Problem A. Synchronised Alpinism

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

The progress does not stand! An institute of Sport Experiments (SE) is preparing new surprises for future Olympic Games. There is no doubt that you are familiar with synchronised swimming. But do you know anything about synchronised boxing or synchronised basketball?

In this problem we will consider only synchronised alpinism. It is a team competition, each team consists of two members. Initially one of them stands at the east side of the mountain and other — at the west side. Their aim is to meet at some point as fast as possible. There is also one additional difficulty: at any moment of time they have to be at the same height.

Unfortunatly, SE's synchronised alpinism experimental contest was unsuccessful. No team managed to fulfill the task, so it was impossible to determine a winner! Now SE is interested, is it really impossible to meet on the mountain under above restrictions or the teams just have to train harder. Try to help SE!

Input

Here we will assume that mountain is flat figure, and its border is a polyline, with a peaks at points $(1, y_1), \ldots, (n, y_n)$. The first line of the input file consists one integer number n ($3 \le n \le 100\,000$). The second line contains integer numbers $y_1, \ldots, y_n$. Points $(1, y_1)$ and (n, y_n) are initial positions of alpinists. It is guaranteed that $y_1 = 0$, $y_n = 0$ and $y_1, \ldots, y_n$ do not exceed $1\,000\,000\,000$ by absolute value.

Output

If it is possible for the alpinists to meet, output "`YES`", otherwise output "`NO`" (quotes for clarity).

Example

alpinism.in	alpinism.out
7 0 3 1 5 2 4 0	YES

Problem B. Bidirected Graph

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

The notion of bidirected graphs is used to formulate and solve wide range of combinatorial optimization problems: matchings, b-matchings, T-joins, T-paths packing and so on. Bidirected graph is a generalization of directed graph, where every arc has two orientations — one for every end.

More formally, bidirected graph is a tuple $(V, E, \text{ends}, \omega)$. V is a set of nodes, E is a set of edges, *ends* is a mapping from E: $\text{ends}(e) = \{u, v\}$, where $u, v \in V$, $u \neq v$. For every $e \in E$ and $v \in \text{ends}(e)$ we define $\omega(e, v) \in \{-1, +1\}$. It is called the direction of the edge e in the node v.

An s-t walk in a bidirected graph G is an alternating sequence

$$P = (s = v_0, e_1, v_1, \ldots, e_k, v_k = t),$$

where $v_i \in V$, $e_i \in E$, $\text{ends}(e_i) = \{v_{i-1}, v_i\}$ and $\omega(e_{i+1}, v_i)\omega(e_i, v_i) = -1$.

A directed graph without loops can be considered as bidirected, if we replace each arc $a = (u, v)$ with an edge e_a, and put $\text{ends}(e_a) = \{u, v\}$, $\omega(e_a, u) = -1$, $\omega(e_a, v) = +1$.

An elementary transformation on node v is defined as follows: if for $e \in E$, $\omega(e, v)$ is defined, then negate it. It's clear that elementary transformations don't change the set of walks in G.

We want to determine if it is possible to convert bidirected graph to "directed" graph (i.e. to graph that can be obtained from some directed graph by means of the consideration described above) using several elementary transformations. If it is, we also want to know how to do it using the minimal number of elementary transformations.

Input

The first line of the input contains two integer numbers n and m — the number of vertices and the number of edges in the graph ($1 \le n \le 100\,000$, $0 \le m \le 500\,000$). Next m lines describe the edges of the graph. Each line consists of four numbers $a_i, b_i, d_{i,1}, d_{i,2}$, where a_i and b_i are the ends of the egde e and $d_{i,1} = \omega(e, a_i), d_{i,2} = \omega(e, b_i)$; $1 \le a_i, b_i \le n$, $d_{i,j} \in \{-1, 1\}$.

