Problem A. Hierarchy

Input:	*standard input*
Output:	*standard output*
Time limit:	1 second
Memory limit:	64 mebibytes

Nick's company employed n people. Now Nick needs to build a tree hierarchy of "supervisor-surbodinate" relations in the company (this is to say that each employee, except one, has exactly one supervisor). There are m applications written in the following form: *"employee a_i is ready to become a supervisor of employee b_i at extra cost c_i"*. The qualification q_j of each employee is known, and for each application the following is true: $q_{a_i} > q_{b_i}$.

Would you help Nick calculate the minimum cost of such a hierarchy, or find out that it is impossible to build it.

Input

The first input line contains integer n ($1 \leq n \leq 1000$) — amount of employees in the company. The following line contains n space-separated numbers q_j ($0 \leq q_j \leq 10^6$)— the employees' qualifications. The following line contains number m ($0 \leq m \leq 10000$) — amount of received applications. The following m lines contain the applications themselves, each of them in the form of three space-separated numbers: a_i, b_i and c_i ($1 \leq a_i, b_i \leq n$, $0 \leq c_i \leq 10^6$). Different applications can be similar, i.e. they can come from one and the same employee who offered to become a supervisor of the same person but at a different cost. For each application $q_{a_i} > q_{b_i}$.

Output

Output the only line — the minimum cost of building such a hierarchy, or `-1` if it is impossible to build it.

Examples

standard input	standard output

4
7 2 3 1
4
1 2 5
2 4 1
3 4 1
1 3 5
	11

3
1 2 3
2
3 1 2
3 1 3
	-1

Note

In the first sample one of the possible ways for building a hierarchy is to take applications with indexes 1, 2 and 4, which give 11 as the minimum total cost. In the second sample it is impossible to build the required hierarchy, so the answer is `-1`.

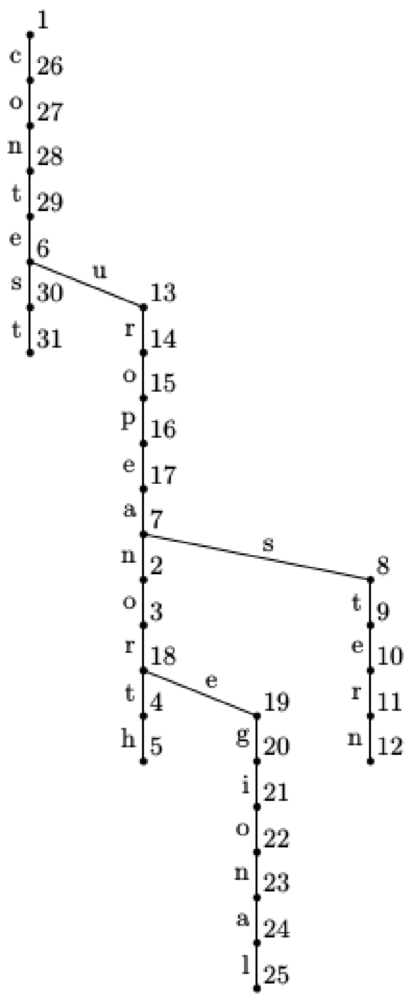
Problem B. Dictionary

Input:	*standard input*
Output:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Petr and Dmitry are working on a novel data compression scheme. Their task is to compress a given set of words. To compress a given set of words they have to build a rooted tree. Each edge of the tree is marked with exactly one letter.

Let us define a dictionary that is produced by this kind of tree as a set of words that can be constructed by concatenating letters on edges on any path from any vertex in the tree (not necessarily root) and going away from root down to the leaves (but not necessarily finishing on a leaf).

Boys have to construct such a tree with a dictionary that is a superset of the set of words that they are given to compress. This tree should have the smallest number of vertices between trees that satisfy the above condition. Any tree with the same number of vertices will do. Your task is to help them.



