GPC-UPC TopoSort, Bridges, Articulation points and SCC Contest

A. Tourist Reform

4 seconds, 256 megabytes

Berland is a tourist country! At least, it can become such — the government of Berland is confident about this.

There are n cities in Berland, some pairs of which are connected by two-ways roads. Each road connects two different cities. In Berland there are no roads which connect the same pair of cities. It is possible to get from any city to any other city using given two-ways roads.

According to the reform each road will become one-way. It will be oriented to one of two directions.

To maximize the tourist attraction of Berland, after the reform for each city i the value r_i will be calculated. It will equal to the number of cities x for which there is an oriented path from the city i to the city x. In other words, r_i will equal the number of cities which can be reached from the city i by roads.

The government is sure that tourist's attention will be focused on the minimum value of r_i.

Help the government of Berland make the reform to maximize the minimum of r_i.

Input

The first line contains two integers n, m ($2 \le n \le 400\,000$, $1 \le m \le 400\,000$) — the number of cities and the number of roads.

The next m lines describe roads in Berland: the j-th of them contains two integers u_j and v_j ($1 \le u_j, v_j \le n$, $u_j \ne v_j$), where u_j and v_j are the numbers of cities which are connected by the j-th road.

The cities are numbered from 1 to n. It is guaranteed that it is possible to get from any city to any other by following two-ways roads. In Berland there are no roads which connect the same pair of cities.

Output

In the first line print single integer — the maximum possible value $min_{1 \le i \le n}\{r_i\}$ after the orientation of roads.

The next m lines must contain the description of roads after the orientation: the j-th of them must contain two integers u_j, v_j, it means that the j-th road will be directed from the city u_j to the city v_j. Print roads in the same order as they are given in the input data.

input

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

output

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

B. Cutting Figure

2 seconds, 256 megabytes

You've gotten an $n \times m$ sheet of squared paper. Some of its squares are painted. Let's mark the set of all painted squares as A. Set A is connected. Your task is to find the minimum number of squares that we can delete from set A to make it not connected.

A set of painted squares is called *connected*, if for every two squares a and b from this set there is a sequence of squares from the set, beginning in a and ending in b, such that in this sequence any square, except for the last one, shares a common side with the square that follows next in the sequence. An empty set and a set consisting of exactly one square are connected by definition.

Input

The first input line contains two space-separated integers n and m ($1 \le n, m \le 50$) — the sizes of the sheet of paper.

Each of the next n lines contains m characters — the description of the sheet of paper: the j-th character of the i-th line equals either "#", if the corresponding square is painted (belongs to set A), or equals "." if the corresponding square is not painted (does not belong to set A). It is guaranteed that the set of all painted squares A is connected and isn't empty.

Output

On the first line print the minimum number of squares that need to be deleted to make set A not connected. If it is impossible, print -1.

input

```
5 4
####
#..#
#..#
#..#
####
```

output

```
2
```

input

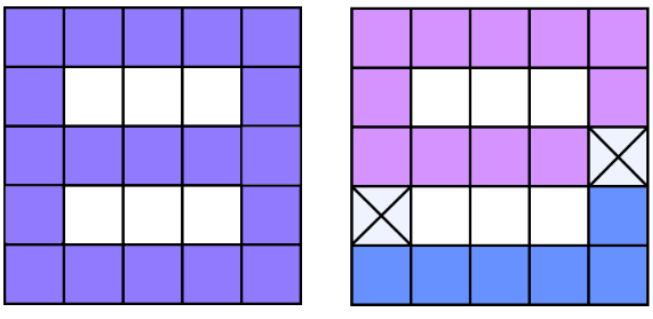
```
5 5
#####
#...#
#####
#...#
#####
```

output

```
2
```

In the first sample you can delete any two squares that do not share a side. After that the set of painted squares is not connected anymore.

The note to the second sample is shown on the figure below. To the left there is a picture of the initial set of squares. To the right there is a set with deleted squares. The deleted squares are marked with crosses.



C. Break Up

3 seconds, 256 megabytes

Again, there are hard times in Berland! Many towns have such tensions that even civil war is possible.

