A. Two Buttons

2 seconds, 256 megabytes

Vasya has found a strange device. On the front panel of a device there are: a red button, a blue button and a display showing some positive integer. After clicking the red button, device multiplies the displayed number by two. After clicking the blue button, device subtracts one from the number on the display. If at some point the number stops being positive, the device breaks down. The display can show arbitrarily large numbers. Initially, the display shows number n.

Bob wants to get number m on the display. What minimum number of clicks he has to make in order to achieve this result?

Input

The first and the only line of the input contains two distinct integers n and m ($1 \le n, m \le 10^4$), separated by a space .

Output

Print a single number — the minimum number of times one needs to push the button required to get the number m out of number n.

input
4 6
output
2

input
10 1
output
9

In the first example you need to push the blue button once, and then push the red button once.

In the second example, doubling the number is unnecessary, so we need to push the blue button nine times.

B. Greg and Graph

8.0 s, 256 megabytes

Greg has a weighed directed graph, consisting of n vertices. In this graph any pair of distinct vertices has an edge between them in both directions. Greg loves playing with the graph and now he has invented a new game:

- The game consists of n steps.
- On the i-th step Greg removes vertex number x_i from the graph. As Greg removes a vertex, he also removes all the edges that go in and out of this vertex.
- Before executing each step, Greg wants to know the sum of lengths of the shortest paths between all pairs of the remaining vertices. The shortest path can go through any remaining vertex. In other words, if we assume that $d(i, v, u)$ is the shortest path between vertices v and u in the graph that formed before deleting vertex x_i, then Greg wants to know the value of the following sum: $\sum_{v,u,v\neq u} d(i, v, u)$.

Help Greg, print the value of the required sum before each step.

Input

The first line contains integer n $(1 \le n \le 500)$ — the number of vertices in the graph.

Next n lines contain n integers each — the graph adjacency matrix: the j-th number in the i-th line a_{ij} $(1 \le a_{ij} \le 10^5, a_{ii} = 0)$ represents the weight of the edge that goes from vertex i to vertex j.

The next line contains n distinct integers: $x_1, x_2, ..., x_n$ $(1 \le x_i \le n)$ — the vertices that Greg deletes.

Output

Print n integers — the i-th number equals the required sum before the i-th step.

Please, do not use the `%lld` specifier to read or write 64-bit integers in *C++*. It is preferred to use the `cin`, `cout` streams of the `%I64d` specifier.

input
1 0 1
output
0

input
2 0 5 4 0 1 2
output
9 0

input
4 0 3 1 1 6 0 400 1 2 4 0 1 1 1 1 0 4 1 2 3
output
17 23 404 0

C. Fight Against Traffic

1 second, 256 megabytes

Little town Nsk consists of n junctions connected by m bidirectional roads. Each road connects two distinct junctions and no two roads connect the same pair of junctions. It is possible to get from any junction to any other junction by these roads. The distance between two junctions is equal to the minimum possible number of roads on a path between them.

In order to improve the transportation system, the city council asks mayor to build one new road. The problem is that the mayor has just bought a wonderful new car and he really enjoys a ride from his home, located near junction s to work located near junction t. Thus, he wants to build a new road in such a way that the distance between these two junctions won't decrease.

You are assigned a task to compute the number of pairs of junctions that are not connected by the road, such that if the new road between these two junctions is built the distance between s and t won't decrease.

Input

The firt line of the input contains integers n, m, s and t ($2 \le n \le 1000$, $1 \le m \le 1000$, $1 \le s, t \le n$, $s \ne t$) — the number of junctions and the number of roads in Nsk, as well as the indices of junctions where mayors home and work are located respectively. The i-th of the following m lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$), meaning that this road connects junctions u_i and v_i directly. It is guaranteed that there is a path between any two junctions and no two roads connect the same pair of junctions.

Output

Print one integer — the number of pairs of junctions not connected by a direct road, such that building a road between these two junctions won't decrease the distance between junctions s and t.

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

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

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

D. Shortest Path

3 seconds, 256 megabytes

In Ancient Berland there were n cities and m two-way roads of equal length. The cities are numbered with integers from 1 to n inclusively. According to an ancient superstition, if a traveller visits three cities a_i, b_i, c_i in row, without visiting other cities between them, a great disaster awaits him. Overall there are k such city triplets. Each triplet is ordered, which means that, for example, you are allowed to visit the cities in the following order: a_i, c_i, b_i. Vasya wants to get from the city 1 to the city n and not fulfil the superstition. Find out which minimal number of roads he should take. Also you are required to find one of his possible path routes.