For example, in a tree on the picture above with the root marked as 1, a path from 7 to 5 reads "`north`", a path from 16 to 12 reads "`eastern`", a path from 29 to 2 reads "`european`", a path from 3 to 25 reads "`regional`", and a path from 1 to 31 reads "`contest`".

Input

The first line of the input file contains the number of words in a given set n ($1 \le n \le 50$). The following n lines contain different non-empty words, one word per line, consisting of lowercase English letters. The length of each word is at most 10 characters.

Output

On the first line output the number of vertices in the tree m. The following m lines shall contain descriptions of tree vertices, one description per line. Vertices are indexed from 1 to n in the order of their corresponding description lines. If the corresponding vertex is a tree root, then its description line shall contain a single integer number 0, otherwise its description line shall contain an index of its parent node and a letter on the edge to its parent node, separated by a space.

Example

standard input	standard output
5	31
north	0
eastern	18 n
european	2 o
regional	3 r
contest	4 t
	5 h
	29 e
	16 a
	8 s
	9 t
	10 e
	7 u
	12 r
	13 o
	14 p
	15 e
	8 n
	11 r
	18 e
	19 g
	20 i
	21 o
	22 n
	23 a
	24 l
	1 c
	26 o
	27 n
	28 t
	7 s
	30 t

Problem C. (Smurf)Land protection

Input:	*standard input*
Output:	*standard output*
Time limit:	2.5 seconds
Memory limit:	512 mebibytes

In SmurfLand there are n Smurf villages and m roads connecting them. Each road can be used to transport goods only in one direction. Some roads may lead from some village to itself and there might be more than one road connecting a pair of villages. The Smurfs have developed trade unions. Each trade union is a maximal subset of villages with the property that it is possible to transport goods from any village to any other village inside that trade union. Gargamel is planning to destroy one of the villages. It would be a disaster if after the village is destroyed the number of trade unions would have to increase. Help Smurfs decide which villages will need to be protected to ensure that the disaster doesn't happen.

Input

The first line of input contains two integers n and m ($1 \le n \le 2 \cdot 10^5$, $1 \le m \le 5 \cdot 10^5$) – the number of villages and roads. The next m lines describe the roads: each line contains two integers u, v ($1 \le u, v \le n$) specifying that there is a road directly connecting villages with numbers u and v.

Output

Ouput n lines: ith line should contain the word "YES" (without quotes) if Smurfs must protect ith village or the word "NO" otherwise.

Examples

standard input	standard output

7 9
1 2
3 1
2 3
1 3
3 5
3 4
4 1
5 6
6 5
	YES
	NO
	YES
	NO
	NO
	NO
	NO

Problem D. Redundant Edges

Input:	`redundant.in`
Output:	`redundant.out`
Time limit:	1 second
Memory limit:	256 megabytes

Consider a directed graph G consisting of N nodes and M directed edges. Nodes are numbered from 1 to N and edges are numbered from 1 to M. Let's select some node R as the starting one and find all nodes that are reachable from R by moving along edges (denote this set as C_0). Define $C(e)$ as the set of nodes which are reachable from R using any edges except the one with number e. Edge e is called *redundant* if $C(e)$ is equal to C_0.

You are given the graph G and the starting node R. Your goal is to find all *redundant* edges.

Input

The first line of input contains three integers N, M and R ($1 \le N \le 100\,000$, $1 \le M \le 200\,000$, $1 \le R \le N$): the number of nodes, the number of edges and the starting node, respectively. Next M lines describe the edges of the graph: i-th of them contains two integers a_i and b_i ($1 \le a_i, b_i \le N$) which introduce a directed edge from node a_i to node b_i. It is guaranteed that there are no loops, and for any two nodes u and v, there is at most one edge from u to v.

Output

On the first line, print the number of *redundant* edges. On the second line, print the numbers of all *redundant* edges in any order. Edges are numbered starting from 1 according to their order in the input.