There are n towns in Reberland, some pairs of which connected by two-way roads. It is not guaranteed that it is possible to reach one town from any other town using these roads.

Towns s and t announce the final break of any relationship and intend to rule out the possibility of moving between them by the roads. Now possibly it is needed to close several roads so that moving from s to t using roads becomes impossible. Each town agrees to spend money on closing no more than one road, therefore, the total number of closed roads will be **no more than two**.

Help them find set of no more than two roads such that there will be no way between s and t after closing these roads. For each road the budget required for its closure was estimated. Among all sets find such that the total budget for the closure of a set of roads is minimum.

Input

The first line of the input contains two integers n and m ($2 \le n \le 1000$, $0 \le m \le 30\,000$) — the number of towns in Berland and the number of roads.

The second line contains integers s and t ($1 \le s, t \le n$, $s \ne t$) — indices of towns which break up the relationships.

Then follow m lines, each of them contains three integers x_i, y_i and w_i ($1 \le x_i, y_i \le n$, $1 \le w_i \le 10^9$) — indices of towns connected by the i-th road, and the budget on its closure.

All roads are bidirectional. It is allowed that the pair of towns is connected by more than one road. Roads that connect the city to itself are allowed.

Output

In the first line print the minimum budget required to break up the relations between s and t, if it is allowed to close no more than two roads.

In the second line print the value c ($0 \le c \le 2$) — the number of roads to be closed in the found solution.

In the third line print in any order c diverse integers from 1 to m — indices of closed roads. Consider that the roads are numbered from 1 to m in the order they appear in the input.

If it is impossible to make towns s and t disconnected by removing no more than 2 roads, the output should contain a single line `-1`.

If there are several possible answers, you may print any of them.

input

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

output

```
8
2
2 7
```

input

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

output

```
9
2
4 5
```

input

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

output

```
1
1
2
```

input

```
2 3
1 2
1 2 734458840
1 2 817380027
1 2 304764803
```

output

```
-1
```

D. We Need More Bosses

2 seconds, 256 megabytes

Your friend is developing a computer game. He has already decided how the game world should look like — it should consist of n locations connected by m **two-way** passages. The passages are designed in such a way that it should be possible to get from any location to any other location.

Of course, some passages should be guarded by the monsters (if you just can go everywhere without any difficulties, then it's not fun, right?). Some crucial passages will be guarded by really fearsome monsters, requiring the hero to prepare for battle and designing his own tactics of defeating them (commonly these kinds of monsters are called **bosses**). And your friend wants you to help him place these bosses.

The game will start in location s and end in location t, but these locations are not chosen yet. After choosing these locations, your friend will place a boss in each passage such that it is impossible to get from s to t without using this passage. Your friend wants to place as much bosses as possible (because more challenges means more fun, right?), so he asks you to help him determine the maximum possible number of bosses, considering that any location can be chosen as s or as t.

Input

The first line contains two integers n and m ($2 \le n \le 3 \cdot 10^5$, $n - 1 \le m \le 3 \cdot 10^5$) — the number of locations and passages, respectively.

Then m lines follow, each containing two integers x and y ($1 \le x, y \le n$, $x \ne y$) describing the endpoints of one of the passages.

It is guaranteed that there is no pair of locations directly connected by two or more passages, and that any location is reachable from any other location.

Output

Print one integer — the maximum number of bosses your friend can place, considering all possible choices for s and t.

input

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

output

```
2
```

input

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

output

```
3
```

E. Tourism

2 seconds, 256 megabytes

Alex decided to go on a touristic trip over the country.

For simplicity let's assume that the country has n cities and m bidirectional roads connecting them. Alex lives in city s and initially located in it. To compare different cities Alex assigned each city a score w_i which is as high as interesting city seems to Alex.

Alex believes that his trip will be interesting only if he will not use any road twice in a row. That is if Alex came to city v from city u, he may choose as the next city in the trip any city connected with v by the road, except for the city u.

Your task is to help Alex plan his city in a way that maximizes total score over all cities he visited. Note that for each city its score is counted at most once, even if Alex been there several times during his trip.

