaten connections (the connections are numbered from 1 in the same order as they are listed in the input). Every connection appears in the list at most once.

Output

Your program should output exactly k lines. The i-th of these lines should contain a single integer, denoting the number of transmission nodes to which the TV series should be transmitted after the i-th Godzilla's attack.

Examples

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

Problem E. Intelligence test

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

One of the tasks in the Byteotian Intelligence Test (BIT) is to cross out numbers from an initial sequence in such a way that leaves as a result certain given sequences. Byteasar longs to become the IQ Master of Byteotia, but he is no good in this kind of tasks. But since practice makes perfect, he intends to practise a lot. So much in fact that he asks you to write a program that will facilitate the training by verifying his answers quickly.

Input

The first line of the standard input contains one integer m ($1 \leq m \leq 1\,000\,000$). The second line holds m integers $a_1, a_2, \ldots, a_m$ ($1 \leq a_i \leq 1\,000\,000$ for $1 \leq i \leq m$), separated by single spaces, that constitute the initial sequence of the test. The third line of the input holds one integer n. The following $2n$ lines describe the sequences to be obtained by crossing out numbers from the initial sequence. Each sequence's description takes two successive lines. The first of these two lines contains an integer m_i ($1 \leq m_i \leq 1\,000\,000$). The second contains an m_i-element long sequence of integers $b_{i,1}, b_{i,2}, \ldots, b_{i,m_i}$ ($1 \leq b_{i,j} \leq 1\,000\,000$ for $1 \leq j \leq m_i$) separated by single spaces. You may assume that the total length on given n sequences does not exceed 1 000 000.

Output

Your program should print out n lines to the standard output. The i-th line (for $1 \leq i \leq n$) should hold one word, "`TAK`" (*yes* in Polish) if the i-th input sequence can be obtained by crossing out (i.e., removing) some, not necessarily contiguous, numbers from the initial sequence, or "`NIE`" (*no* in Polish) otherwise. Mind you, only the words should be printed, no quotation marks. Of course, the order of the numbers left after crossing out is important, as can be seen in the example.

Examples

standard input	standard output
7 1 5 4 5 7 8 6 4 5 1 5 5 8 6 3 2 2 2 3 5 7 8 4 1 5 7 4	TAK NIE TAK NIE

Problem F. Lamp

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

In the middle of the night Bitratio turned on the lamp at the entrance to the building that Byteasar lives in. Now the strong light prevents Byteasar from sleeping. While the lamp does not shine directly on Byteasar's windows, it does so by reflecting in other windows. Deprived of sleep, Byteasar is becoming irritated. To remedy that he tries to occupy his mind, but all he can think of is the light. Thus Byteasar looked out the window and wondered if his neighbours suffer similar torture, i.e., whether the light shines on their windows as well. Now that is an interesting question, at least in Byteasar's opinion. You learn of the puzzle sooner than you would wish: unable to solve the problem all by himself, thinking little of sleep now (be it his and yours), Byteasar calls you to ask for help. You know him well enough to understand that you too will not get any sleep until you write a program that solves his problem.

Byteasar lives in the building B, which has n windows. The lamp is situated on a wall at the very bottom of this building. Opposite the building B, exactly 10 meters apart, there is another building, C. The wall of this m-windowed building is parallel to the wall of B, the Byteasar's building.

The lamp light behaves like you would expect, i.e., in the way predicted by geometrical optics (or ray optics). Namely the light propagates along rays, and if *a ray hits a window*, it is reflected. Due to The Law of Reflection, the angle of the ray's reflection equals the angle of incidence.

We introduce coordinate systems on the the walls of the two buildings in the following way. Both X axes are horizontal, while both Y axes are vertical; the axes on both walls are identically oriented, and the $(0, 0)$ points of the walls are opposite one another. The windows (on either building) are simply rectangles with sides parallel to the axes of the coordinate system. A ray is reflected only in the interior of any window; it is absorbed on the window's boundary. In each building, no two windows share any part of their interiors. The lamp is located on the wall of the B building at the point $(0, 0)$, which is neither inside nor at the boundary of any window.