Examples

`redundant.in`	`redundant.out`

`3 3 1`
`1 2`
`2 3`
`3 1`
	`1`
	`3`

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

Problem E. Winnie-the-Pooh Needs Help

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

You know Winnie-the-Pooh has only sawdust in his head, that's why he isn't afraid of anything. To tell the truth he isn't afraid of anything except *heffalumps* and *woozles*. If he sees a woozle sitting on a trail, Winnie-the-Pooh will not use the trail and will find a roundabout route.

There are n meadows in the Hundred Acre Wood, and they are connected with m one-way trails. The j-th trail connects two distinct meadows a_j, b_j and it is possible to walk it in the direction from a_j to b_j. If Winnie-the-Pooh wants to go from the a_j-th meadow to the b_j-th meadow but sees a woozle on the j-th trail, he will go round the trail by going the shortest route connecting a_j and b_j which doesn't go through the j-th trail.

Winnie-the-Pooh is very curious and cautious at the same time. That is why for each trail j he wants to know the length of the shortest route connecting a_j and b_j which doesn't use the j-th trail. Winnie-the-Pooh is just a toy bear and he can't write a program to find the required value for each trail. Help him and write a program for him.

Input

The input contains one or more testcases. Each testcase starts with a line containing a pair of integers n and m ($2 \le n \le 900; 1 \le m \le 150000$), where n is the number of meadows and m is the number of trails. The meadows are numbered from 1 to n. The following m lines contain the descriptions of trails, one description per line. The j-th description contains two integers a_j, b_j ($1 \le a_j, b_j \le n$; $a_j \ne b_j$), where a_j and b_j are meadows connected by the j-th trail. It is possible to walk the j-th trail in the direction from a_j to b_j but not vice versa. For each pair (a, b) there is at most one trail from a to b.

The total number of meadows in all testcases doesn't exceed 900, the total number of trails doesn't exceed 150000.

Output

For each testcase print the sequence $f_1, f_2, \ldots, f_m$ on a single line. The value f_j stands for the number of trails in a shortest route which starts at a_j, finishes at b_j and avoids the j-th trail. Print `0` if there is no such route.

Examples

standard input	standard output

3 3
1 2
2 3
1 3
3 3
1 2
2 1
1 3
3 6
1 2
2 1
1 3
3 1
3 2
2 3
	0 0 2
	0 0 0
	2 2 2 2 2 2

Problem F. Ear Decomposition

Input:	*standard input*
Output:	*standard output*
Time limit:	2 seconds
Memory limit:	64 megabytes

Consider a connected undirected graph with no loops or parallel edges. Let us call the number of edges incident to vertex v its *degree* and denote it as $\deg v$.

A simple path $v_0 - v_1 - \ldots - v_k$ such that $\deg v_0 \geq 2$, $\deg v_k \geq 2$, and for all i from 1 to $k-1$ $\deg v_i = 2$, is called an *ear*. In particular, if $k = 1$, an edge $v_0 - v_1$ connecting two vertices of degree at least 2 is also an ear. The vertices v_0 and v_k can coincide.

Let us consider an ear $v_0 - v_1 - \ldots - v_k$ and remove all of its edges and intermediate vertices $(v_1, v_2, \ldots, v_{k-1})$ from the graph. This operation is called *ear cut*. If the ear has no intermediate vertices, it can be cut by simply removing its edge.

The *ear decomposition* of the graph G is the sequence of ear cuts, such that after each cut the graph remains connected, and after the last cut the remaining graph consists of a single vertex.

Given graph G find its ear decomposition or report that it doesn't have one.

Input

The first line of the input file contains n and m — the number of vertices and edges of G respectively ($2 \leq n \leq 20\,000$, $n-1 \leq m \leq 100\,000$). Let the vertices be numbered from 1 to n. The following m lines describe graph edges, each line contains two integer numbers — the numbers of vertices connected by the corresponding edge. It is guaranteed that G is connected. No two vertices of G are connected by more than one edge. No edge connects a vertex to itself.

