Codeforces Round #119 (Div. 2)

A. Cut Ribbon

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarpus has a ribbon, its length is n. He wants to cut the ribbon in a way that fulfils the following two conditions:

- After the cutting each ribbon piece should have length a, b or c.
- After the cutting the number of ribbon pieces should be maximum.

Help Polycarpus and find the number of ribbon pieces after the required cutting.

Input
The first line contains four space-separated integers n, a, b and c ($1 \le n, a, b, c \le 4000$) — the length of the original ribbon and the acceptable lengths of the ribbon pieces after the cutting, correspondingly. The numbers a, b and c can coincide.

Output
Print a single number — the maximum possible number of ribbon pieces. It is guaranteed that at least one correct ribbon cutting exists.

Examples

input
5 5 3 2
output
2

input
7 5 5 2
output
2

Note
In the first example Polycarpus can cut the ribbon in such way: the first piece has length 2, the second piece has length 3.

In the second example Polycarpus can cut the ribbon in such way: the first piece has length 5, the second piece has length 2.

B. Counting Rhombi

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have two positive integers w and h. Your task is to count the number of rhombi which have the following properties:

- Have positive area.
- With vertices at integer points.
- All vertices of the rhombi are located inside or on the border of the rectangle with vertices at points $(0, 0)$, $(w, 0)$, (w, h), $(0, h)$. In other words, for all vertices (x_i, y_i) of the rhombus the following conditions should fulfill: $0 \le x_i \le w$ and $0 \le y_i \le h$.
- Its diagonals are parallel to the axis.

Count the number of such rhombi.

Let us remind you that a *rhombus* is a quadrilateral whose four sides all have the same length.

Input

The first line contains two integers w and h $(1 \le w, h \le 4000)$ — the rectangle's sizes.

Output

Print a single number — the number of sought rhombi.

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

Examples

input
2 2
output
1

input
1 2
output
0

Note

In the first example there exists only one such rhombus. Its vertices are located at points $(1, 0)$, $(2, 1)$, $(1, 2)$, $(0, 1)$.

C. Permutations

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Happy PMP is freshman and he is learning about algorithmic problems. He enjoys playing algorithmic games a lot.

One of the seniors gave Happy PMP a nice game. He is given two permutations of numbers 1 through n and is asked to convert the first one to the second. In one move he can remove the last number from the permutation of numbers and inserts it back in an arbitrary position. He can either insert last number between any two consecutive numbers, or he can place it at the beginning of the permutation.

Happy PMP has an algorithm that solves the problem. But it is not fast enough. He wants to know the minimum number of moves to convert the first permutation to the second.

Input

The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the quantity of the numbers in the both given permutations.

Next line contains n space-separated integers — the first permutation. Each number between 1 to n will appear in the permutation exactly once.

Next line describe the second permutation in the same format.

Output

Print a single integer denoting the minimum number of moves required to convert the first permutation to the second.

Examples

input
3 3 2 1 1 2 3
output
2

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

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

Note

In the first sample, he removes number 1 from end of the list and places it at the beginning. After that he takes number 2 and places it between 1 and 3.

In the second sample, he removes number 5 and inserts it after 1.

In the third sample, the sequence of changes are like this:

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

So he needs three moves.

D. AlgoRace

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

PMP is getting a warrior. He is practicing a lot, but the results are not acceptable yet. This time instead of programming contests, he decided to compete in a car racing to increase the spirit of victory. He decides to choose a competition that also exhibits algorithmic features.

AlgoRace is a special league of car racing where different teams compete in a country of n cities. Cities are numbered 1 through n. Every two distinct cities in the country are connected with one bidirectional road. Each competing team should introduce one driver and a set of cars.

The competition is held in r rounds. In i-th round, drivers will start at city s_i and finish at city t_i. Drivers are allowed to change their cars at most k_i times. Changing cars can take place in any city in no time. One car can be used multiple times in one round, but total number of changes should not exceed k_i. Drivers can freely choose their path to destination.

PMP has prepared m type of purpose-built cars. Beside for PMP's driving skills, depending on properties of the car and the road, a car traverses each road in each direction in different times.

