SGU 185. Two shortest

time limit per test: 0.50 sec.
memory limit per test: 4096 KB
input: standard input
output: standard output

Yesterday Vasya and Petya quarreled badly, and now they don't want to see each other on their way to school. The problem is that they live in one and the same house, leave the house at the same time and go at the same speed by the shortest road. Neither of them wants to change their principles, that is why they want to find two separate shortest routes, which won't make them go along one road, but still they can meet at any junction. They ask you to help them. They number all the junctions with numbers from 1 to N (home and school are also considered as junctions). So their house has the number 1 and the school has the number N, each road connects two junctions exactly, and there cannot be several roads between any two junctions.

Input

The first line contains two integer numbers N and M (2<=N<=400), where M is the number of roads Petya and Vasya noticed. Each of the following M lines contains 3 integers: X, Y and L (1<=X, Y<=N, 1<=L<=10000), where X and Y - numbers of junctions, connected by the road and L is the length of the road.

Output

Write to the first line numbers of the junctions in the way they passed them on the first route. Write to the second line numbers of the junctions in the way they passed them on the second route. If it is impossible to help guys, then output "No solution".

Sample test(s)

Input
```
6 8
1 2 1
3 2 1
3 4 1
1 3 2
4 2 2
4 5 1
5 6 1
4 6 2
```

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

SGU 226. Colored graph

time limit per test: 0.50 sec.
memory limit per test: 4096 KB
input: standard
output: standard

You are given an oriented graph. Each edge of the graph is colored in one of the three colors. Your task is to find the length of the shortest path from the first vertex to the N-th. Note that any two successive edges in the path can't have the same color.

Input
The first line of the input file consists of two integers N and M (2 <= N <= 200; 0 <= M <= N*N). Next M lines contain descriptions of the edges. Each edge description is a list of three integers X, Y, C (1 <= X, Y <= N, 1 <= C <= 3), where X is the starting vertex of the edge, Y is the finishing vertex and C is the color of the edge.

Output
Output the length of the shortest path between the first and the N-th vertexes. Output "-1" if the path doesn't exist.

Sample test(s)

Input

```
Test #1
4 4
1 2 1
2 3 2
3 4 3
2 4 1

Test #2
3 2
1 2 1
2 3 1
```

Output

```
Test #1
3

Test #2
-1
```

SGU 298. King Berl VI

Time limit per test: 4 second(s)
Memory limit: 65536 kilobytes
input: standard
output: standard

King Berl VI has *N* daughters and no sons. During his long life he gave a number of promises to his daughters. All the promises have the following wording: "daughter X_i, I promise to give you dowry not less than C_i burles more than to daughter Y_i", where *i* represents the number of the promise. Before his death the king decided to give some amount of money to each daughter. As far as he was the fair king, he decided to fullfill all his promises. But he was not only fair but also very greedy, he decided that he can give negative amount of burles as a dowry (i.e. daughter should pay this amount of burles to the treasury). Because of his born greed and by advice of the minister of finances, he made a decision that absolute value of each dowry should not exceed 10000 burles and the difference between dowry of the oldest and of the youngest daughters should be as small as possible (note, this value can be negative).

I.e. if the dowry given to the *i*-th daughter is A_i, folllowing conditions should be satisfied:

- $-10000 \leq A_i \leq 10000$
- $A_N - A_1$ should be minimal

Input
The fist line of the input file contains two integers numbers *N* and *M* ($2 \leq N \leq 10000$; $0 \leq M \leq 100000$), where *N* is the number of daughters and *M* is the number of promises. The following *M* lines contain the description of promises in the following form: X_i, Y_i, C_i ($1 \leq X_i, Y_i \leq N; X_i \neq Y_i; 0 \leq C_i \leq 1000$). The youngest daughter has the number one, the oldest — *N*. Each pair X_i, Y_i can appear in the input several times.

Output
Write to the output number -1 if there is no solution for the problem (i.e. there is no sequence of *N* integers which satisfies all described above requirements). Write to the output *N* integer numbers — the amount of dowry of each daughter in burles, if solution exists. If there are several solutions output any of them.

