Day 5. Shortest Paths

A. BFS

1 second, 256 megabytes

You are given an adjacency matrix of an undirected graph with n vertices. Find the length of the shortest path from vertex s to vertex f.

Input

The first line contains three integers n, s and f ($1 \le n \le 100$, $1 \le s, f \le n$) — number of vertices in the graph, vertex s and vertex f, respectively.

Each of the following n lines contains n integers, each of them is either 0 or 1.

In the i-th line on the j-th place there is 1 if vertices i and j are connected by an edge, and 0 if there is no edge between them.

Output

Print a single integer — the length of the shortest path from vertex s to vertex f.

If there is no path between this vertices, print 0.

input
5 5 3 0 0 1 1 0 0 0 0 1 0 1 0 0 0 1 1 1 0 0 1 0 0 1 1 0
output
1

input
4 4 4 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0
output
0

B. Shortest path

1 second, 256 megabytes

You are given a directed weighted graph without negative edges. Find the shortest distance from one vertex to another one.

Input

The first line of the input contains three integers n, s and f ($1 \le n \le 2000$, $1 \le s, f \le n$) — the number of vertices, the start vertex and the end vertex. Next n lines contain n integers each — the adjacency matrix of the graph, where -1 means non-existence of an edge. The weight of each edge does not exceed 10^9. The main diagonal contains only zeros.

Output

The sole line of the output should contain the distance from the start vertex to the end vertex, or -1 if there is no path between them.

input
3 1 2 0 -1 2 3 0 -1 -1 4 0
output
6

C. The shortest path of length K

4 seconds, 256 megabytes

You are given a directed graph. Please, find the shortest paths from S to all other vertices that consist of exactly K edges.

Input

The first line of the input contains four integers n, m, K and S ($1 \le n \le 10^4, 0 \le m \le 10^4, 0 \le K \le 100, 1 \le S \le n$) — the number of vertices, the number of edges, the total number of edges on paths, and the start vertex.

Next m lines contain the description of edges. Each edge is represented with three integers a_i, b_i, w ($1 \leqslant a_i, b_i \leqslant n, -10^5 \leqslant w \leqslant 10^5$) — the ends of the edge and its weight.

Output

The output should contain n integers on a separated line. i-th integer should represent the length of the minimal path between S and t with exactly K edges, or -1 if there is no such path.

input
3 3 1 1 1 2 100 2 3 300 1 3 2
output
-1 100 2

input
3 3 2 1 1 2 100 2 3 300 1 3 2
output
-1 -1 400

D. Dikjstra

1 second, 256 megabytes

You are given a simple undirected connected weighted graph. All edges weights are positive integers.

Your task is to find the shortest distances from the vertex 1 to all other vertices.

Input

The first line of the input contains two integers n and m ($2 \le n \le 10^5; n - 1 \le m \le min(\frac{n(n-1)}{2}, 10^5)$) — the number of vertices and the number of edges in the graph, respectively.

The next m lines describe edges. The edge i is given as three integers x_i, y_i ($1 \le x_i, y_i \le n; x_i \ne y_i$) and w_i ($1 \le w_i \le 10^4$), where x_i and y_i are vertices that the i-th edge connects and w_i is the weight of the i-th edge.

It is guaranteed that the given graph is undirected, simple and connected.

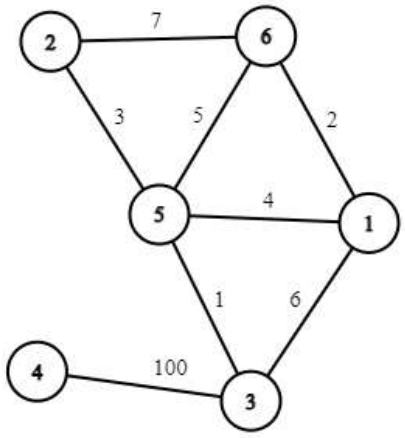
Output

Print n integers $d_1, d_2, \ldots, d_n$, where d_i is the shortest distance from the vertex 1 to the vertex i.

input
6 8 1 3 6 1 5 4 1 6 2 2 5 3 2 6 7 3 4 100 3 5 1 5 6 5
output
0 7 5 105 4 2

input
3 2 1 2 10000 2 3 10000
output
0 10000 20000

The picture corresponding to the first example:



E. Clocks

1 second, 256 megabytes

Electronic clocks show h_1 hours and m_1 minutes. To edit the time, there are two buttons. If the first button is pressed, the hours increase by a_1, the minutes increase by b_1. If the second button is pressed, the hours increase by a_2, the minutes increase by b_2.