Input

In the first line of the standard input there are two integers n and m ($1 \le n, m \le 600$), separated by a single space, denoting the number of windows in the first and second building respectively. The n lines that follow describe the windows in Byteasar's building (the B building), one per line.

The line no. $(i + 1)$ (for $1 \le i \le n$) holds four integers $x_{1,i}$, $y_{1,i}$, $x_{2,i}$, $y_{2,i}$ ($-1\,000 \le x_{1,i} < x_{2,i} \le 1\,000$, $0 \le y_{1,i} < y_{2,i} \le 1\,000$), separated by single spaces. Such quadruple means that the i-th window of the B building is a rectangle whose lower-left and upper-right corners' coordinates in meters are $(x_{1,i}, y_{1,i})$ and $(x_{2,i}, y_{2,i})$ respectively.

The following m lines specify the windows of the C building is a similar way.

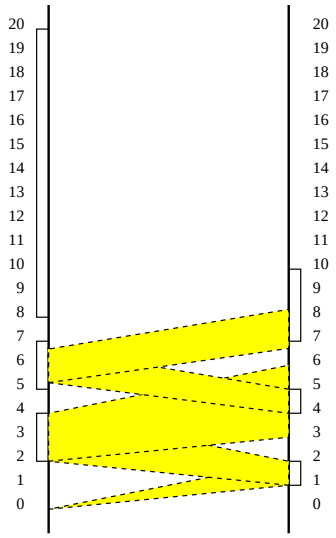
Output

In the first line of the standard output your program should print the number of windows in the B building whose interiors are hit by some ray. You may assume that in every test instance there will be at least one such window (the Byteasar's window).

In the second line the numbers of these windows (windows are numbered starting from 1) should be printed in increasing order, separated by single spaces.

Examples

standard input	standard output
3 3 -1 2 1 4 -1 5 1 7 -3 8 -2 20 -1 1 1 2 -1 4 1 5 -1 7 1 10	2 1 2



Example explanation: A ray hits the first window of the B building after being reflected in the first window of the C building (for instance, after being reflected at the point $(0, 1.5)$ of C it hits the point $(0, 3)$ of B). To hit the second window of B, the ray has to be reflected three times — the very same ray hits the point $(0, 4.5)$ inside the second window of C, and then $(0, 6)$ in the second window of B. No ray hits the third window of B. Let us mention that every point on the wall of the C building is hit by some ray, but in the figure we depicted only those rays that hit some windows of B after reflection.

Problem G. Leonardo's Numbers

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

We define the sequence of Leonardo's numbers (L_i) as follows:

$$L_0 = L_1 = 1,$$

$$L_{i+1} = L_i + L_{i-1} + 1 \quad \text{for } i \geq 1.$$

Today, 800 years after Leonardo Fibonacci's death, we would like to find the value of the expression:

$$L_0^k + L_1^k + L_2^k + \cdots + L_n^k$$

for given values of parameters n and k.

Input

The first and only line of input contains two positive integers n and k ($k \leq 13$, n fits in a 64-bit unsigned integer type).

Output

The only line of output should contain the last 9 digits of the decimal representation of the sum specified in the problem statement.

Examples

standard input	standard output
3 2	000000036

The requested sum equals $1^2 + 1^2 + 3^2 + 5^2 = 36$.

Problem H. Monotonicity

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

For an integer sequence $a_1, a_2, \ldots, a_n$ we define its *monotonicity scheme* as the sequence $s_1, s_2, \ldots, s_{n-1}$ of symbols $<$, $>$ or $=$. The symbol s_i represents the relation between a_i and a_{i+1}. For example, the monotonicity scheme of the sequence $2, 4, 3, 3, 5, 3$ is $<, >, =, <, >$.

We say that an integer sequence $b_1, b_2, \ldots, b_{n+1}$ with monotonicity scheme $s_1, s_2, \ldots, s_n$, *realizes* another monotonicity scheme $s'_1, s'_2, \ldots, s'_k$ if for every $i = 1, 2, \ldots, n$ it holds that $s_i = s'_{((i-1) \bmod k)+1}$. In other words, the sequence $s_1, s_2, \ldots, s_n$ can be obtained by repeating the sequence $s'_1, s'_2, \ldots, s'_k$ and removing appropriate suffix from that repetition. For example, the sequence $2, 4, 3, 3, 5, 3$ realizes each and every one of the following schemes:

- $<, >, =$
- $<, >, =, <, >$
- $<, >, =, <, >, <, <, =$
- $<, >, =, <, >, =, >, >$

as well as many others.