Example(s)

sample input	sample output

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

sample input	sample output
2 2	-7 -7
1 2 0	
2 1 0	

XI Olimpiad in Informatics 2003/2004

Task: WSC

East-West

III stage competition

Source file: `wsc.*`

Memory limit: 32 MB

International agreements signed by the Awfully Vast State impose on it transit obligations - it has to enable the transport of nuclear waste from its eastern neighbours power plants to recycling facilities in the West, by means of its railway system. Ecological reasons impose such traffic organization, that the waste-carrying trains leave the territory of the state as quickly as possible. The railway network in this country has a very peculiar structure. It consists of *n* cities - railway junctions and *n-1* two-way railway track segments, connecting the junctions. Transport is possible between each pair of cities. Furthermore, there is a section of the railway track, whose ends are not border cities and every connection from the eastern to the western border has to lead through that very section.
All waste-carrying trains arrive at the eastern border on the same day, before dawn, at distinct checkpoints, however. For safety reasons the trains only move during the day. Only a single waste-carrying train can move on a given track section at a time, but an unlimited amount of them can wait at a junction. It takes one day for a train to traverse a section of the track. The traffic has to be organized in such a way, that allows each waste-carrying train to reach a distinct destination on the western border.
How many days, at least, do the waste-carrying trains have to spend on the territory of the Awfully Vast State?

Task

Your task is to write a programme which:

- reads from the standard input a description of the railway network and border checkpoints at which the waste-carrying trains have arrived,
- finds the minimal number of days the transit has to last,
- writes the solution to the standard output.

Input

The first line of the input contains three integers $1 < n, w, z < 10^6$, $n >= w+z+2$, separated by single spaces. *n* denotes the number of junctions (which are labelled with *1,...,n*), while *w* and *z* denote the number of border checkpoints on the eastern and western border, respectively. The

checkpoints on the eastern border are labelled with *1,...,w*, while those on the western border with *n-z+1,...,n*.
In the following *n-1* lines there is a decription of the railway network. Each line contains two distinct integers, *1 < a,b < n*, separated by a single space. They denote junctions connected by a section of the railway.
The *n+1*st line contains a single integer *p*, *1 < p < w*, *1 < p < z*, denoting the number of waste-carrying trains. In the next (and last) line of the input there are *p* distinct integers, all of which are not greater than *w*, separated by single spaces. These are the numbers of checkpoints on the eastern border, at which the waste-carrying trains have arrived.

Output

The first and only line of the output should contain exactly one integer, denoting the minimal amount of days the waste have to spend on the territory of the state.

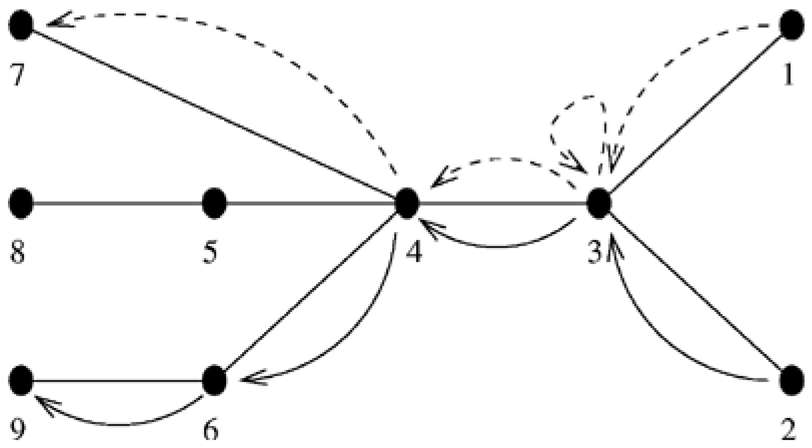
Example

For the input data:

```
9 2 3
1 3
2 3
4 3
4 5
4 6
7 4
5 8
9 6
2
1 2
```

the correct outcome is:

```
4
```



The arrows denote the movement of the trains in consecutive days for one of the optimal organizations of the railway traffic - a loop means that on a given day the train waited at the junction.

BalticOI 2008 Mafia

Memory limit: 32 MB

The police in Byteland got an anonymous tip that the local mafia bosses are planning a big transport from the harbour to one of the secret warehouses in the countryside. The police knows the date of the transport and they know that the transport will use the national highway network. The highway network consists of two-way highway segments, each segment directly connecting two distinct toll stations. A toll station may be connected with many other stations. A vehicle can enter or exit the highway network at toll stations only. The mafia transport is known to enter the highways at the toll station closest to the harbour and leave it at the toll station closest to the warehouse (it will not leave and re-enter the highways in between). Special police squads are to be located in selected toll stations. When the transport enters a toll station under surveillance, it will be caught by the police.
From this point of view, the easiest choice would be to place the police squad either at the entry point or the exit point of the transport. However, controlling each toll station has a certain cost, which may vary from station to station. The police wants to keep the overall cost as low as possible, so they need to identify a *minimal controlling set* of toll stations, which satisfies the two conditions:

- all traffic from the harbour to the warehouse must pass through at least one station from that set,
- the cost of monitoring these stations (i.e. the sum of their individual monitoring costs) is minimal.