The clocks can only show hours from 0 to 23, so the next value after 23 is 0 again. Similarly, the next value after 59 minutes is 0 minutes. If the minutes go from 59 to 0, after some button is pressed, the hours are not increased because of that.

Find the minimum number of presses needed to set h_2 hours and m_2 minutes.

Input

The first line contains integers a_1, b_1, a_2 and b_2 ($0 \le a_1, a_2 \le 23; 0 \le b_1, b_2 \le 59$).

The second line contains integers h_1, m_1 ($0 \le h_1 \le 23; 0 \le m_1 \le 59$).

The third line contains integers h_2, m_2 ($0 \le h_2 \le 23; 0 \le m_2 \le 59$).

Output

Print the smallest number of presses needed. If there is no solution, print -1.

input
1 0 0 1 1 10 2 1
output
52

input
2 5 1 1 0 0 5 11
output
3

input
3 58 17 8 1 59 1 0
output
-1

F. Strange Game

1 second, 256 megabytes

Vitya plays a game for one player. He chooses three integers n, a and b, then, starting from the integer a he wants to reach b with minimum possible number of steps. In each step he can move from integer x to one of the integers $(x + 1) \bmod n$, $(x \cdot x + 1) \bmod n$ or $(x \cdot x \cdot x + 1) \bmod n$. Operation $x \bmod y$ denotes taking the remainder in division of x by y.

Your task is to determine the minimum possible number of steps and the sequence of integers that appear on the steps (including initial a and the target b).

Note that in this problem you should use 64-bit integers.

Input

The first line contains integers n, a and b ($2 \le n \le 100000$, $0 \le a, b < n$).

Output

If there is no way to reach integer b from the integer a by the steps mentioned in the statements, then print -1 in the first line.

Otherwise, print the minimum possible number of steps l in the first line. In the second line print the sequence $a = a_0, a_1, ..., a_l = b$ of integers in the order they appear during the steps.

input
7 5 2
output
2 5 6 2

input
9 2 2
output
0 2

input
100000 99999 0
output
1 99999 0

G. Dikjstra But Cooler

1 second, 256 megabytes

You are given a simple undirected connected weighted graph. But this graph is a bit special. Instead of weighed edges it has **weighed vertices**. All vertices weights are positive integers. All edges have **zero weight**.

Your task is to find the shortest distances from the vertex 1 to all other vertices.

Input

The first line of the input contains two integers n and m ($2 \le n \le 10^5; n - 1 \le m \le min(\frac{n(n-1)}{2}, 10^5)$) — the number of vertices and the number of edges in the graph, respectively.

The second line of the input contains n integers $w_1, w_2, \ldots, w_n$ ($1 \le w_i \le 10^4$), where w_i is the weight of the i-th vertice.

The next m lines describe edges. The edge i is given as two integers x_i and y_i ($1 \le x_i, y_i \le n; x_i \ne y_i$), where x_i and y_i are vertices that the i-th edge connects.

It is guaranteed that the given graph is undirected, simple and connected.

Output

Print n integers $d_1, d_2, \ldots, d_n$, where d_i is the shortest distance from the vertex 1 to the vertex i.

input
6 8 2 5 7 100 2 3 1 3 1 5 1 6 2 5 2 6 3 4 3 5 5 6
output
2 9 9 109 4 5

input
3 2 10000 10000 10000 1 2 2 3
output
10000 20000 30000

H. Raiffeisenbank Logistics

2 seconds, 512 megabytes

One of important business tasks for Raiffeisenbank is to improve the process of delivering cards, valuables, papers and so forth between banks, clients, pick-up points and other locations. You work in a pilot project that plans to revolutionize the world of delivery by using air drones and optimal routing algorithms.

You are now performing the very first tests at onsite training ground that consists of n landing spots. There is a drone at spot 1 and it needs to fly to spot n. Drone's remote control can run m routing programs, the i-th program is able to move drone from spot u_i to spot v_i (if the drone is not at spot u_i, the program will do nothing), and it was written using t_i version of your drone control library. As this is one of the first tests of your project you are not sure about compatibility of different versions of software, so each routing program you run (except for the very first) must have version of drone control library strictly greater, then the previous one.

