Problem A. Chinese Girls' Amusement

Input file:	`china.in`
Output file:	`china.out`
Time limit:	1 second

You must have heard that the Chinese culture is quite different from that of Europe or Russia. So some Chinese habits seem quite unusual or even weird to us.

So it is known that there is one popular game of Chinese girls. N girls stand forming a circle and throw a ball to each other. First girl holding a ball throws it to the K-th girl on her left ($1 \le K \le N/2$). That girl catches the ball and in turn throws it to the K-th girl on her left, and so on. So the ball is passed from one girl to another until it comes back to the first girl. If for example $N = 7$ and $K = 3$, the girls receive the ball in the following order: 1, 4, 7, 3, 6, 2, 5, 1.

To make the game even more interesting the girls want to choose K as large as possible, but they want one condition to hold: each girl must own the ball during the game.

Input

Input file contains one integer number N ($3 \le N \le 10^{2000}$) — the number of Chinese girls taking part in the game.

Output

Output the only number — K that they should choose.

Example

china.in	china.out
7	3
6	1

Problem B. Reactor Cooling

Input file:	`cooling.in`
Output file:	`cooling.out`
Time limit:	1 second

The terrorist group leaded by a well known international terrorist Ben Bladen is buliding a nuclear reactor to produce plutonium for the nuclear bomb they are planning to create. Being the wicked computer genius of this group, you are responsible for developing the cooling system for the reactor.

The cooling system of the reactor consists of the number of pipes that special cooling liquid flows by. Pipes are connected at special points, called nodes, each pipe has the starting node and the end point. The liquid must flow by the pipe from its start point to its end point and not in the opposite direction.

Let the nodes be numbered from 1 to N. The cooling system must be designed so that the liquid is circulating by the pipes and the amount of the liquid coming to each node (in the unit of time) is equal to the amount of liquid leaving the node. That is, if we designate the amount of liquid going by the pipe from i-th node to j-th as f_{ij}, (put $f_{ij} = 0$ if there is no pipe from node i to node j), for each i the following condition must hold:

$$\sum_{j=1}^{N} f_{ij} = \sum_{j=1}^{N} f_{ji}$$

Each pipe has some finite capacity, therefore for each i and j connected by the pipe must be $f_{ij} \le c_{ij}$ where c_{ij} is the capacity of the pipe. To provide sufficient cooling, the amount of the liquid flowing by the pipe going from i-th to j-th nodes must be at least l_{ij}, thus it must be $f_{ij} \ge l_{ij}$.

Given c_{ij} and l_{ij} for all pipes, find the amount f_{ij}, satisfying the conditions specified above.

Input

The first line of the input file contains the number N ($1 \le N \le 200$) - the number of nodes and and M — the number of pipes. The following M lines contain four integer number each - i, j, l_{ij} and c_{ij} each. There is at most one pipe connecting any two nodes and $0 \le l_{ij} \le c_{ij} \le 10^5$ for all pipes. No pipe connects a node to itself. If there is a pipe from i-th node to j-th, there is no pipe from j-th node to i-th.

Output

On the first line of the output file print `YES` if there is the way to carry out reactor cooling and `NO` if there is none. In the first case M integers must follow, k-th number being the amount of liquid flowing by the k-th pipe. Pipes are numbered as they are given in the input file.

Example

cooling.in	cooling.out
4 6 1 2 1 2 2 3 1 2 3 4 1 2 4 1 1 2 1 3 1 2 4 2 1 2	NO
4 6 1 2 1 3 2 3 1 3 3 4 1 3 4 1 1 3 1 3 1 3 4 2 1 3	YES 1 2 3 2 1 1

Problem C. New Year Bonus Grant

Input file:	`grant.in`
Output file:	`grant.out`
Time limit:	1 second

All programmers of *Mocrosoft* software company are organized in a strict subordination hierarchy. Every programmer has exactly one chief, except Bill Hates who is also the head of the company and has no chief.