Output

The first line of the output file must contain d — the number of cuts in the ear decomposition of G. The following d lines must describe cuts. Each cut must be described by k — the number of edges in the corresponding ear, followed by $k+1$ numbers — the vertices of the ear as they appear along it.

If there is no ear decomposition of the given graph, output $d = -1$.

Example

standard input	standard output

5 8
1 2
2 3
3 5
5 1
2 4
4 1
3 4
4 5
	4
	1 2 3
	2 4 3 5
	2 4 5 1
	3 4 1 2 4

3 2
1 2
1 3
	-1

Problem G. Algorithm of Two Chinese

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

You are given a weighted directed graph. You have to find the minimum possible value c, such that there exists a subset A of arcs of total weight c and it's possible to get from node 1 to any other node using only arcs in A.

Input

The first line of the input contains two integers n and m ($1 \le n \le 1\,000$, $0 \le m \le 10\,000$) — the number of nodes and arcs in the graph.

The next m lines describe arcs of the graph. Each arc is given by three integers a_i, b_i and w_i ($1 \le a_i, b_i \le n$; $-10^9 \le w_i \le 10^9$) — the index of starting node, the index of ending node and the weight of this arc.

Output

If there is no subset A such that it is possible to get from node 1 to any other node using only arcs in A, print "NO" in the only line of the output.

Otherwise, print "YES" in the first line of the output. The second line should contain a single integer c — the minimum possible total weight of A.

Examples

standard input	standard output
2 1 2 1 10	NO

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

Problem H. Network

Input:	*standard input*
Output:	*standard output*
Time limit:	1 second
Memory limit:	512 megabytes

In some computer company, Mouse Inc., there is very complicated network structure. There are a lot of branches in different countries, so the only way to communicate with each other is the Internet. And it's worth to say that interaction is the key to the popularity and success of the Mouse Inc.

The CEO of this company is interested now to figure out whether there is a way to attack and devastate whole structure. Only two hackers are capable to perpetrate such an outrage — Vasya and Petya, who can destroy any two channels. If after that there are at least two servers without connection between them, then they succeed.

In other words, the company is a set of servers, some of them connected with bidirectional channels. It's guaranteed that all the servers are connected directly or indirectly. The hackers' goal is to divide network into at least two parts without any connection between them. Each hacker can destroy exactly one channel. And they can't destroy the same channel together. You are asked to count the number of ways for hackers to win.

Input

There are two integer numbers (n, m) in the first line of input file: the number of servers and channels respectively $(1 \le n \le 2\,000,\ 0 \le m \le 100\,000)$. In the each of the next m lines there are exactly two numbers — the indices of servers connected by channel. Channels can connect a server to itself. There can be multiple channels between one pair of servers. The servers are numbered from 1 to n.

Output

There must be exactly one integer — the answer to the question described in the problem.

Example

standard input	standard output

3 3
1 2
2 3
3 1
	3

Problem I. Magnetic Triangles

Input:	*standard input*
Output:	*standard output*
Time limit:	1 second
Memory limit:	64 mebibytes

In the Future City all building are skyscrapers. To move between them some triples of skyscrapers are connected with special magnetic triangles. Each triangle connects exactly three skyscrapers. It is possible to move from one skyscraper to another if they are directly or indirectly connected with triangles.

The system of magnetic triangles is organized in such a way that it is possible to get from any skyscraper to any other skyscraper.

However, operating magnetic triangles is very expensive so city authorities want to pick one of them and turn it off. They ask you to determine for each of the triangles, whether it will be possible to get from any skyscraper to any other skyscraper if this particular triangle will be turned off.

Input

The first line of the input file contains two integers n and m — the number of skyscrapers and the number of magnetic triangles respectively ($3 \le n \le 100\,000$, $1 \le m \le 100\,000$). Each of the next m lines contains three distinct integers — indices of skyscrapers connected by this triangle. Skyscrapers are numbered from 1 to n. It's guaranteed that it's possible to get from any skyscraper to any other skyscraper using only existing triangles.

