A. Kyoya and Colored Balls

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

Kyoya Ootori has a bag with n colored balls that are colored with k different colors. The colors are labeled from 1 to k. Balls of the same color are indistinguishable. He draws balls from the bag one by one until the bag is empty. He noticed that he drew the last ball of color i before drawing the last ball of color $i + 1$ for all i from 1 to $k - 1$. Now he wonders how many different ways this can happen.

Input

The first line of input will have one integer k $(1 \le k \le 1000)$ the number of colors.

Then, k lines will follow. The i-th line will contain c_i, the number of balls of the i-th color $(1 \le c_i \le 1000)$.

The total number of balls doesn't exceed 1000.

Output

A single integer, the number of ways that Kyoya can draw the balls from the bag as described in the statement, modulo $1\,000\,000\,007$.

Sample test(s)

input

```
3
2
2
1
```

output

```
3
```

input

```
4
1
2
3
4
```

output

```
1680
```

Note

In the first sample, we have 2 balls of color 1, 2 balls of color 2, and 1 ball of color 3. The three ways for Kyoya are:

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

B. Kyoya and Permutation

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

Let's define the permutation of length n as an array $p = [p_1, p_2, ..., p_n]$ consisting of n distinct integers from range from 1 to n. We say that this permutation maps value 1 into the value p_1, value 2 into the value p_2 and so on.

Kyota Ootori has just learned about *cyclic representation* of a permutation. A *cycle* is a sequence of numbers such that each element of this sequence is being mapped into the next element of this sequence (and the last element of the cycle is being mapped into the first element of the cycle). The *cyclic representation* is a representation of p as a collection of cycles forming p. For example, permutation $p = [4, 1, 6, 2, 5, 3]$ has a *cyclic representation* that looks like $(142)(36)(5)$ because 1 is replaced by 4, 4 is replaced by 2, 2 is replaced by 1, 3 and 6 are swapped, and 5 remains in place.

Permutation may have several cyclic representations, so Kyoya defines the *standard cyclic representation* of a permutation as follows. First, reorder the elements within each cycle so the largest element is first. Then, reorder all of the cycles so they are sorted by their first element. For our example above, the *standard cyclic representation* of $[4, 1, 6, 2, 5, 3]$ is $(421)(5)(63)$.

Now, Kyoya notices that if we drop the parenthesis in the standard cyclic representation, we get another permutation! For instance, $[4, 1, 6, 2, 5, 3]$ will become $[4, 2, 1, 5, 6, 3]$.

Kyoya notices that some permutations don't change after applying operation described above at all. He wrote all permutations of length n that do not change in a list in lexicographic order. Unfortunately, his friend Tamaki Suoh lost this list. Kyoya wishes to reproduce the list and he needs your help. Given the integers n and k, print the permutation that was k-th on Kyoya's list.

Input

The first line will contain two integers n, k ($1 \le n \le 50$, $1 \le k \le min\{10^{18}, l\}$ where l is the length of the Kyoya's list).

Output

Print n space-separated integers, representing the permutation that is the answer for the question.

Sample test(s)

input

```
4 3
```

output

```
1 3 2 4
```

input

```
10 1
```

output

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

Note

The standard cycle representation is $(1)(32)(4)$, which after removing parenthesis gives us the original permutation. The first permutation on the list would be $[1, 2, 3, 4]$, while the second permutation would be $[1, 2, 4, 3]$.

C. Love Triangles

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

There are many anime that are about "love triangles": Alice loves Bob, and Charlie loves Bob as well, but Alice hates Charlie. You are thinking about an anime which has n characters. The characters are labeled from 1 to n. Every pair of two characters can either mutually love each other or mutually hate each other (there is no neutral state).

You hate love triangles (A-B are in love and B-C are in love, but A-C hate each other), and you also hate it when nobody is in love. So, considering any three characters, you will be happy if exactly one pair is in love (A and B love each other, and C hates both A and B), or if all three pairs are in love (A loves B, B loves C, C loves A).

You are given a list of m known relationships in the anime. You know for sure that certain pairs love each other, and certain pairs hate each other. You're wondering how many ways you can fill in the remaining relationships so you are happy with every triangle. Two ways are considered different if two characters are in love in one way but hate each other in the other. Print this count modulo $1\,000\,000\,007$.

Input

The first line of input will contain two integers n, m ($3 \le n \le 100\,000$, $0 \le m \le 100\,000$).

The next m lines will contain the description of the known relationships. The i-th line will contain three integers a_i, b_i, c_i. If c_i is 1, then a_i and b_i are in love, otherwise, they hate each other ($1 \le a_i, b_i \le n$, $a_i \ne b_i$, $c_i \in \{0, 1\}$).

Each pair of people will be described no more than once.

Output

Print a single integer equal to the number of ways to fill in the remaining pairs so that you are happy with every triangle modulo $1\,000\,000\,007$.

Sample test(s)

input

```
3 0
```

output

```
4
```

input

```
4 4
1 2 1
2 3 1
3 4 0
4 1 0
```

output

```
1
```

input

```
4 4
1 2 1
2 3 1
3 4 0
4 1 1
```

output

```
0
```

Note

In the first sample, the four ways are to:

- Make everyone love each other
- Make 1 and 2 love each other, and 3 hate 1 and 2 (symmetrically, we get 3 ways from this).

In the second sample, the only possible solution is to make 1 and 3 love each other and 2 and 4 hate each other.

D. Nudist Beach

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

Nudist Beach is planning a military operation to attack the Life Fibers. In this operation, they will attack and capture several cities which are currently under the control of the Life Fibers.