Input

First line of input contains two integers n and m, ($1 \le n \le 2 \cdot 10^5$, $0 \le m \le 2 \cdot 10^5$) which are numbers of cities and roads in the country.

Second line contains n integers $w_1, w_2, \ldots, w_n$ ($0 \le w_i \le 10^9$) which are scores of all cities.

The following m lines contain description of the roads. Each of these m lines contains two integers u and v ($1 \le u, v \le n$) which are cities connected by this road.

It is guaranteed that there is at most one direct road between any two cities, no city is connected to itself by the road and, finally, it is possible to go from any city to any other one using only roads.

The last line contains single integer s ($1 \le s \le n$), which is the number of the initial city.

Output

Output single integer which is the maximum possible sum of scores of visited cities.

input

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

output

```
27
```

input

```
10 12
1 7 1 9 3 3 6 30 1 10
1 2
1 3
3 5
5 7
2 3
5 4
6 9
4 6
3 7
6 8
9 4
9 10
6
```

output

```
61
```

F. Ralph and Mushrooms

2.5 seconds, 512 megabytes

Ralph is going to collect mushrooms in the Mushroom Forest.

There are m directed paths connecting n trees in the Mushroom Forest. On each path grow some mushrooms. When Ralph passes a path, he collects all the mushrooms on the path. The Mushroom Forest has a magical fertile ground where mushrooms grow at a fantastic speed. New mushrooms regrow as soon as Ralph finishes mushroom collection on a path. More specifically, after Ralph passes a path the i-th time, there regrow i mushrooms less than there was before this pass. That is, if there is initially x mushrooms on a path, then Ralph will collect x mushrooms for the first time, x - 1 mushrooms the second time, x - 1 - 2 mushrooms the third time, and so on. However, the number of mushrooms can never be less than 0.

For example, let there be 9 mushrooms on a path initially. The number of mushrooms that can be collected from the path is 9, 8, 6 and 3 when Ralph passes by from first to fourth time. From the fifth time and later Ralph can't collect any mushrooms from the path (but still can pass it).

Ralph decided to start from the tree s. How many mushrooms can he collect using only described paths?

Input

The first line contains two integers n and m ($1 \le n \le 10^6$, $0 \le m \le 10^6$), representing the number of trees and the number of directed paths in the Mushroom Forest, respectively.

Each of the following m lines contains three integers x, y and w ($1 \le x, y \le n$, $0 \le w \le 10^8$), denoting a path that leads from tree x to tree y with w mushrooms initially. There can be paths that lead from a tree to itself, and multiple paths between the same pair of trees.

The last line contains a single integer s ($1 \le s \le n$) — the starting position of Ralph.

Output

Print an integer denoting the maximum number of the mushrooms Ralph can collect during his route.

input

```
2 2
1 2 4
2 1 4
1
```

output

```
16
```

input

```
3 3
1 2 4
2 3 3
1 3 8
1
```

output
8

In the first sample Ralph can pass three times on the circle and collect $4 + 4 + 3 + 3 + 1 + 1 = 16$ mushrooms. After that there will be no mushrooms for Ralph to collect.

In the second sample, Ralph can go to tree 3 and collect 8 mushrooms on the path from tree 1 to tree 3.

G. Scheme

2 seconds, 256 megabytes

To learn as soon as possible the latest news about their favourite fundamentally new operating system, BolgenOS community from Nizhni Tagil decided to develop a scheme. According to this scheme a community member, who is the first to learn the news, calls some other member, the latter, in his turn, calls some third member, and so on; i.e. a person with index i got a person with index f_i, to whom he has to call, if he learns the news. With time BolgenOS community members understood that their scheme doesn't work sometimes — there were cases when some members didn't learn the news at all. Now they want to supplement the scheme: they add into the scheme some instructions of type (x_i, y_i), which mean that person x_i has to call person y_i as well. What is the minimum amount of instructions that they need to add so, that at the end everyone learns the news, no matter who is the first to learn it?