You may assume that it is possible to get from the harbour to the warehouse using the highways.

Task

Write a program, that:

- reads the description of the highway network, the monitoring costs and the locations of the entry and exit points of the transport from the standard input,
- finds a minimal controlling set of toll stations,
- writes this set to the standard output.

Input

The first line of the standard input contains two integers $n$and m($2 \leq n \leq 200$, $1 \leq m \leq 20000$) - the number of toll stations and the number of direct highway segments. The toll stations are numbered from $1$to n.
The second line contains two integers $a$and b($1 \leq a, b \leq n$, $a \neq b$) - the numbers of the toll stations closest to the harbour and to the warehouse, respectively.
The following $n$lines describe the monitoring costs. The i-th of these lines (for $1 \leq i \leq n$) contains one integer - the monitoring cost of the i-th station (which is positive number not exceeding $10\,000\,000$).

The following m lines describe the highway network. The j-th of these lines (for $1 \leq j \leq m$) contains two integers x and y ($1 \leq x < y \leq n$), indicating that there is a direct highway segment between toll stations numbered x and y. Each highway segment is listed once.
Additionally, in test cases worth about 40 points, $n \leq 20$.

Output

The only line of the output should contain the numbers of toll stations in a minimal controlling set, given in increasing order, separated by single spaces. If there is more than one minimal controlling set, your program may output anyone of them.

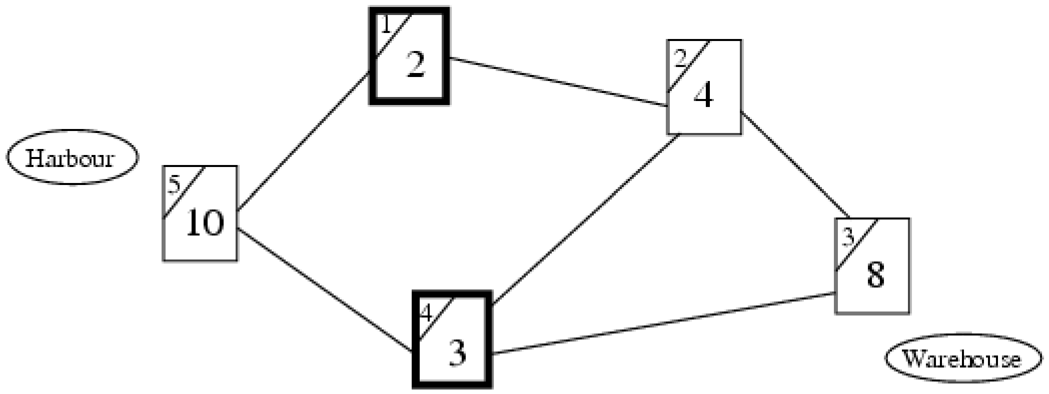
Example

For the input data:

```
5 6
5 3
2
4
8
3
10
1 5
1 2
2 4
4 5
2 3
3 4
```

the correct result is:

```
1 4
```



The figure shows the highway network with the toll station numbers (in the upper-left corners) and the monitoring costs. Stations number 1 and 4 constitute the minimal controlling set with total controlling cost 5.

