6195 The Dueling Philosophers Problem

Following a sad and strange incident involving a room full of philosophers, several plates of spaghetti, and one too few forks, the faculty of the Department of Philosophy at ... University have been going through the papers of a recently deceased colleague. The faculty members were amazed to find numerous unpublished essays. They believe that the essays, collected into one volume, may constitute a major work of scholarship that will give their department some much-needed positive publicity. Naturally, all of the faculty members began to vie for the honor (to say nothing of the fame) of serving as editor of the collection.

After much debate, the faculty members have narrowed the list to two candidates. Both applicants were asked to explain how they would arrange the essays within the final book. Both have noted that many of the essays define terminology and concepts that are explored in other essays, and both have agreed to the basic principle that an essay that *uses* a term must itself *define* that term or appear *after* the essay that defines it.

One of the candidates has presented what he claims is the only possible arrangement of the essays under those constraints, and is arguing that he should be given the job simply because he has already done this major part of the work. The second candidate scoffs at this claim, insisting that there are many possible arrangements of the essays, and that an editor of true skill (himself) is needed to choose the optimal arrangement.

Write a program to determine if zero, one, or more than one arrangement of the essays is possible.

Input

There will be multiple test cases in the input.

Each test case will begin with a line with two integers, n ($1 \leq n \leq 1,000$) and m ($1 \leq m \leq 50,000$), where n is the number of essays, and m is the number of relationships between essays caused by sharing terms.

On each of the next m lines will be two integers, d and u ($1 \leq u, d \leq n, d \neq u$) which indicate that a term is defined in essay d and used in essay u.

The input will end with two `0`'s on their own line.

Output

For each test case, print a single line of output containing a '`0`' if no arrangement is possible, a '`1`' if exactly one arrangement is possible, or a '`2`' if multiple arrangements are possible (the output will be '`2`' no matter how many arrangements there are).

Sample Input

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

Sample Output

```
2
1
0
```

Hongcow Builds A Nation

Hongcow is ruler of the world. As ruler of the world, he wants to make it easier for people to travel by road within their own countries.

The world can be modeled as an undirected graph with n nodes and m edges. k of the nodes are home to the governments of the k countries that make up the world.

There is at most one edge connecting any two nodes and no edge connects a node to itself. Furthermore, for any two nodes corresponding to governments, **there is no path between those two nodes**. Any graph that satisfies all of these conditions is *stable*.

Hongcow wants to add as many edges as possible to the graph while keeping it stable. Determine the maximum number of edges Hongcow can add.

Input

The first line of input will contain three integers n, m and k ($1 \le n \le 1\,000$, $0 \le m \le 100\,000$, $1 \le k \le n$) — the number of vertices and edges in the graph, and the number of vertices that are homes of the government.

The next line of input will contain k integers $c_1, c_2, \ldots, c_k$ ($1 \le c_i \le n$). These integers will be pairwise distinct and denote the nodes that are home to the governments in this world.

The following m lines of input will contain two integers u_i and v_i ($1 \le u_i, v_i \le n$). This denotes an undirected edge between nodes u_i and v_i.

It is guaranteed that the graph described by the input is stable.

Output

Output a single integer, the maximum number of edges Hongcow can add to the graph while keeping it stable.

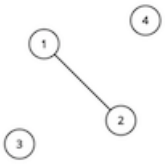
Examples

input
4 1 2 1 3 1 2
output
2

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

Note

For the first sample test, the graph looks like this:



Vertices 1 and 3 are special. The optimal solution is to connect vertex 4 to vertices 1 and 2. This adds a total of 2 edges. We cannot add any more edges, since vertices 1 and 3 cannot have any path between them.

Chocolate

Polycarpus likes giving presents to Paraskevi. He has bought two chocolate bars, each of them has the shape of a segmented rectangle. The first bar is $a_1 \times b_1$ segments large and the second one is $a_2 \times b_2$ segments large.

Polycarpus wants to give Paraskevi one of the bars at the lunch break and eat the other one himself. Besides, he wants to show that Polycarpus's mind and Paraskevi's beauty are equally matched, so the two bars must have the same number of squares.

To make the bars have the same number of squares, Polycarpus eats a little piece of chocolate each minute. Each minute he does the following:

- he either breaks one bar exactly in half (vertically **or** horizontally) and *eats exactly a half* of the bar,
- or he chips of exactly one third of a bar (vertically or horizontally) and *eats exactly a third* of the bar.

In the first case he is left with a *half*, of the bar and in the second case he is left with *two thirds* of the bar.

Both variants aren't always possible, and sometimes Polycarpus cannot chip off a half nor a third. For example, if the bar is 16×23, then Polycarpus can chip off a half, but not a third. If the bar is 20×18, then Polycarpus can chip off both a half and a third. If the bar is 5×7, then Polycarpus cannot chip off a half nor a third.

What is the minimum number of minutes Polycarpus needs to make two bars consist of the same number of squares? Find not only the required minimum number of minutes, but also the possible sizes of the bars after the process.

Input

The first line of the input contains integers a_1, b_1 ($1 \le a_1, b_1 \le 10^9$) — the initial sizes of the first chocolate bar. The second line of the input contains integers a_2, b_2 ($1 \le a_2, b_2 \le 10^9$) — the initial sizes of the second bar.

You can use the data of type `int64` (in Pascal), `long long` (in C++), `long` (in Java) to process large integers (exceeding $2^{31} - 1$).

Output

In the first line print m — the sought minimum number of minutes. In the second and third line print the possible sizes of the bars after they are leveled in m minutes. Print the sizes using the format identical to the input format. Print the sizes (the numbers in the printed pairs) in any order. The second line must correspond to the first bar and the third line must correspond to the second bar. If there are multiple solutions, print any of them.

If there is no solution, print a single line with integer `-1`.

Examples

input
2 6 2 3

output
1 1 6 2 3

input
36 5 10 16

output
3 16 5 5 16

input
3 5 2 1

output
-1