Problem A: Cut the graph

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Given an undirected graph and queries sequentially applied to that graph. There are two types of queries:

- `cut` – cut the graph, i.e. remove an edge.
- `ask` – check, if two of vertices are in the same component of connectivity.

After all `cut` operations, there will be no edges left in the graph. You are to find the correct answers to all `ask` queries.

Input

The input consists of

- one line containing n, m and k ($1 \leq n \leq 5 \cdot 10^4, 0 \leq m \leq k \leq 1.5 \cdot 10^5$) – the amounts of vertices, edges and queries
- m lines describing the edges of the graph, with the i-th line containing u_i and v_i ($1 \leq u_i, v_i \leq n$) – the indices of the vertices incident to the i-th edge. The graph doesn't contain loops and duplicated edges.
- k lines containing the queries:
  - a `cut`-type query is given as "`cut` u v" ($1 \leq u, v \leq n$), it means you need to remove the edge between vertices u and v
  - an `ask` query is given as "`ask` u v" ($1 \leq u, v \leq n$), it means you need to tell if the vertices u and v are in the same component of connectivity at that moment

It is guaranteed that every edge appears in `cut` queries only once.

Output

For every `ask` query output the answer “`YES`” if the vertices are in the same component of connectivity and “`NO`” otherwise. The order of answers should be the same as that of the `ask` queries in the input.

Sample input and output

Input	**Output**
3 3 7 1 2 2 3 3 1 ask 3 3 cut 1 2 ask 1 2 cut 1 3 ask 2 1 cut 2 3 ask 3 1	YES YES NO NO

Problem B: Bees

Algorithms for Programming Contests

Restrictions

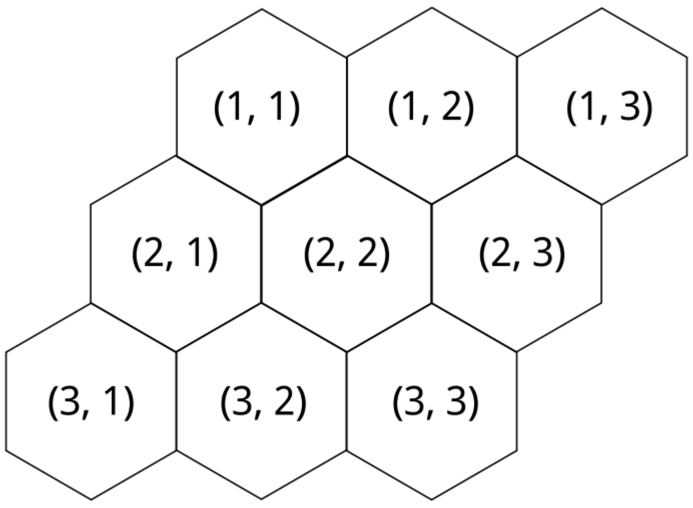
Time: 2 seconds
Memory: 512 MB

Problem description

Honeyland is a realm focused on producing honey and inhabited by bees. Obviously honeyland has the shape of a honeycomb, consisting of n rows of m regular hexagons each. Each hexagon is an independent state of Honeyland. Because it currently is very difficult to get from one state to another, the queen bee decided to connect all states by honey streams. It is clear that only neighbouring states can be directly connected by a stream.

Also, honey streams are rivers of pure honey, so bees need lots of honey to create them and it takes great effort. Unfortunately, bees - while famously very industrious - aren't smart enough to solve this task in an efficient way, but you can help them, right?

The amount of honey needed to create the stream between two neighbouring states is calculated by a simple formula: $(x_1x_2 + p_1y_1y_2) \bmod p_2$, where (x_1, y_1) and (x_2, y_2) are the coordinates of the two states and p_1, p_2 are some given prime numbers. The coordinates simply describe the row the state is in and its position in that row, where the rows are numerated from top to down and states in a row from left to right, s.t. the top left state of Honeyland has coordinates $(1, 1)$. You can see a map of the realm with $n = m = 3$ below.



Input

The input contains the four positive integers n, m, p_1 and p_2, each not exceeding 1000, where p_1 and p_2 are guaranteed to be prime numbers.

Output

Output a single integer - the minimum amount of honey required to realize the queen bee's plan to connect all states with honey streams.

Sample input and output

Input	**Output**
3 3 11 23	28

Problem C: Horrible Roads

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

According to a popular saying, nothing good happens late at night. But sometimes nights are so fine nonetheless, that great ideas cross some people's minds. For example, recently a professor of the Ordinary Military Group (OMG) had an excellent idea that he still is very excited about, although it is completely useless to anyone but him:

The professor loves to walk alongside roads, especially on paths with ring roads and crossroads, but is horrified by the sight of DacMonalds, traffic jams and so on. His idea was to characterize each road by its own level of horror, which he derived from their respective abundance of these features.

The road system of the city where he lives is a set of **bidirectional roads** connecting two junctions each. One can reach every junction from any other junction and there are no roads directly connecting any junction to itself. It just so happens to be the case that the level of horror is different for every road in this city.

The professor also decided to call a road "horrible" if there is a cycle (a path without duplicated roads that starts and ends at the same junction) in which that road is the one he assigned the highest level of horror to.

"But wouldn't all roads be horrible...?", the professor pondered. Yet it was too sad a thought, so he wished to not think about it himself. Instead he wants you to find out if there are any non-horrible roads in his city.

