Problem Set 6 Shortest Paths 15-295 Fall 2019

A. Valid BFS?

time limit per test: 2 seconds
memory limit per test: 256 megabytes

The BFS algorithm is defined as follows.

1. Consider an undirected graph with vertices numbered from 1 to n. Initialize q as a new queue containing only vertex 1, mark the vertex 1 as used.
2. Extract a vertex v from the head of the queue q.
3. Print the index of vertex v.
4. Iterate in arbitrary order through all such vertices u that u is a neighbor of v and is not marked yet as used. Mark the vertex u as used and insert it into the tail of the queue q.
5. If the queue is not empty, continue from step 2.
6. Otherwise finish.

Since the order of choosing neighbors of each vertex can vary, it turns out that there may be multiple sequences which BFS can print.

In this problem you need to check whether a given sequence corresponds to some valid BFS traversal of the given tree **starting from vertex** 1. The tree is an undirected graph, such that there is exactly one simple path between any two vertices.

Input

The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$) which denotes the number of nodes in the tree.

The following $n - 1$ lines describe the edges of the tree. Each of them contains two integers x and y ($1 \le x, y \le n$) — the endpoints of the corresponding edge of the tree. It is guaranteed that the given graph is a tree.

The last line contains n distinct integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the sequence to check.

Output

Print "Yes" (quotes for clarity) if the sequence corresponds to some valid BFS traversal of the given tree and "No" (quotes for clarity) otherwise.

You can print each letter in any case (upper or lower).

Examples

input	output
4 1 2 1 3 2 4 1 2 3 4	Yes

input	output
4 1 2 1 3 2 4 1 2 4 3	No

Note

Both sample tests have the same tree in them.

In this tree, there are two valid BFS orderings:

- $1, 2, 3, 4,$
- $1, 3, 2, 4.$

The ordering $1, 2, 4, 3$ doesn't correspond to any valid BFS order.

B. Anna, Svyatoslav and Maps

time limit per test: 2 seconds
memory limit per test: 256 megabytes

The main characters have been omitted to be short.

You are given a directed unweighted graph without loops with n vertexes and a path in it (that path is not necessary simple) given by a sequence $p_1, p_2, \ldots, p_m$ of m vertexes; for each $1 \le i < m$ there is an arc from p_i to p_{i+1}.

Define the sequence $v_1, v_2, \ldots, v_k$ of k vertexes as *good*, if v is a subsequence of p, $v_1 = p_1$, $v_k = p_m$, and p is one of the shortest paths passing through the vertexes $v_1, \ldots, v_k$ in that order.

A sequence a is a subsequence of a sequence b if a can be obtained from b by deletion of several (possibly, zero or all) elements. It is obvious that the sequence p is good but your task is to find the **shortest** good subsequence.

If there are multiple shortest good subsequences, output any of them.

Input

The first line contains a single integer n ($2 \le n \le 100$) — the number of vertexes in a graph.

The next n lines define the graph by an adjacency matrix: the j-th character in the i-st line is equal to 1 if there is an arc from vertex i to the vertex j else it is equal to 0. It is guaranteed that the graph doesn't contain loops.

The next line contains a single integer m ($2 \le m \le 10^6$) — the number of vertexes in the path.

The next line contains m integers $p_1, p_2, \ldots, p_m$ ($1 \le p_i \le n$) — the sequence of vertexes in the path. It is guaranteed that for any $1 \le i < m$ there is an arc from p_i to p_{i+1}.

Output

In the first line output a single integer k ($2 \le k \le m$) — the length of the shortest good subsequence. In the second line output k integers $v_1, \ldots, v_k$ ($1 \le v_i \le n$) — the vertexes in the subsequence. If there are multiple shortest subsequences, print any. Any two consecutive numbers should be distinct.

Examples

input	output
4 0110 0010 0001 1000 4 1 2 3 4	3 1 2 4

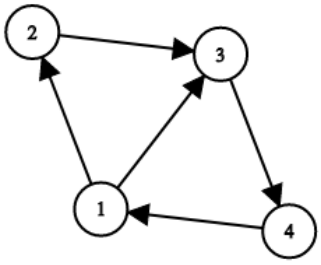
input	output
4 0110 0010 1001 1000 20 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4	11 1 2 4 2 4 2 4 2 4 2 4

input	output
3 011 101 110 7 1 2 3 1 3 2 1	7 1 2 3 1 3 2 1

input	output
4 0110 0001 0001 1000 3 1 2 4	2 1 4

Note

Below you can see the graph from the first example:



The given path is passing through vertexes $1, 2, 3, 4$. The sequence $1 - 2 - 4$ is good because it is the subsequence of the given path, its first and the last elements are equal to the first and the last elements of the given path respectively, and the shortest path passing through vertexes 1, 2 and 4 in that order is $1 - 2 - 3 - 4$. Note that subsequences $1 - 4$ and $1 - 3 - 4$ aren't good because in both cases the shortest path passing through the vertexes of these sequences is $1 - 3 - 4$.