Output

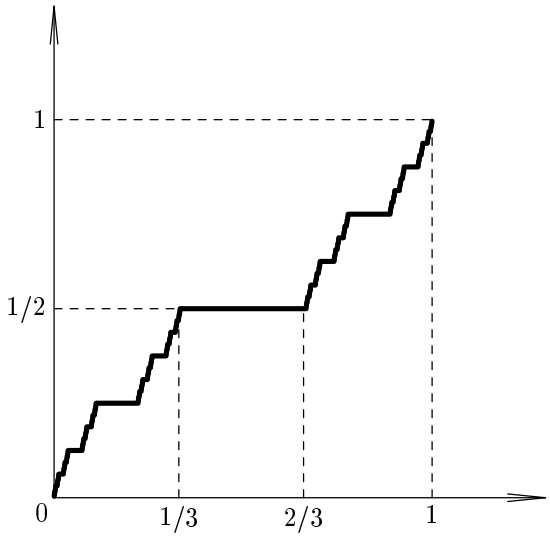
If it is impossible to perform required transformation, output should contain exactly one string "`NO`" (quotes for clarity only). Otherwise, the first line should contain string "`YES`", the second line should contain the number of elementary transformations in your solution, and the following lines should describe those transformations. Each transformation should be described by exactly one number — the index of the vertex of this transformation. If there are several answers with the minimal number of transformations, output any of them.

Examples

bidirected.in	bidirected.out
4 3 1 3 -1 -1 2 3 -1 -1 3 4 1 1	YES 1 3
3 3 1 2 1 1 2 3 1 1 1 3 1 1	NO

Problem C. Cantor Function

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



The Cantor function $f(x)$ (see picture) is defined as the function on $[0, 1]$ as follows:

1. If x belongs to Cantor set ($x = \sum_i 2 \cdot 3^{-n_i}$ where n_i are different positive integers), then $f(x) = \sum_i 2^{-n_i}$.
2. f is continuous and monotonous function.

In 2004, Gorin and Kukushkin showed that $I_n = \int_0^1 f^n(x)\,dx$ is rational. You are to find I_n and output it as irreducible fraction.

Input

First line contains one integer number n ($0 \le n \le 50$).

Output

You should output I_n in the form p/q, where p and q are the numerator and the denominator of I_n respectively. Note that p and q must be natural and (p, q) must be equal to 1. You should output p and q without leading zeroes.

Examples

cantor.in	cantor.out
0	1/1
1	1/2
2	3/10

Problem D. Caravans

Input file:	caravans.in
Output file:	caravans.out
Time limit:	6 seconds
Memory limit:	256 megabytes

In this task your goal is to prey upon caravans.

There are n oases in the desert (for our purposes they are points on the plane). Sometimes caravans go from one oasis to another one. In order to prey upon them, you should predict their paths. But how to do it? The answer was given by Nomad. Caravans' velocity is constant, and they try to minimize maximal period of time outside oases. So, you can conclude, that the optimal path is a polyline. You are given several pairs of oases, and you are to output length of the maximal segment of the optimal path of a caravan which starts its way from the first oasis of the pair and ends in the second one. All oases have distinct locations and there are three oases that do not belong to one line.

Input

First line of the input contains n — amount of oases ($3 \le n \le 100\,000$). The following n lines describe them. Each line contains two integer numbers — x_i and y_i ($0 \le x_i, y_i \le 10\,000$). Next line contains one integer number q — amount of caravans ($1 \le q \le 100\,000$). The next q lines contain start and end oases of caravans — s_i and t_i ($1 \le s_i, t_i \le n$).

Output

Output q lengths with relative or absolute error 10^{-9} — one number at a line.

Example

caravans.in	caravans.out
3 0 0 50 10 150 0 3 1 2 1 3 2 3	50.9901951359 100.4987562112 100.4987562112

Problem E. Country

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

Once upon a time in a faraway kingdom there was a very wise King. This King wanted to improve the road system of the country. For the purposes of this problem the road system can be considered as an undirected graph without loops and parallel edges (the vertices of this graph correspond to the cities and the edges correspond to the roads). The King was fond of mathematics, so he reformed the road system of the kingdom in the following mathematical way: for each two distinct cities there was exactly one common neighbour (any city is not neighbour of itself). The road reform was successful, but with the lapse of time some roads were destroyed. The government of the kingdom wants to know the distances between cities at any moment of time. Your task is to help the government with this problem.

Input

The first line of the input contains two integer numbers n and m — the number of cities and the number of roads immediately after the reform ($3 \le n \le 100\,000$). The next m lines describe the roads of the kingdom. Each of these lines contains two integer numbers x_i and y_i — 1-based indices of cities connected by this road. The following lines describe queries of the government and information about road destruction. If the line describes a destruction of the road, it will be formatted as "`DELETE` x", where x is 1-based index of the road. Otherwise it will be formated as "`LENGTH` x y", where x and y are 1-based indices of cities. The queries will appear in the input till the end of the file. The total number of queries doesn't exceed 200 000.

Output

For each "`LENGTH`" query output the answer on the separate line, or -1 if there's no way to travel from one city to another.