There are n cities, labeled from 1 to n, and m bidirectional roads between them. Currently, there are Life Fibers in every city. In addition, there are k cities that are fortresses of the Life Fibers that cannot be captured under any circumstances. So, the Nudist Beach can capture an arbitrary non-empty subset of cities with no fortresses.

After the operation, Nudist Beach will have to defend the captured cities from counterattack. If they capture a city and it is connected to many Life Fiber controlled cities, it will be easily defeated. So, Nudist Beach would like to capture a set of cities such that for each captured city the ratio of Nudist Beach controlled neighbors among all neighbors of that city is as high as possible.

More formally, they would like to capture a non-empty set of cities S with no fortresses of Life Fibers. The strength of a city $x \in S$ is defined as (number of neighbors of x in S) / (total number of neighbors of x). Here, two cities are called neighbors if they are connnected with a road. The goal is to maximize the strength of the weakest city in S.

Given a description of the graph, and the cities with fortresses, find a non-empty subset that maximizes the strength of the weakest city.

Input

The first line of input contains three integers n, m, k ($2 \le n \le 100\,000$, $1 \le m \le 100\,000$, $1 \le k \le n - 1$).

The second line of input contains k integers, representing the cities with fortresses. These cities will all be distinct.

The next m lines contain the roads. The i-th of these lines will have 2 integers a_i, b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$). Every city will have at least one road adjacent to it.

There is no more than one road between each pair of the cities.

Output

The first line should contain an integer r, denoting the size of an optimum set ($1 \le r \le n - k$).

The second line should contain r integers, denoting the cities in the set. Cities may follow in an arbitrary order. This line should not contain any of the cities with fortresses.

If there are multiple possible answers, print any of them.

Sample test(s)

input

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

output

```
3
1 4 5
```

input

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

output

```
8
1 5 4 8 10 6 3 7
```

Note

The first example case achieves a strength of 1/2. No other subset is strictly better.

The second example case achieves a strength of 1. Note that the subset doesn't necessarily have to be connected.

E. Kyoya and Train

time limit per test: 8 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Kyoya Ootori wants to take the train to get to school. There are n train stations and m one-way train lines going between various stations. Kyoya is currently at train station 1, and the school is at station n. To take a train, he must pay for a ticket, and the train also takes a certain amount of time. However, the trains are not perfect and take random amounts of time to arrive at their destination. If Kyoya arrives at school strictly after t time units, he will have to pay a fine of x.

Each train line is described by a ticket price, and a probability distribution on the time the train takes. More formally, train line i has ticket cost c_i, and a probability distribution $p_{i,k}$ which denotes the probability that this train will take k time units for all $1 \le k \le t$. Amounts of time that each of the trains used by Kyouya takes are mutually independent random values (moreover, if Kyoya travels along the same train more than once, it is possible for the train to take different amounts of time and those amounts are also independent one from another).

Kyoya wants to get to school by spending the least amount of money in expectation (for the ticket price plus possible fine for being late). Of course, Kyoya has an optimal plan for how to get to school, and every time he arrives at a train station, he may recalculate his plan based on how much time he has remaining. What is the expected cost that Kyoya will pay to get to school if he moves optimally?

Input

The first line of input contains four integers n, m, t, x ($2 \le n \le 50$, $1 \le m \le 100$, $1 \le t \le 20\,000$, $0 \le x \le 10^6$).

The next $2m$ lines contain the description of the trains.

The $2i$-th line will have 3 integers a_i, b_i, c_i, representing a one way train from station a_i to b_i with ticket cost c_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$, $0 \le c_i \le 10^6$). There will always be at least one path from any station to the school.

The $(2i + 1)$-th line will contain t integers, $p_{i,1}, p_{i,2}, \ldots, p_{i,t}$ where $p_{i,k} / 100000$ is the probability that this train will take k units of time to traverse ($0 \le p_{i,k} \le 100\,000$ for $1 \le k \le t$, $\sum_{k=1}^{t} p_{i,k} = 100000$).

It is guaranteed that there is no more than one train between each pair of platforms in each of the directions.

Output

Print a single real number that is equal to an optimal expected cost of getting to school. The answer will be considered correct if its relative or absolute error doesn't exceed 10^{-6}.

Sample test(s)

input

```
4 4 5 1
1 2 0
50000 0 50000 0 0
2 3 0
10000 0 0 0 90000
3 4 0
100000 0 0 0 0
2 4 0
0 0 0 50000 50000
```

output

```
0.7000000000
```

input

```
4 4 5 1
1 2 100
50000 0 50000 0 0
2 3 100
10000 0 0 0 90000
3 4 100
100000 0 0 0 0
2 4 100
0 0 0 50000 50000
```

output

```
200.7500000000
```

Note

The optimal strategy in the first case is as follows:

First, travel along first train line. With probability $1 / 2$ Kyoya will take 1 time unit. Otherwise, Kyoya will take 3 time units.

If the train takes 1 time unit, travel along the 4th train line. Kyoya will make it to school in time with probability $1 / 2$. Otherwise, if the train takes 3 time units, travel along the 2nd train line. Kyoya will make it to school in time with probability $1 / 10$.

Since the cost of all train lines are zero, we can just look at the probability that Kyoya will incur the penalty. The probability that Kyoya will have to pay the penalty is $1/2 \times 1/2 + 1/2 \times 9/10 = 7/10$. We can show that no other strategy is strictly better.

The optimal strategy in the second case is to travel along $1 \to 2 \to 4$ no matter what. Kyoya will incur the penalty with probability $3/4$, and the cost of the trains is 200, thus the expected cost is 200.75.