In the third example, the graph is full so any sequence of vertexes in which any two consecutive elements are distinct defines a path consisting of the same number of vertexes.

In the fourth example, the paths $1 - 2 - 4$ and $1 - 3 - 4$ are the shortest paths passing through the vertexes 1 and 4.

C. Edge Deletion

time limit per test: 2.5 seconds
memory limit per test: 256 megabytes

You are given an undirected connected weighted graph consisting of n vertices and m edges. Let's denote the length of the shortest path from vertex 1 to vertex i as d_i.

You have to erase some edges of the graph so that at most k edges remain. Let's call a vertex i **good** if there still exists a path from 1 to i with length d_i after erasing the edges.

Your goal is to erase the edges in such a way that the number of **good** vertices is maximized.

Input

The first line contains three integers n, m and k ($2 \le n \le 3 \cdot 10^5$, $1 \le m \le 3 \cdot 10^5$, $n - 1 \le m$, $0 \le k \le m$) — the number of vertices and edges in the graph, and the maximum number of edges that can be retained in the graph, respectively.

Then m lines follow, each containing three integers x, y, w ($1 \le x, y \le n$, $x \ne y$, $1 \le w \le 10^9$), denoting an edge connecting vertices x and y and having weight w.

The given graph is connected (any vertex can be reached from any other vertex) and simple (there are no self-loops, and for each unordered pair of vertices there exists at most one edge connecting these vertices).

Output

In the first line print e — the number of edges that should remain in the graph ($0 \le e \le k$).

In the second line print e **distinct** integers from 1 to m — the indices of edges that should remain in the graph. Edges are numbered in the same order they are given in the input. The number of **good** vertices should be as large as possible.

Examples

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

input	output
4 5 2 4 1 8 2 4 1 2 1 3 3 4 9 3 1 5	2 3 2

D. Labyrinth

time limit per test: 2 seconds
memory limit per test: 512 megabytes

You are playing some computer game. One of its levels puts you in a maze consisting of n lines, each of which contains m cells. Each cell either is free or is occupied by an obstacle. The starting cell is in the row r and column c. In one step you can move one square up, left, down or right, if the target cell is not occupied by an obstacle. You can't move beyond the boundaries of the labyrinth.

Unfortunately, your keyboard is about to break, so you can move left no more than x times and move right no more than y times. There are no restrictions on the number of moves up and down since the keys used to move up and down are in perfect condition.

Now you would like to determine for each cell whether there exists a sequence of moves that will put you from the starting cell to this particular one. How many cells of the board have this property?

Input

The first line contains two integers n, m ($1 \le n, m \le 2000$) — the number of rows and the number columns in the labyrinth respectively.

The second line contains two integers r, c ($1 \le r \le n$, $1 \le c \le m$) — index of the row and index of the column that define the starting cell.

The third line contains two integers x, y ($0 \le x, y \le 10^9$) — the maximum allowed number of movements to the left and to the right respectively.

The next n lines describe the labyrinth. Each of them has length of m and consists only of symbols '.' and '*'. The j-th character of the i-th line corresponds to the cell of labyrinth at row i and column j. Symbol '.' denotes the free cell, while symbol '*' denotes the cell with an obstacle.

It is guaranteed, that the starting cell contains no obstacles.

Output

Print exactly one integer — the number of cells in the labyrinth, which are reachable from starting cell, including the starting cell itself.

Examples

input	output
4 5 3 2 1 2 ..... .***. ...** *....	10

input	output
4 4 2 2 0 1 .... ..*.	7

Note

Cells, reachable in the corresponding example, are marked with '+'.

First example:

```
++++..
+***.
+++**
*+++.
```

Second example:

```
.++.
.+*.
.++.
.++.
```

E. Shortest Cycle

time limit per test: 1 second
memory limit per test: 256 megabytes

You are given n integer numbers $a_1, a_2, \ldots, a_n$. Consider graph on n nodes, in which nodes i, j ($i \neq j$) are connected if and only if, a_i AND $a_j \neq 0$, where AND denotes the bitwise AND operation.

Find the length of the shortest cycle in this graph or determine that it doesn't have cycles at all.

Input

The first line contains one integer n ($1 \leq n \leq 10^5$) — number of numbers.

The second line contains n integer numbers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq 10^{18}$).