Input

The input consists of

- one line containing n and m ($n, m \leq 10^5$) – the numbers of junctions and roads
- m lines describing the roads, with the i-th line containing u_i and v_i – the indices of the junctions linked by the i-th road. The roads are given in ascending order of their level of horror and are numerated from 1 to

m in the same order. It's guaranteed that every junction is reachable from any other junction and that no road links a junction to itself, but there may be several roads between two junctions.

Output

On the first line output a single integer - the amount of non-horrible roads. On the next line output the numbers of those roads in ascending order.

Sample input and output

Input	**Output**

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

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

Problem D: Battleship

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

For decades, your galaxy has been ravaged by a ruthless war between the *Galactic Empire*, a totalitarian regime that used to be in control of the entire galaxy, and the rebels trying to take it down and instate democracy. Now – finally – there may be light at the end of the tunnel for the rebels:

While some of their own ships are hiding nearby, they have managed to get you – their most skilled agent – onto the enormous battleship from whose command center the *Empire* controls its entire fleet (most of which is currently assembled around it). Your mission now is to disrupt said command center for long enough that the nearby rebels can launch a surprise attack on the then-unguided imperial fleet – in the hopes of destroying it entirely, thereby forcing the *Empire* to surrender.

Unfortunately, you are still very much in the periphery of the giant ship, whereas the command center is located in what can only be described as its heart, protected by countless layers of security. To prevent things like those you are intending to do from happening, the ship's security is organized in a strictly hierarchical fashion: the ship essentially consists of small rooms, lengthy hallways and large halls and hangars, all carefully separated by thick metal doors, each of which requires identification by card of someone with a certain authentication level to open.

To be able to advance to the command room, you plan to somehow steal an imperial soldier's identity card. However, as this will probably be harder, the higher-ranking the soldier is (due to higher-ups generally attracting more attention by other soldiers), you first want to figure out exactly what authentication level you would need for there to be a path from your current position to the command room on which you could open all doors.

After careful consideration, you decide that it makes more sense to merely aim for a room adjacent to the command room, since the doors to the room itself all require extremely high authentication levels. Then you could just wait for someone legitimate entering it and slip in with him – a tactic so conspicuous you wouldn't want to try it for a larger number of doors.

Input

The input consists of

- one line containing N ($3 \le N \le 10^5$) – the number of rooms, halls and hallways – and M ($2 \le M \le 10^6$) – the number of doors connecting them
- one line containing s and t ($1 \le s, t \le N$) – the index of the room you are currently in and that of the command center
- M lines describing the doors, with the i-th line containing three integers u_i, v_i and ℓ_i ($1 \le u_i, v_i \le N, 1 \le \ell_i \le 100$) – denoting that the i-th door connects rooms u_i and v_i and requires an authentication level of ℓ_i or higher. All doors can be used the same way in both directions. It is guaranteed that with maximum authentication level one could get from every room to every other room and also that the room you are currently in is not adjacent to the command center.

Output

Output the lowest authentication level for which there is a path from your current position to a room adjacent (connected by a door) to the command center on which said level would suffice to open all doors.

Sample input and output

Input	**Output**
3 2	10
1 3	
1 2 10	
2 3 95	
6 8	20
1 4	
1 2 30	
1 5 10	
2 3 20	
2 5 15	
3 4 90	
3 6 80	
4 6 90	
5 6 70	

Problem E: Union day

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

In Byteland there are n independent cities and no roads at all between them. The King of the realm decided he needed to change this, so he commanded his subjects to build roads – each of which should connect two cities in a straight line – in such a way that afterwards every city is reachable from any other city by those roads.

When the construction is done the King plans to celebrate it as the "Union Day". Unfortunately the realm's treasury is almost empty, so he wants to save as much as possible by minimizing the total length of the roads.

Input

The input consists of

- one line containing n $(1 \leq n \leq 5000)$ – the number of cities
- n lines describing the cities, with the i-th line containing integers x_i and y_i $(-10^4 \leq x_i, y_i \leq 10^4)$ – the coordinates of the i-th city. Obviously no two cities are located in the same spot.

Output

Output the minimal total length L of the roads he needs to build with precision 10^{-6}.

Sample input and output

Input	**Output**
6	9.6568542
1 1	
7 1	
2 2	
6 2	
1 3	
7 3	

Problem F: Fun with Queries

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

You are to answer queries concerning an undirected graph that is built edge by edge. At time zero, there are N vertices and no edges between them. Then, for $i = 1, ..., M$, at time i the i-th edge is added to the graph. The queries you need to answer each consist of two distinct vertices and request the time at which these vertices first become part of the same subgraph (i.e. reachable from one another through a series of already added edges).

Input

The input consists of

- one line containing N, M and Q ($2 \leq N \leq 10^5$, $N - 1 \leq M \leq 3 \cdot 10^5$, $1 \leq Q \leq 10^5$) – the amounts of vertices, edges and queries, respectively
- M lines providing the edges, with the i-th line containing integers u_i and v_i ($1 \leq u_i, v_i \leq N$, $u_i \neq v_i$) – the vertices connected by the i-th edge
- Q lines providing the queries, with the i-th line containing integers u_i and v_i ($1 \leq u_i, v_i \leq N$, $u_i \neq v_i$) – the arguments of the i-th query.

It is guaranteed that – after all edges are added – the graph is connected.

Output

Output one line containing Q numbers, the i-th of which is the answer to the i-th query. In accordance with the input restrictions, all answers should lie between 1 and M (inclusively).

Sample input and output

Input	**Output**

2 1 1
1 2
1 2
	1

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