An integer sequence $a_1, a_2, \ldots, a_n$ and a monotonicity scheme $s_1, s_2, \ldots, s_k$ are given. Your task is to find the longest subsequence $a_{i_1}, a_{i_2}, \ldots, a_{i_m}$ $(1 \le i_1 < i_2 < \ldots < i_m \le n)$ of the former that realizes the latter.

Input

The first line of the standard input holds two integers n and k $(1 \le n \le 500\,000,\ 1 \le k \le 500\,000)$, separated by a single space, denoting the lengths of the sequences (a_i) and monotonicity scheme (s_j) respectively.

The second input line gives the sequence (a_i), i.e, it holds n integers a_i separated by single spaces $(1 \le a_i \le 1\,000\,000)$.

Finally, the third lines gives the monotonicity scheme (s_j), i.e., it holds k symbols s_j of the form `<`, `>` or `=` separated by single spaces.

Output

In the first line of the standard output your program should print out a single integer m, the maximum length of a subsequence of $a_1, a_2, \ldots, a_n$ that realizes the scheme $s_1, s_2, \ldots, s_k$.

In the second line it should print out any such subsequence $a_{i_1}, a_{i_2}, \ldots, a_{i_m}$, separating its elements by single spaces.

Examples

standard input	standard output
7 3 2 4 3 1 3 5 3 < > =	6 2 4 3 3 5 3

Problem I. Sheep

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

The habitants of the Byteotian Highland bred sheep for centuries. Every sane shepherd has a fenced pasture in the shape of a convex polygon[1] for the sheep to graze on. Every sane sheep in turn has its own favourite feeding spot on the pasture where it spends all days. Sometimes however, the sheep want to play. As they play in pairs, every shepherd keeps an even number of sheep, so that his every sheep has a partner to play with.

The shepherds are concerned about a decree recently issued by the Byteburg's High Commissioner for Agriculture. The decree states that as of the next year the sheep can only graze on triangle-shaped pastures. Thus every shepherd whose pasture is an n-gon for $n > 3$ is to partition it into triangles by putting $n - 3$ fences inside. Each single new fence, of course, is going to be a segment connecting two vertices of the polygon (pasture). Additionally, the fences can intersect only in these vertices. A shepherd who does not fulfil these requirements will no longer be subsidized.

Byteasar, as a shepherd, has to decide on a way of partitioning his pasture. In fact, he is unsure how many partitions are possible. He is only interested in such partitions that no fence is drawn through a favourite spot of any sheep, and such that every resulting triangle contains the favourite spots of an even number of sheep, so that these sheep can play in pairs. Help Byteasar by writing a program that calculates the number of such partitions!

Input

The first line of the standard input contains three integers n, k and m ($4 \leq n \leq 600$, $2 \leq k \leq 20\,000$, $2 \mid k$, $2 \leq m \leq 20\,000$), separated by single spaces, that denote respectively: the number of vertices of the polygon forming the pasture, the number of the sheep, and a certain positive integer m. Each of the following n lines contains two integers x_i and y_i ($-15\,000 \leq x_i, y_i \leq 15\,000$), separated by a single space, denoting the coordinates of the i-th vertex of the pasture. The vertices are given in a clockwise order. Each of the k lines that follow holds two integers p_j, q_j ($-15\,000 \leq p_j, q_j \leq 15\,000$), separated by a single space, denoting the coordinates of the favourite spot of the j-th sheep. All the favourite spots lie strictly inside (i.e., not on the boundary) the pasture.