After looking at the code of routing programs you found out that you can make little changes there. In particular, you can take any program i and swap its starting and destination spots. So, if the i-th program moves drone from spot u_i to spot v_i, after the changes it will move from from spot v_i to spot u_i. Library version t_i will remain unchanged.

Now you wonder, what is the minimum number of programs you need to change, so there will exist a sequence of programs $p_1, p_2, \ldots, p_k$ that being run in this order moves drone from spot 1 to spot n, and $t_{p_1} < t_{p_2} < t_{p_3} < \ldots < t_{p_k}$.

Input

The first line of the input contains a single integer t ($1 \le t \le 1000$) — the number of test cases. Then follow t tests descriptions.

Each test description starts with two integers n and m ($2 \le n \le 500\,000$, $1 \le m \le 500\,000$), the number of landing spots at the training ground and the number of routing programs respectively.

Then follow m lines, the i-th of them describes the i-th program with three integers u_i, v_i and t_i ($1 \le u_i, v_i \le n$, $1 \le t_i \le 10^9$), meaning this program moves drone from spot u_i to spot v_i and uses t_i-th version of drone control library.

There can be multiple programs with the same version of software. There can be multiple programs connecting the same pair of spots (in each direction). There can be programs that move drone from some spot to itself.

It is guaranteed that sum of n over all tests won't exceed $500\,000$. It is guaranteed that sum of m over all tests won't exceed $500\,000$.

Output

If there is no way to change programs and move drone from spot 1 to spot n obeying condition that every next program you run must have greater version of drone control library, print -1 in the only line of the output.

Otherwise, print the minimum number of programs you need to change to achieve the goal.

input

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

output

```
2
```

input

```
2
4 3
2 1 1
2 3 2
4 3 2
8 9
1 2 5
2 3 10
4 3 15
4 5 20
5 8 25
1 6 2
6 5 30
7 6 3
8 7 4
```

output

```
-1
1
```

I. Cycle of negative weight

2 seconds, 256 megabytes

You are given a directed weighted graph. Please find the negative cycle, or answer that it does not exist.

Input

The first line of the input contains one integer n ($1 \le n \le 100$) — the number of vertices. Next n lines contain n integers each and represent an adjacency matrix with weights. The weights do not exceed $10\,000$ in absolute value. If there is no edge, the corresponding weight is equal to $100\,000$.

Output

The first line of the output should contain "YES", if the cycle exists, or "NO", otherwise. If the cycle exists the second line should contain the number of vertices in the cycle, and the third line should contain the vertices in the order of the cycle.

input

```
2
0 -1
-1 0
```

output

```
YES
2
2 1
```

J. Transportation

2 seconds, 256 megabytes

You are an executive of the transportation company in town 1 and you have to transfer a lot of cups into town with identifier n. You have one car and there is a limit on each road — the maximal possible weight of the car. You are given exactly 24 hours to deliver, find the maximum possible number of cups. Please note that once the car gets to town n it cannot return to town 1.

Input

The first line of the input contains two integers n and m ($2 \le n \le 500$, $0 \le m \le \frac{n \cdot (n-1)}{2}$) — the number of towns and the number of bidirected roads, correspondingly.

Next m lines contain the description of roads. Each road is specified with four integers a_i, b_i, t_i, w_i ($t_i \le 1440$, $w_i \le 10^9$, $1 \le a_i, b_i \le n$) — the ends of the road, the time to pass the road in minutes, and the maximum possible weight in grams.

There can be only one road between any pair of towns. Each cup weighs 100 grams and an empty car weighs 3 tons.

Output

The sole line of the output should contain one integer — the maximum number of cups that can be transferred in 24 hours.

input

```
3 3
1 2 10 3000220
2 3 20 3000201
1 3 1 3000099
```

output

```
2
```

K. Random's Games

10 seconds, 512 megabytes

Many years have passed since the Chaos was defeated in a final battle, and Random, son of Oberon, was chosen as the new king of Amber. Since nothing interesting is happening anymore, Random got bored and is now playing his favorite game again — that is, he takes some well-known algorithmic problem he likes and introduces some randomness to its input data. This time he picks the shortest path in undirected graphs.