Due to the celebration of the new 2003 year, chief accountant of *Mocrosoft* decided to pay a New Year Bonus Grant of 1000 dollars to some programmers. However being extremely concerned of the company wealth she would like to designate the least possible amount of money for these grants. On the other hand she didn't want to be accused of being too greedy or of giving preferences to some programmers. To do this, she developed the following scheme of grants appointment:

- Each programmer may either assign a grant to one of his subordinates or have a grant assigned to him by his chief or none of the above.
- No programmer can simultaneously receive a grant and assign a grant to one of his subordinates.
- No programmer can assign a grant to more than one of his subordinates

The scheme seemed to be designed perfectly — nobody would like to assign a grant to anybody since in this case he himself would not receive money. But programmers somehow discovered the plan of chief accountant and decided to make a trick to get the most money possible and share them fairly afterwards. The idea was to make such grant assignments that the total amount of grant money received is maximum possible.

You were selected to write the program which will find the optimal grants appointment.

Input

The first line of the input file contains integer N — the number of programmers in *Mocrosoft* company ($2 \le N \le 500\,000$). Each programmer is assigned his unique identifier — integer number ranging from 1 to N. Bill Hates has number 1 and each programmer has the number greater then the number of his chief. The second line of the input file contains $N - 1$ integers, i-th of which being the number of the chief of the worker whose number is $(i + 1)$.

Output

On the first line of the output file print the maximum possible amount of money workers can get. On the second line output the numbers of programmers that will receive grant in ascending order.

Example

grant.in	grant.out
4 1 1 2	2000 3 4

Problem D. Matrix Multiplication

Input file:	`matrix.in`
Output file:	`matrix.out`
Time limit:	1 second

Let us consider undirected graph $G = \langle V, E \rangle$ which has N vertices and M edges. Incidence matrix of this graph is $N \times M$ matrix $A = \{a_{ij}\}$, such that a_{ij} is 1 if i-th vertex is one of the ends of j-th edge and 0 in the other case. Your task is to find the sum of all elements of the matrix $A^T A$.

Input

The first line of the input file contains two integer numbers — N and M ($2 \le N \le 10\,000$, $1 \le M \le 100\,000$). $2M$ integer numbers follow, forming M pairs, each pair describes one edge of the graph. All edges are different and there are no loops (i.e. edge ends are distinct).

Output

Output the only number — the sum requested.

Example

matrix.in	matrix.out
4 4 1 2 1 3 2 3 2 4	18

Problem E. Nice Patterns Strike Back

Input file:	`nice.in`
Output file:	`nice.out`
Time limit:	1 second

You might have noticed that there is the new fashion among rich people to have their yards tiled with black and white tiles, forming a pattern. The company *Broken Tiles* is well known as the best tiling company in our region. It provides the widest choices of nice patterns to tile your yard with. The pattern is *nice* if there is no square of size 2×2, such that all tiles in it have the same color. So patterns on the figure 1 are nice, while patterns on the figure 2 are not.

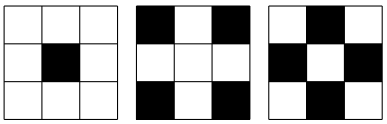

Figure 1.

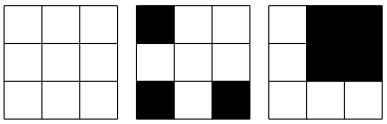

Figure 2.

The president of the company wonders whether the variety of nice patterns he can provide to the clients is large enough. Thus he asks you to find out the number of nice patterns that can be used to tile the yard of size $N \times M$. Now he is interested in the long term estimation, so he suggests $N \leq 10^{100}$. However, he does not like big numbers, so he asks you to find the answer modulo P.

Input

The input file contains three integer numbers: N ($1 \leq N \leq 10^{100}$), M ($1 \leq M \leq 5$) and P ($1 \leq P \leq 10000$).

Output

Write the number of nice patterns of size $N \times M$ modulo P to the output file.

Example

nice.in	nice.out
2 2 5	4
3 3 23	0

Problem F. Get Out!

Input file:	`out.in`
Output file:	`out.out`
Time limit:	1 second

Captain Faraway on his famous circular ship *Kolobok* is lost among the islands of the archipelago that he has just discovered. No he wonders whether he can get out of there. Help him!

All islands in the archipelago can be composed of pieces that have circular form. You are given the map of archipelago and the position of captain. Find out whether captain can get out of there, i.e. can get as far from the point he is in the beginning as he likes.

Input

The first line contains N — the number of circular island parts ($1 \le N \le 300$). N lines follow, each containing x_i, y_i, r_i — coordinates of center and radius of the i-th circle. All coordinates and radii are real. Objects may overlap with each other in arbitrary way. All objects are considered solid.