Input

The first input line contains number n ($2 \le n \le 10^5$) — amount of BolgenOS community members. The second line contains n space-separated integer numbers f_i ($1 \le f_i \le n, i \ne f_i$) — index of a person, to whom calls a person with index i.

Output

In the first line output one number — the minimum amount of instructions to add. Then output one of the possible variants to add these instructions into the scheme, one instruction in each line. If the solution is not unique, output any.

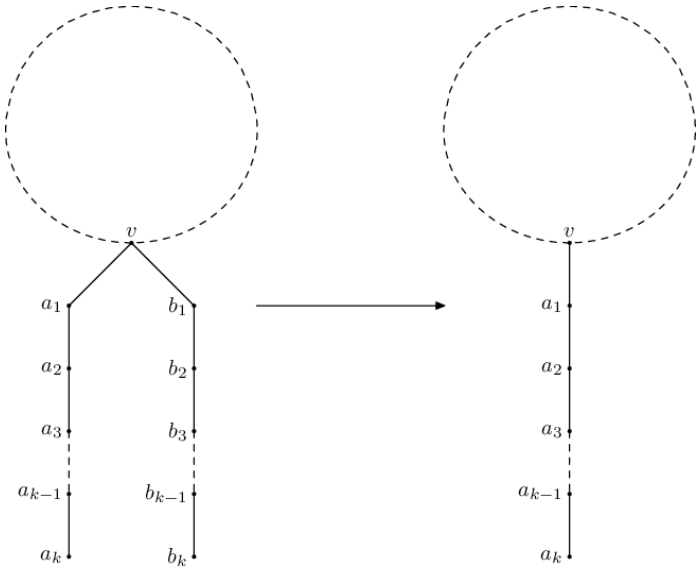
input
3 3 3 2
output
1 3 1

input
7 2 3 1 3 4 4 1
output
3 2 5 2 6 3 7

H. Tree Folding

2 seconds, 512 megabytes

Vanya wants to minimize a tree. He can perform the following operation multiple times: choose a vertex v, and two disjoint (except for v) paths of equal length $a_0 = v, a_1, ..., a_k$, and $b_0 = v, b_1, ..., b_k$. Additionally, vertices $a_1, ..., a_k, b_1, ..., b_k$ must not have any neighbours in the tree other than adjacent vertices of corresponding paths. After that, one of the paths may be merged into the other, that is, the vertices $b_1, ..., b_k$ can be effectively erased:



Help Vanya determine if it possible to make the tree into a path via a sequence of described operations, and if the answer is positive, also determine the shortest length of such path.

Input

The first line of input contains the number of vertices n ($2 \le n \le 2 \cdot 10^5$).

Next $n - 1$ lines describe edges of the tree. Each of these lines contains two space-separated integers u and v ($1 \le u, v \le n, u \ne v$) — indices of endpoints of the corresponding edge. It is guaranteed that the given graph is a tree.

Output

If it is impossible to obtain a path, print `-1`. Otherwise, print the minimum number of edges in a possible path.

input
6 1 2 2 3 2 4 4 5 1 6
output
3

input
7 1 2 1 3 3 4 1 5 5 6 6 7
output
-1

In the first sample case, a path of three edges is obtained after merging paths $2 - 1 - 6$ and $2 - 4 - 5$.

It is impossible to perform any operation in the second sample case. For example, it is impossible to merge paths $1 - 3 - 4$ and $1 - 5 - 6$, since vertex 6 additionally has a neighbour 7 that is not present in the corresponding path.

I. Minimal Labels

1 second, 256 megabytes

You are given a directed acyclic graph with n vertices and m edges. There are no self-loops or multiple edges between any pair of vertices. Graph can be disconnected.

You should assign labels to all vertices in such a way that:

- Labels form a valid permutation of length n — an integer sequence such that each integer from 1 to n appears exactly once in it.
- If there exists an edge from vertex v to vertex u then $label_v$ should be smaller than $label_u$.
- Permutation should be lexicographically smallest among all suitable.

Find such sequence of labels to satisfy all the conditions.

Input
The first line contains two integer numbers n, m ($2 \le n \le 10^5$, $1 \le m \le 10^5$).