First, he considers n shadow worlds and removes all the ways to travel between them. Then, for m times Random selects two distinct worlds u and v **at random**. This means that any pair is equiprobable even it was already selected before. Then, one of the following happens.

1. Random creates a bidirectional connection between world u and world v.
2. Random wants to know whether it is possible to travel from world u to world v using only connections he already created (using some intermediate worlds is allowed). If the answer is positive, he wants to know the minimum number of connections one has to use in order to do so.

Random believes that a random structure of the connections' system makes it possible to solve the problem under the given constraints. Prove him right.

Input

The first line of the input contains two integers n ($n = 100\,000$) and m ($m = 1\,000\,000$) — the number of worlds Random considers and the number of actions he performs.

Then follow m triples t_i, u_i and v_i ($1 \le t_i \le 2$, $1 \le u_i, v_i \le n$, $u_i \ne v_i$) — the type of the i-th action and indices of the world being involved respectively.

It is guaranteed that there will be exactly $100\,000$ **shortest path queries**. Moreover, it's guaranteed that they are randomly and equiprobably distributed among all m queries. The number of worlds and steps is the same in all tests except for a sample.

Output

For each question of the second type print one integer. If there is no way to travel between two worlds in question, print -1. Otherwise, print the minimum number of direct connections one has to use.

input
8 10 2 1 2 1 1 3 1 4 3 1 6 8 1 5 4 2 6 8 2 1 5 1 1 3 1 1 4 2 5 1
output
-1 1 3 2

We guarantee the randomness of test cases on the "best effort" level. Pseudo-random generators and human-generated random seed's were used. No further selection was done to avoid any sort of feedback loop.

L. Don't Annoy the Mayor

4 seconds, 256 megabytes

There are n intersections in Nsk and m bidirectional roads connecting them. It is guaranteed that:

- any intersection can be reached from any other intersection using only given roads,
- any pair of intersections is connected by at most one road, and
- no road connects an intersection to itself.

A *path* of length $(l - 1)$ is a sequence of intersections $p_1, p_2, p_3, \ldots, p_l$ such that there is a road connecting any pair of consecutive intersections. Any intersection can appear in the sequence an arbitrary number of times. We will define the beginning of the path as p_1, and the end of the path as p_l. The number l is considered to be positive.

The mayor plans to inspect the town entrusted to him, so he ordered his assistant to plan a path from intersection s to intersection t. No other restrictions were given, so the assistant is free to choose **any** path that has s as its beginning and t as its end.

For each road in town, the assistant knows a value w_i that defines how much will the mayor's annoyance change every time he passes through this road. Moreover, he knows how much money c_i must be spent to decrease the value w_i by one. For any road, this operation can be applied any number of times (this number must be a non-negative integer).

The assistant also knows that initial mayor's annoyance is equal to a, and if this value exceeds some threshold value b at the end of the trip, the mayor can accidentally fire someone while having a tantrum. Taking all this information into account, the assistant wants to spent as little money as possible on decreasing some w_i, and then create a path from s to t such that the mayor's annoyance will not be greater than the threshold value **at the end** of the journey. As the mayor is very emotional, one may assume his annoyance can take any integer value.

Input

On the first line of the input, two integers n and m are given: the number of intersections and the number of roads connecting them, respectively ($1 \le n, m \le 200\,000$).

On the second line, there are two numbers s and t: the first and last intersections of the desired path ($1 \le s, t \le n$).

The third line contains the initial annoyance of the mayor a and the threshold value b that must not be exceeded after passing the last road of the path ($-10^9 \le a \le b \le 10^9$).

The next m lines contain road descriptions. Each of these lines contains v_i and u_i, the numbers of connected intersections, followed by the initial integer value of w_i and integer cost c_i of decreasing the latter value by one ($1 \le v_i, u_i \le n$, $v_i \ne u_i$, $0 \le |w_i|, c_i \le 10^9$).

Output

Print a single integer to the output: the minimum possible amount of money that must be spent so that it is then possible to create a path beginning in s and ending in t which guarantees the assistant will not be fired.

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

input
3 2 3 1 2 2 1 2 -1 10 3 2 1 10
output
0