Example

country.in	country.out
3 3 1 2 2 3 3 1 LENGTH 1 2 DELETE 1 LENGTH 1 2 LENGTH 2 3 DELETE 3 LENGTH 1 2	1 2 1 -1

Problem F. Highlander

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

Football judge is a very important profession. To make stern but fair decisions during the match all judges must be in good shape. For this purpose they regularly have special judge trainings. One of the most popular trainings is a game called "Highlander". The rules of this game are quite simple. Initially, each judge receives one red card with the name of some other judge (no judge gets the card with his own name, and all the cards are different). Then, the game starts. Judges can run anywhere on the football field. The purpose is to catch the judge whose name is written on the card and show the card to him. When judge A catches judge B, judge B loses and doesn't play anymore, and A gets all his cards. Then the game continues. The game ends when the distribution of the cards reaches such a state that it is impossible to catch anybody because no judge has the card with the name of another judge who is still playing. The winner is the judge who gets the most number of cards. If several players get the same maximal amount of cards, they are all considered to be winners.

It is clear that after the distribution of cards some judges have no chance to win. Your task is to determine the expected number of judges that have a theoretical chance to win. All transpositions of cards where no judge gets the card with his name are considered equiprobable.

Input

Input file contains exactly one integer number n — the number of judges in the game ($2 \le n \le 100$).

Output

Output should contain one real number with relative or absolute error 10^{-9} — the answer to the problem.

Example

highlander.in	highlander.out
2	2.00

Problem G. Happy Birthday, Jedi Knight!

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

Jedi Knight has sneaked into the new model of Death Star. He is searching for very important enemy documents. For each document he finds he will get a certain amount of money from the Jedi Council.

A new model of Death Star has the form of an n-dimensional parallelepiped. Its edges are parallel to vectors $v_1, \ldots, v_n$ with integer coordinates and have the lengths equal to lengths of the corresponding vectors. There is one document located in every integer point inside or on the border of the Death Star. If for any k ($0 \le k \le n$) a document is inside some k-dimensional facet of the parallelepiped, but either $k = 0$ or it is not inside any $(k-1)$-dimensional facet, then its price is 2^k.

Your task is to calculate the total amount of money Jedi can earn if he gets all the documents on the Death Star. The answer can be enormous, but Jedi isn't afraid of this fact, so you should output it modulo prime number p.

Input

The first line of the input file contains numbers n and p ($2 \le n \le 50$, $2 \le p \le 10\,007$). The next n lines contain description of vectors v_i. Each of these lines contains n integer numbers a_{ij} ($0 \le a_{ij} \le p-1$) — coordinates of the vector v_i. It is guaranteed that these vectors are linearly independent.

Output

Output must contain one number — the answer modulo p.

Example

jedi.in	jedi.out
2 43 1 0 0 1	4

Problem H. Lazy Judges

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

It's no secret that in order to build a baseball stadium one needs to create a project first. The chief engineer in the town NN is preparing such a project now and he has met some difficulties. There is only one brigade of baseball judges in NN, and they are very lazy, so they have some restrictions for the project. As you know, a baseball field has a form of square with bases in its corners. There is one chief judge in the brigade and he wants the center of the baseball field to be located at a fixed point $(0, 0)$ (he will stay at this point and see all the course of the game). All the other n judges will also stay at some predefined points. Each of them will see only one segment with ends in the points with coordinates (x_{i_1}, y_{i_1}) and (x_{i_2}, y_{i_2}). It is also important that every base should belong to at least one of these segments. If there are several ways to build the baseball field, the chief engineer acts in the following way. He considers the set of all possible positions of the first base and chooses one of them randomly with a uniform distribution.

Your task is to calculate the expected area of the field.

Input

The first line of the input file contains an integer number n, $1 \le n \le 50$. Each of the following n lines contains four integer numbers $x_{i_1}, y_{i_1}, x_{i_2}, y_{i_2}$ — coordinates of ends of the i-th segment. All coordinates do not exceed 100 by absolute value. It is guaranteed that there is at least one way to build the baseball field. All segments are non-degenerate. They may intersect, but it is guaranteed that they do not overlap.

Output

Output should contain one real number — the expected area of the field with relative or absolute error 10^{-9}.