Output

If the graph doesn't have any cycles, output -1. Else output the length of the shortest cycle.

Examples

input	output
4 3 6 28 9	4

input	output
5 5 12 9 16 48	3

input	output
4 1 2 4 8	-1

Note

In the first example, the shortest cycle is $(9, 3, 6, 28)$.

In the second example, the shortest cycle is $(5, 12, 9)$.

The graph has no cycles in the third example.

F. Two Paths

time limit per test: 2 seconds
memory limit per test: 64 megabytes

As you know, Bob's brother lives in Flatland. In Flatland there are n cities, connected by $n - 1$ two-way roads. The cities are numbered from 1 to n. You can get from one city to another moving along the roads.

The «Two Paths» company, where Bob's brother works, has won a tender to repair two paths in Flatland. A path is a sequence of different cities, connected sequentially by roads. The company is allowed to choose by itself the paths to repair. The only condition they have to meet is that the two paths shouldn't cross (i.e. shouldn't have common cities).

It is known that the profit, the «Two Paths» company will get, equals the product of the lengths of the two paths. Let's consider the length of each road equals 1, and the length of a path equals the amount of roads in it. Find the maximum possible profit for the company.

Input

The first line contains an integer n ($2 \le n \le 200$), where n is the amount of cities in the country. The following $n - 1$ lines contain the information about the roads. Each line contains a pair of numbers of the cities, connected by the road a_i, b_i ($1 \le a_i, b_i \le n$).

Output

Output the maximum possible profit.

Examples

input	output
4 1 2 2 3 3 4	1

input	output
7 1 2 1 3 1 4 1 5 1 6 1 7	0

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

G. Breaking Good

time limit per test: 2 seconds
memory limit per test: 256 megabytes

Breaking Good is a new video game which a lot of gamers want to have. There is a certain level in the game that is really difficult even for experienced gamers.

Walter William, the main character of the game, wants to join a gang called Los Hermanos (The Brothers). The gang controls the whole country which consists of n cities with m bidirectional roads connecting them. There is no road is connecting a city to itself and for any two cities there is at most one road between them. The country is connected, in the other words, it is possible to reach any city from any other city using the given roads.

The roads aren't all working. There are some roads which need some more work to be performed to be completely functioning.

The gang is going to rob a bank! The bank is located in city 1. As usual, the hardest part is to escape to their headquarters where the police can't get them. The gang's headquarters is in city n. To gain the gang's trust, Walter is in charge of this operation, so he came up with a smart plan.

First of all the path which they are going to use on their way back from city 1 to their headquarters n must be as short as possible, since it is important to finish operation as fast as possible.

Then, gang has to blow up all other roads in country that don't lay on this path, in order to prevent any police reinforcements. In case of non-working road, they don't have to blow up it as it is already malfunctional.

If the chosen path has some roads that doesn't work they'll have to repair those roads before the operation.

Walter discovered that there was a lot of paths that satisfied the condition of being shortest possible so he decided to choose among them a path that minimizes the total number of affected roads (both roads that have to be blown up and roads to be repaired).

Can you help Walter complete his task and gain the gang's trust?

Input

The first line of input contains two integers n, m ($2 \le n \le 10^5, 0 \le m \le \min(\frac{n(n-1)}{2}, 10^5)$), the number of cities and number of roads respectively.

In following m lines there are descriptions of roads. Each description consists of three integers x, y, z ($1 \le x, y \le n, z \in \{0, 1\}$) meaning that there is a road connecting cities number x and y. If $z = 1$, this road is working, otherwise it is not.

Output

In the first line output one integer k, the minimum possible number of roads affected by gang.

In the following k lines output three integers describing roads that should be affected. Each line should contain three integers x, y, z ($1 \le x, y \le n$, $z \in \{0, 1\}$), cities connected by a road and the new state of a road. $z = 1$ indicates that the road between cities x and y should be repaired and $z = 0$ means that road should be blown up.

You may output roads in any order. Each affected road should appear exactly once. You may output cities connected by a single road in any order. If you output a road, it's original state should be different from z.

After performing all operations accroding to your plan, there should remain working only roads lying on some certain shortest past between city 1 and n.

If there are multiple optimal answers output any.

Examples

input	output
2 1 1 2 0	1 1 2 1

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

input	output
8 9 1 2 0 8 3 0 2 3 1 1 4 1 8 7 0 1 5 1 4 6 1 5 7 0 6 8 0	3 2 3 0 1 5 0 6 8 1

Note

In the first test the only path is $1 - 2$

In the second test the only shortest path is $1 - 3 - 4$

In the third test there are multiple shortest paths but the optimal is $1 - 4 - 6 - 8$