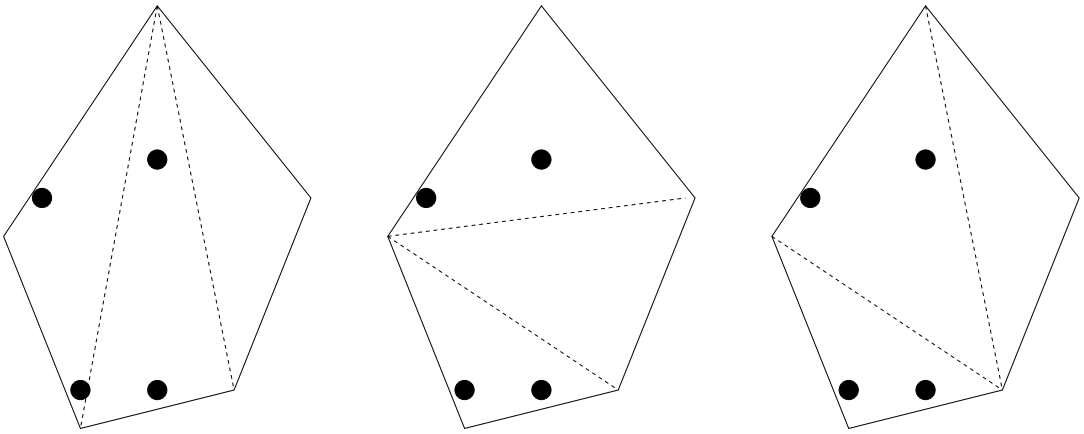
Output

Your program should print one integer on the standard output, namely the remainder of division by m of the number of partitions of the pasture in triangles, such that no fence is drawn through a favourite spot of any sheep, and every resulting triangle contains the favourite spots of an even number of sheep.

Examples

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

[1] A convex polygon is a simple polygon (i.e., one with no self-intersections), in which every internal angle is smaller than 180°.



The figure depicts three possible partitions into triangles. The favourite spots of the sheep are marked with dots.

Problem J. Teleportation

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

King Byteasar is the ruler of the whole solar system that contains n planets. This number is so large that people have abandoned the silly custom of naming the planets and use numbers instead. The planets are thus conveniently numbered from 1 to n. Byteasar's palace is on the planet no. 1, while his military base on the planet no. 2. A long time ago Byteasar had a teleportation portal established between these two planets, which allows travelling from either planet to another in two hundred and fifty minutes (slightly over four hours).

Nowadays the teleportation technology is more mature, and the recent teleportation devices shorten the travel time to just a single hour. Let us note here, that all the portals, both the Byteasar's old one and the new ones available on the market, are of course bidirectional, and that the teleportation travel time is irrespective of the distance travelled. Some planets of the system are already connected with these new teleportation portals. In fact, it is already possible to travel between the planets no. 1 and 2 without using the king's private portal, though this involves several other portals and is thus no faster than the king's portal. Byteasar finds this rather fortunate, as he believes that such possibility would be a security breach.

The technology itself is increasingly available, and as everyone realises its economic significance, each pair of planets that are not currently directly connected with a portal are petitioning for establishing such a connection. Being a wise ruler, Byteasar intends to give his consent to as many constructions as possible, though keeping himself secure, i.e., not allowing the travel between planets 1 and 2 faster than with his private portal. Help the king determine how many portals he can agree to.

Input

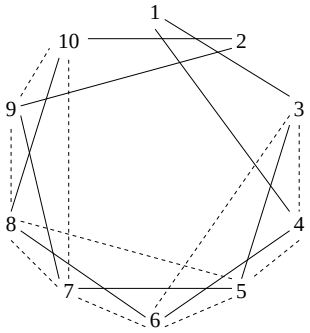
Two integers are given in the first line of the standard input, n and m ($2 \le n \le 40\,000$, $0 \le m \le 1\,000\,000$), separated by a single space, denoting the number of planets in Byteasar's realm and the number of new portals that already exist. These teleportation portals are described in the m lines that follow. Each such line contains two integers a_i and b_i ($1 \le a_i < b_i \le n$), separated by a single space, denoting that there is a teleportation portal of the new kind connecting a_i and b_i. No pair of numbers appears twice. You may assume that the existing network of new portals allows travel from planet no. 1 to planet no. 2, but in no less than 250 minutes.

Output

Your program should print out just a single integer, namely the maximum number of portals Byteasar can agree to without breaching his security.

Examples

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



Existing portals are marked with solid lines, while those Byteasar can agree to with dashed lines.