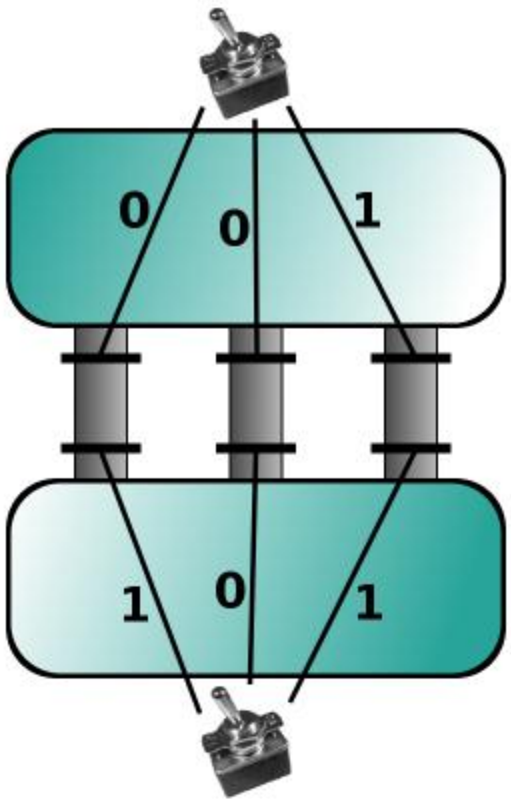
BalticOI 2008 Day 1 Gates

Memory limit: 96 MB

After many years of working as a software developer you have decided to try something entirely different, and started looking at random job offers. The one that really caught your eye was a job in fish farming (a form of aquaculture). 'Cool!', you thought, and besides, fish are nice creatures. So you applied, got accepted, and today is your first day at work.

Your boss has already assigned you a task. You have to isolate one water reservoir from another. After looking at some schemes you've been given, here's what you've figured out.

The two water reservoirs are connected by several channels. Each channel has two gates. The channel is open when both gates are open, and is closed otherwise. The gates are controlled using switches. The same switch may operate several gates, but each gate is operated by exactly one switch. It is possible that both gates on a channel are controlled by the same switch and that a switch controls no gates.



Example with three channels and two switches.

The switch may operate the gate in one of two ways:

- the gate is open when the switch is on, and is closed when the switch is off,
- the gate is closed when the switch is on, and is open when the switch is off.

After playing a bit with the switches you suddenly realize that your programming experience will come in very handy. Write a program that, given the configuration of gates and switches, determines whether it is possible to close all channels, and if it is, then finds a state of every switch in one such valid configuration.

Input

The first line of the standard input contains two integers n $(1 \leq n \leq 250\,000)$ and m $(1 \leq m \leq 500\,000)$, the number of channels and switches respectively. Switches are numbered from 1 to m. Additionally, in test cases worth at least 30% points, n will not exceed 10 and m will not exceed 20.
The following n lines describe channels, each channel is described by a separate line containing four integers: a, s_a, b, s_b. Numbers a and b represent switches ($1 \leq a, b \leq m$) that operate gates of this channel. Numbers s_a and s_b can be either 0 or 1 and correspond to the described operation modes: $s_i = 0$ means that the gate is closed if and only if the switch i is off and $s_i = 1$ means that the gate is closed if and only if the switch i is on.

Output

If it is possible to close all the channels, the standard output should contain m lines. Line i should contain 0, if switch i should be off, and 1 if switch i should be on. If there are many possible solutions, your program may output any of them.
If it is impossible to close all channels, your program should output one line, containing a single word `IMPOSSIBLE`.

Example

For the input data:
```
3 2
1 0 2 1
1 0 2 0
1 1 2 1
```
the correct result is:
```
0
1
```
and for the input data:
```
2 1
1 0 1 0
1 1 1 1
```
the correct result is:
```
IMPOSSIBLE
```
The first example corresponds to the picture from the task description.

SGU 381. Bidirected Graph

Time limit per test: 4 second(s)
Memory limit: 262144 kilobytes
input: standard
output: standard

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 ($V, E, ends, \omega$). V is a set of nodes, E is a set of edges, *ends* is a mapping from E: $ends(e) = \{u, v\}$, where $u, v \in V$, $u \neq v$. For every $e \in E$ and $v \in ends(e)$ 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 $v_i \in V$, $e_i \in E$, $ends(e_i) = \{v_{i-1}, v_i\}$ 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 $ends(e_a) = \{u, v\}$, $\omega(e_a, u) = -1$, $\omega(e_a, v) = +1$.

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 ($1 \leq n \leq 100\,000$, $0 \leq m \leq 500\,000$). 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 $d_{i,1}=\omega(e,a_i)$, $d_{i,2}=\omega(e,b_i)$; $1 \leq a_i, b_i \leq n$, $d_{i,j} \in \{-1, 1\}$.

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.

Example(s)

sample input	sample output

4 3
1 3 -1 -1
2 3 -1 -1
3 4 1 1
	YES
	1
	3

sample input	sample output

3 3
1 2 1 1
2 3 1 1
1 3 1 1
	NO