Input

The first line contains three integers n, m, k ($2 \le n \le 3000$, $1 \le m \le 20000$, $0 \le k \le 10^5$) which are the number of cities, the number of roads and the number of the forbidden triplets correspondingly.

Then follow m lines each containing two integers x_i, y_i ($1 \le x_i, y_i \le n$) which are the road descriptions. The road is described by the numbers of the cities it joins. No road joins a city with itself, there cannot be more than one road between a pair of cities.

Then follow k lines each containing three integers a_i, b_i, c_i ($1 \le a_i, b_i, c_i \le n$) which are the forbidden triplets. Each ordered triplet is listed mo more than one time. All three cities in each triplet are distinct.

City n can be unreachable from city 1 by roads.

Output

If there are no path from 1 to n print `-1`. Otherwise on the first line print the number of roads d along the shortest path from the city 1 to the city n. On the second line print $d + 1$ numbers — any of the possible shortest paths for Vasya. The path should start in the city 1 and end in the city n.

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

input
3 1 0 1 2
output
-1

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

E. Elevator

3 seconds, 256 megabytes

You work in a big office. It is a 9 floor building with an elevator that can accommodate up to 4 people. It is your responsibility to manage this elevator.

Today you are late, so there are queues on some floors already. For each person you know the floor where he currently is and the floor he wants to reach. Also, you know the order in which people came to the elevator.

According to the company's rules, if an employee comes to the elevator earlier than another one, he has to enter the elevator earlier too (even if these employees stay on different floors). Note that the employees are allowed to leave the elevator in arbitrary order.

The elevator has two commands:

- Go up or down one floor. The movement takes 1 second.
- Open the doors on the current floor. During this operation all the employees who have reached their destination get out of the elevator. Then all the employees on the floor get in the elevator in the order they are queued up while it doesn't contradict the company's rules and there is enough space in the elevator. Each employee spends 1 second to get inside and outside the elevator.

Initially the elevator is empty and is located on the floor 1.

You are interested what is the minimum possible time you need to spend to deliver all the employees to their destination. It is not necessary to return the elevator to the floor 1.

Input

The first line contains an integer n ($1 \le n \le 2000$) — the number of employees.

The i-th of the next n lines contains two integers a_i and b_i ($1 \le a_i, b_i \le 9, a_i \ne b_i$) — the floor on which an employee initially is, and the floor he wants to reach.

The employees are given in the order they came to the elevator.

Output

Print a single integer — the minimal possible time in seconds.

input
2 3 5 5 3
output
10

input
2 5 3 3 5
output
12

See on-line problem description for explanation of first example.

F. Dynamic Shortest Path

10 seconds, 512 megabytes

You are given a weighted directed graph, consisting of n vertices and m edges. You should answer q queries of two types:

- `1 v` — find the length of shortest path from vertex 1 to vertex v.
- `2 c` $l_1\ l_2\ ...\ l_c$ — add 1 to weights of edges with indices $l_1, l_2, ..., l_c$.

Input

The first line of input data contains integers n, m, q ($1 \le n, m \le 10^5$, $1 \le q \le 2000$) — the number of vertices and edges in the graph, and the number of requests correspondingly.

Next m lines of input data contain the descriptions of edges: i-th of them contains description of edge with index i — three integers a_i, b_i, c_i ($1 \le a_i, b_i \le n$, $0 \le c_i \le 10^9$) — the beginning and the end of edge, and its initial weight correspondingly.

Next q lines of input data contain the description of edges in the format described above ($1 \le v \le n$, $1 \le l_j \le m$). It's guaranteed that inside single query all l_j are distinct. Also, it's guaranteed that a total number of edges in all requests of the second type does not exceed 10^6.

Output

For each query of first type print the length of the shortest path from 1 to v in a separate line. Print `-1`, if such path does not exists.

input

```
3 2 9
1 2 0
2 3 0
2 1 2
1 3
1 2
2 1 1
1 3
1 2
2 2 1 2
1 3
1 2
```

See on-line problem description for explanations of these inputs and outputs.

output

```
1
0
2
1
4
2
```

input

```
5 4 9
2 3 1
2 4 1
3 4 1
1 2 0
1 5
1 4
2 1 2
2 1 2
1 4
2 2 1 3
1 4
2 1 4
1 4
```

output

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