PMP Warriors wants to devise best strategies of choosing car and roads in each round to maximize the chance of winning the cup. For each round they want to find the minimum time required to finish it.

Input

The first line contains three space-separated integers n, m, r ($2 \le n \le 60, 1 \le m \le 60, 1 \le r \le 10^5$) — the number of cities, the number of different types of cars and the number of rounds in the competition, correspondingly.

Next m sets of $n \times n$ matrices of integers between 0 to 10^6 (inclusive) will follow — describing the time one car requires to traverse different roads. The k-th integer in j-th line of the i-th set is the time that i-th car requires to traverse the road from j-th city to k-th city. These matrices are not necessarily symmetric, but their diagonal is always zero.

Next r lines contain description of the rounds. The i-th of these lines contains space-separated integers s_i, t_i, k_i ($1 \le s_i, t_i \le n, s_i \ne t_i, 0 \le k_i \le 1000$) — the number of starting city, finishing city and the number of possible car changes in i-th round, correspondingly.

Output

For each round you should print the minimum required time to complete the round in a single line.

Examples

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

output
3 4 3

input
4 2 3 0 7 3 3 8 0 10 5 1 1 0 4 8 9 2 0 0 3 3 9 7 0 4 9 3 8 0 4 4 8 9 0 2 3 3 2 1 3 1 2 2

output
4

```
5
3
```

Note

In the first sample, in all rounds PMP goes from city #1 to city #2, then city #3 and finally city #4. But the sequences of types of the cars he uses are (1, 2, 1) in the first round and (1, 2, 2) in the second round. In the third round, although he can change his car three times, he uses the same strategy as the first round which only needs two car changes.

E. Weak Memory

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Zart PMP is qualified for ICPC World Finals in Harbin, China. After team excursion to Sun Island Park for snow sculpture art exposition, PMP should get back to buses before they leave. But the park is really big and he does not know how to find them.

The park has n intersections numbered 1 through n. There are m bidirectional roads that connect some pairs of these intersections. At k intersections, ICPC volunteers are helping the teams and showing them the way to their destinations. Locations of volunteers are fixed and distinct.

When PMP asks a volunteer the way to bus station, he/she can tell him the whole path. But the park is fully covered with ice and snow and everywhere looks almost the same. So PMP can only memorize at most q intersections after each question (excluding the intersection they are currently standing). He always tells volunteers about his weak memory and if there is no direct path of length (in number of roads) at most q that leads to bus station, the volunteer will guide PMP to another volunteer (who is at most q intersections away, of course). ICPC volunteers know the area very well and always tell PMP the best way. So if there exists a way to bus stations, PMP will definitely find it.

PMP's initial location is intersection s and the buses are at intersection t. There will always be a volunteer at intersection s. Your job is to find out the minimum q which guarantees that PMP can find the buses.

Input

The first line contains three space-separated integers n, m, k ($2 \le n \le 10^5, 0 \le m \le 2 \cdot 10^5, 1 \le k \le n$) — the number of intersections, roads and volunteers, respectively. Next line contains k distinct space-separated integers between 1 and n inclusive — the numbers of cities where volunteers are located.

Next m lines describe the roads. The i-th of these lines contains two space-separated integers u_i, v_i ($1 \le u_i, v_i \le n, u_i \ne v_i$) — two intersections that i-th road connects. There will be at most one road between any two intersections.

Last line of input contains two space-separated integers s, t ($1 \le s, t \le n, s \ne t$) — the initial location of PMP and the location of the buses. It might not always be possible to reach t from s.

It is guaranteed that there is always a volunteer at intersection s.

Output

Print on the only line the answer to the problem — the minimum value of q which guarantees that PMP can find the buses. If PMP cannot reach the buses at all, output -1 instead.

Examples

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

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

Note

The first sample is illustrated below. Blue intersections are where volunteers are located. If PMP goes in the path of dashed line, it can reach the buses with $q = 3$:

In the second sample, PMP uses intersection 6 as an intermediate intersection, thus the answer is 3.