Example

lazy.in	lazy.out
3 -3 -1 3 -1 -3 -1 0 2 3 -1 0 2	4.0000000000

Problem I. Soap Opera

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

Brazil script writers are inexhaustible! Recently two greatest writers don Juan and donna Rosa have married. And now they are writing their shared scenario with a great happy end. So they want to maximize the amount of marriages at the end of the serial. In spite of their sympathy, Juan and Rosa have different points of view at the cinema, so there are some pairs of actors (of course, pair is a man and woman), such that Juan considers them photogenic, but Rosa does not and vise versa. In order to settle, newlyweds decided to select such a troup that both Juan and Rosa can combine marriages at the scenario in such a way that each actor will be married. You will be given the number of actors n and sets of successful pairs in the views of Juan and Rosa. Your task is to maximize the size of the troup.

Input

First line of the input file contains three integer numbers n, m_1 and m_2 — the number of actors, number of photogenic pairs in the views of Juan and Rosa respectively, $2 \le n \le 100$. Each of the following m_1 lines contains a pair of integer numbers between 1 and n — successful pair in the view of Juan. It is guaranteed that all Juan's pairs will be different. Next m_2 lines describe Rosa's opinion in the same manner.

Output

In the first line output one integer number k — the maximal possible number of marriages at the end of the serial. Each of the following k lines should contain description of possible marriages in the view of Juan, and then output k lines — description in the view of Rosa.

Example

soap.in	soap.out
4 2 2 1 3 2 4 1 4 2 3	2 1 3 2 4 3 2 4 1

Problem J. Strange Planet

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

Once upon a time, there was an n-dimensional space. And there was a planet, quite a strange planet. One of its strange features was its form — n-dimensional hypercube with unit length side. And there was a strange town in each vertex of the planet. Territory of the planet was divided between three aggressive kingdoms. But several towns preserved their independence — let's call them neutral. An i-th town was neutral if $d_1(i) = d_2(i) = d_3(i)$, where $d_j(i)$ is the distance between i-th town and the capital of j-th kingdom. All distances are measured using Manhattan metrics, because the planet was really strange. You should calculate amount of neutral towns in order to estimate trading potential of the planet. Answer can be quite big, so output it modulo $10^9 + 7$.

Input

First three lines describe the positions of the capitals. Each description is an n-bit string ($1 \le n \le 100\,000$).

Output

You should output the amount of the neutral towns modulo $10^9 + 7$.

Examples

strange.in	strange.out
01 01 10	2
01 01 11	0

Problem K. Tickets

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

Conductor is quite a boring profession, as all you have to do is just to sell tickets to the passengers. So no wonder that once upon a time in a faraway galaxy one conductor decided to diversify this occupation. Now this conductor sells several tickets at a time to each passenger. More precisely, he successively gives tickets to the passenger until the sum of the digits on all the tickets given becomes not less than some integer number k. Then this process repeats for the next passenger. Initially conductor has a tape of tickets numbered successively from l to r, inclusive. This way of tickets distribution is quite good, because passengers are glad to get several tickets when they pay only for one. But there is one disadvantage. Since each passenger gets several tickets, it is possible that conductor won't be able to serve all passengers. Your task is to help conductor in this difficult situation. You should calculate how many passengers is the conductor able to serve.

Input

Input file contains three integer numbers l, r and k ($1 \le l \le r \le 10^{18}$, $1 \le k \le 1000$).

Output

Output should contain exactly one number — the answer to the problem.

Example

tickets.in	tickets.out
40 218 57	29

Problem L. Mr. X

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

Mr. X (he is famous for his 13 feats) has a piece of chequered paper. Some of the cells are marked with a special sign. He can fold the piece of paper anywhere along a line between two successive rows or columns. After the fold is performed, he considers a folded paper as a new piece of paper, so he may continue the folding process. Now Mr. X wonders whether it is possible to combine all the marked cells together performing some sequence of folds (marked cells must be all one under another, and no unmarked cell must be under or above the marked cells). So he asks for your help.

Input

The first line of the input file contains integers n, m and k — number of rows, columns and marked cells correspondingly ($1 \le n, m \le 100\,000, 0 \le k \le 100\,000$). Each of the following k lines describes one marked cell and contains two integer numbers x_i and y_i that are row and column indices ($1 \le x_i \le n$, $1 \le y_i \le m$).

Output

If it is possible for Mr. X to combine all the marked cells together, output "`YES`", otherwise output "`NO`" (quotes for clarity).

Examples

x.in	x.out
4 4 4 1 1 4 1 1 4 4 4	YES
4 4 3 1 1 4 1 1 4	NO