Next m lines contain two integer numbers v and u ($1 \le v, u \le n, v \ne u$) — edges of the graph. Edges are directed, graph doesn't contain loops or multiple edges.

Output
Print n numbers — lexicographically smallest correct permutation of labels of vertices.

input
3 3 1 2 1 3 3 2
output
1 3 2

input
4 5 3 1 4 1 2 3 3 4 2 4
output
4 1 2 3

input
5 4 3 1 2 1 2 3 4 5
output
3 1 2 4 5

J. Coins

2 seconds, 256 megabytes

One day Vasya came across three Berland coins. They didn't have any numbers that's why Vasya didn't understand how their denominations differ. He supposed that if one coin is heavier than the other one, then it should be worth more. Vasya weighed all the three pairs of coins on pan balance scales and told you the results. Find out how the deminations of the coins differ or if Vasya has a mistake in the weighting results. No two coins are equal.

Input
The input data contains the results of all the weighting, one result on each line. It is guaranteed that every coin pair was weighted exactly once. Vasya labelled the coins with letters «A», «B» and «C». Each result is a line that appears as (letter)(> or < sign)(letter). For example, if coin "A" proved lighter than coin "B", the result of the weighting is A<B.

Output
It the results are contradictory, print `Impossible`. Otherwise, print without spaces the rearrangement of letters «A», «B» and «C» which represent the coins in the increasing order of their weights.

input
A>B C<B A>C
output
CBA

input
A<B B>C C>A
output
ACB

K. Coprocessor

1.5 seconds, 256 megabytes

You are given a program you want to execute as a set of tasks organized in a dependency graph. The dependency graph is a directed acyclic graph: each task can depend on results of one or several other tasks, and there are no directed circular dependencies between tasks. A task can only be executed if all tasks it depends on have already completed.

Some of the tasks in the graph can only be executed on a coprocessor, and the rest can only be executed on the main processor. In one coprocessor call you can send it a set of tasks which can only be executed on it. For each task of the set, all tasks on which it depends must be either already completed or be included in the set. The main processor starts the program execution and gets the results of tasks executed on the coprocessor automatically.

Find the minimal number of coprocessor calls which are necessary to execute the given program.

Input
The first line contains two space-separated integers N ($1 \le N \le 10^5$) — the total number of tasks given, and M ($0 \le M \le 10^5$) — the total number of dependencies between tasks.

The next line contains N space-separated integers $E_i \in \{0, 1\}$. If $E_i = 0$, task i can only be executed on the main processor, otherwise it can only be executed on the coprocessor.

The next M lines describe the dependencies between tasks. Each line contains two space-separated integers T_1 and T_2 and means that task T_1 depends on task T_2 ($T_1 \ne T_2$). Tasks are indexed from 0 to $N - 1$. All M pairs (T_1, T_2) are distinct. It is guaranteed that there are no circular dependencies between tasks.

Output
Output one line containing an integer — the minimal number of coprocessor calls necessary to execute the program.

input
4 3 0 1 0 1 0 1 1 2 2 3
output
2

input
4 3 1 1 1 0 0 1 0 2 3 0
output
1

In the first test, tasks 1 and 3 can only be executed on the coprocessor. The dependency graph is linear, so the tasks must be executed in order 3 -> 2 -> 1 -> 0. You have to call coprocessor twice: first you call it for task 3, then you execute task 2 on the main processor, then you call it for for task 1, and finally you execute task 0 on the main processor.

In the second test, tasks 0, 1 and 2 can only be executed on the coprocessor. Tasks 1 and 2 have no dependencies, and task 0 depends on tasks 1 and 2, so all three tasks 0, 1 and 2 can be sent in one coprocessor call. After that task 3 is executed on the main processor.

L. Bitwise Formula

3 seconds, 512 megabytes

Bob recently read about bitwise operations used in computers: AND, OR and XOR. He have studied their properties and invented a new game.

Initially, Bob chooses integer m, bit depth of the game, which means that all numbers in the game will consist of m bits. Then he asks Peter to choose some m-bit number. After that, Bob computes the values of n variables. Each variable is assigned either a constant m-bit number or result of bitwise operation. Operands of the operation may be either variables defined before, or the number, chosen by Peter. After that, Peter's score equals to the sum of all variable values.