The last line of the input file contains three real numbers — coordinates of the center of *Kolobok* and its radius.

You may consider *Kolobok* to be the perfect circle and that it is in the free area in the beginning. *Kolobok* can move along any trajectory and is so strong that he can even touch islands, but no nonzero part of island must intersect *Kolobok* during his motion. You may assume that making calculations with the precision of 10^{-6} is satisfactory.

Output

Output `YES` if *Kolobok* can leave archipelago and `NO` if it cannot.

Example

out.in	out.out
7 2 2 1.1 -2 2 1.1 2 -2 1.0 -2 -2 1.0 2 -5 1.0 0 -8 1.0 -2 -6 1.0 0 0 1	YES
5 2 2 1.1 -2 2 1.1 2 -2 1.0 -2 -2 1.0 0 -3 0.01 0 0 1	NO

Problem G. Beautiful People

Input file:	`people.in`
Output file:	`people.out`
Time limit:	1 second

The most prestigious sports club in one city has exactly N members. Each of its members is strong and beautiful. More precisely, i-th member of this club (members being numbered by the time they entered the club) has strength S_i and beauty B_i. Since this is a very prestigious club, its members are very rich and therefore extraordinary people, so they often extremely hate each other. Strictly speaking, i-th member of the club Mr X hates j-th member of the club Mr Y if $S_i \le S_j$ and $B_i \ge B_j$ or if $S_i \ge S_j$ and $B_i \le B_j$ (if both properties of Mr X are greater then corresponding properties of Mr Y, he doesn't even notice him, on the other hand, if both of his properties are less, he respects Mr Y very much).

To celebrate a new 2003 year, the administration of the club is planning to organize a party. However they are afraid that if two people who hate each other would simultaneouly attend the party, after a drink or two they would start a fight. So no two people who hate each other should be invited. On the other hand, to keep the club prestige at the apropriate level, administration wants to invite as many people as possible.

Being the only one among administration who is not afraid of touching a computer, you are to write a program which would find out whom to invite to the party.

Input

The first line of the input file contains integer N — the number of members of the club. ($2 \le N \le 100\,000$). Next N lines contain two numbers each — S_i and B_i respectively ($1 \le S_i, B_i \le 10^9$).

Output

On the first line of the output file print the maximum number of the people that can be invited to the party. On the second line output N integers — numbers of members to be invited in arbitrary order. If several solutions exist, output any one.

Example

people.in	people.out
4 1 1 1 2 2 1 2 2	2 1 4

Problem H. Cracking' RSA

Input file:	`rsa.in`
Output file:	`rsa.out`
Time limit:	1 second

The following problem is somehow related to the final stage of many famous integer factorization algorithms involved in some cryptoanalytical problems, for example cracking well-known RSA public key system.

The most powerful of such algorithms, so called quadratic sieve descendant algorithms utilize the fact that if $n = pq$ where p and q are large unknown primes needed to be found out, then if $v^2 = w^2 \pmod{n}$ and $u \neq \pm v \pmod{n}$ then $\gcd(v + w, n)$ is a factor of n (either p or q).

Not getting further in the details of these algorithms, let us consider our problem. Given m integer numbers b_1, b_2, ..., b_m such that all their prime factors are from the set of first t primes, the task is to find such $S \subset \{1, 2, \ldots, m\}$ that $\prod_{i \in S} b_i$ is a perfect square i.e. equal to u^2 for some integer u. Given such S we get one pair for testing (product of S elements stands for v when w is known from other steps of algorithms which are of no interest to us, testing performed is checking whether pair is nontrivial, i.e. $u \neq \pm v \pmod{n}$). Since we want to factor n with maximum possible probability, we would like to get as many such sets as possible. So the interesting problem could be to calculate the number of all such sets. This is exactly your task.

Input

The first line of the input file contains two integers t and m ($1 \leq t \leq 100$, $1 \leq m \leq 100$). The second line of the input file contains m integer numbers b_i such that all their prime factors are from t first primes (for example, if $t = 3$ all their prime factors are from the set $\{2, 3, 5\}$). $1 \leq b_i \leq 10^9$ for all i.

Output

Output the number of non-empty subsets of the given set b_i, the product of numbers from which is a perfect square

Example

rsa.in	rsa.out
3 4 9 20 500 3	3