Output

First print the number of magnetic triangles that can't be turned off without lose of connectivity. Then print their numbers. Triangles are numbered starting from 1 in the order they appear in the input file.

Examples

standard input	standard output
3 1 1 2 3	1 1

3 2
1 2 3
3 2 1
	0

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

Problem J. Algorithm of Two Strong Chinese

Input:	*standard input*
Output:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

You are given a weighted directed graph. You have to find the minimum possible value c, such that there exists a subset A of arcs of total weight c and it's possible to get from node 1 to any other node using only arcs in A.

Input

The first line of the input contains two integers n and m ($1 \le n \le 300\,000$, $0 \le m \le 300\,000$) — the number of nodes and arcs in the graph.

The next m lines describe arcs of the graph. Each arc is given by three integers a_i, b_i and w_i ($1 \le a_i, b_i \le n$; $-10^9 \le w_i \le 10^9$) — the index of starting node, the index of ending node and the weight of this arc.

Output

If there is no subset A such that it is possible to get from node 1 to any other node using only arcs in A, print "`NO`" in the only line of the output.

Otherwise, print "`YES`" in the first line of the output. The second line should contain a single integer c — the minimum possible total weight of A.

Examples

standard input	standard output
2 1 2 1 10	NO

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

Problem K. Team Rocket Rises Again

Input:	*standard input*
Output:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

This problem is provided by `codeforces.com`. Please, be kind to other participants and do not look for the solutions in the Internet.

It's the turn of the year, so Bash wants to send presents to his friends. There are n cities in the Himalayan region and they are connected by m bidirectional roads. Bash is living in city s. Bash has exactly one friend in each of the other cities. Since Bash wants to surprise his friends, he decides to send a Pikachu to each of them. **Since there may be some cities which are not reachable from Bash's city, he only sends a Pikachu to those friends who live in a city reachable from his own city**. He also wants to send it to them as soon as possible.

He finds out the minimum time for each of his Pikachus to reach its destination city. Since he is a perfectionist, he informs all his friends with the time their gift will reach them. A Pikachu travels at a speed of 1 meters per second. His friends were excited to hear this and would be unhappy if their presents got delayed. Unfortunately Team Rocket is on the loose and they came to know of Bash's plan. They want to maximize the number of friends who are unhappy with Bash.

They do this by destroying exactly one of the other $n-1$ cities. This implies that **the friend residing in that city dies, so he is unhappy as well**.

Note that **if a city is destroyed, all the roads directly connected to the city are also destroyed and the Pikachu may be forced to take a longer alternate route**.

Please also note that only friends that are waiting for a gift count as unhappy, even if they die.

Since Bash is already a legend, can you help Team Rocket this time and find out the maximum number of Bash's friends who can be made unhappy by destroying exactly one city.

Input

The first line contains three space separated integers n, m and s ($2 \le n \le 2 \cdot 10^5$, $1 \le m \le min\left(\frac{n(n-1)}{2}, 3 \cdot 10^5\right)$, $1 \le s \le n$) — the number of cities and the number of roads in the Himalayan region and the city Bash lives in.

Each of the next m lines contain three space-separated integers u, v and w ($1 \le u, v \le n$, $u \ne v$, $1 \le w \le 10^9$) denoting that there exists a road between city u and city v of length w meters.

It is guaranteed that no road connects a city to itself and there are no two roads that connect the same pair of cities.

Output

Print a single integer, the answer to the problem.

Examples

standard input	standard output

4 4 3
1 2 1
2 3 1
2 4 1
3 1 1
	2

7 11 2
1 2 5
1 3 5
2 4 2
2 5 2
3 6 3
3 7 3
4 6 2
3 4 2
6 7 3
4 5 7
4 7 7
	4

Note

In the first sample, on destroying the city 2, the length of shortest distance between pairs of cities $(3, 2)$ and $(3, 4)$ will change. Hence the answer is 2.