Bob wants to know, what number Peter needs to choose to get the minimum possible score, and what number he needs to choose to get the maximum possible score. In both cases, if there are several ways to get the same score, find the minimum number, which he can choose.

Input

The first line contains two integers n and m, the number of variables and bit depth, respectively ($1 \le n \le 5000$; $1 \le m \le 1000$).

The following n lines contain descriptions of the variables. Each line describes exactly one variable. Description has the following format: name of a new variable, space, sign ":=", space, followed by one of:

1. Binary number of exactly m bits.
2. The first operand, space, bitwise operation ("AND", "OR" or "XOR"), space, the second operand. Each operand is either the name of variable defined before or symbol '?', indicating the number chosen by Peter.

Variable names are strings consisting of lowercase Latin letters with length at most 10. All variable names are different.

Output

In the first line output the minimum number that should be chosen by Peter, to make the sum of all variable values minimum possible, in the second line output the minimum number that should be chosen by Peter, to make the sum of all variable values maximum possible. Both numbers should be printed as m-bit binary numbers.

input

```
3 3
a := 101
b := 011
c := ? XOR b
```

output

```
011
100
```

input

```
5 1
a := 1
bb := 0
cx := ? OR a
d := ? XOR ?
e := d AND bb
```

output

```
0
0
```

In the first sample if Peter chooses a number 011_2, then $a = 101_2$, $b = 011_2$, $c = 000_2$, the sum of their values is 8. If he chooses the number 100_2, then $a = 101_2$, $b = 011_2$, $c = 111_2$, the sum of their values is 15.

For the second test, the minimum and maximum sum of variables a, bb, cx, d and e is 2, and this sum doesn't depend on the number chosen by Peter, so the minimum Peter can choose is 0.

M. The Light Square

2 seconds, 256 megabytes

For her birthday Alice received an interesting gift from her friends – The Light Square. The Light Square game is played on an $N \times N$ lightbulbs square board with a magical lightbulb bar of size $N \times 1$ that has magical properties. At the start of the game some lights on the square board and magical bar are turned on. The goal of the game is to transform the starting light square board pattern into some other pattern using the magical bar without rotating the square board. The magical bar works as follows:

It can be placed on any row or column

The orientation of the magical lightbulb must be left to right or top to bottom for it to keep its magical properties

The entire bar needs to be fully placed on a board

The lights of the magical bar never change

If the light on the magical bar is the same as the light of the square it is placed on it will switch the light on the square board off, otherwise it will switch the light on

The magical bar can be used an infinite number of times

Alice has a hard time transforming her square board into the pattern Bob gave her. Can you help her transform the board or let her know it is impossible? If there are multiple solutions print any.

Input

The first line contains one positive integer number N $(1 \le N \le 2000)$ representing the size of the square board.

The next N lines are strings of length N consisting of 1's and 0's representing the initial state of the square board starting from the top row. If the character in a string is 1 it means the light is turned on, otherwise it is off.

The next N lines are strings of length N consisting of 1's and 0's representing the desired state of the square board starting from the top row that was given to Alice by Bob.

The last line is one string of length N consisting of 1's and 0's representing the pattern of the magical bar in a left to right order.

Output

Transform the instructions for Alice in order to transform the square board into the pattern Bob gave her. The first line of the output contains an integer number M $(0 \le M \le 10^5)$ representing the number of times Alice will need to apply the magical bar.

The next M lines are of the form "col X" or "row X", where X is 0-based index of the matrix, meaning the magical bar should be applied to either row X or column X. If there is no solution, print only -1. In case of multiple solutions print any correct one.

input

```
2
11
11
00
01
11
```

output

```
-1
```

input

```
2
10
00
00
00
10
```

output

```
1
row 0
```

input

```
3
110
011
100
100
011
100
100
```

output

```
3
row 0
col 0
col 1
```

Example 1: It is impossible to transform square board from one format to another

Example 2: Magic bar can be applied on first row or column.