Traveling Shoemaker Problem

Once upon a time there was a very peaceful country named Nlogonia. Back then, Poly the Shoemaker could come to the country and travel freely from city to city doing his job without any harassment. This task was very easy, as every city in Nlogonia had a direct road to every other city in the country. He could then easily travel the whole country visiting each city exactly once and fixing everybody's shoes.

But not anymore. The times have changed and war has come to Nlogonia. The age when people could travel freely is over.

Confederations identified by colors were formed among the cities all over the country, and now each city belongs to at least one and at most two confederations. When trying to enter a city, you must give to the border officer a ticket from one of the confederations this city belongs to. When leaving the city, you receive a ticket from the other confederation the city belongs to (i.e. different from the one you gave when entering) or from the same confederation if the city only belongs to one.

As Poly the Shoemaker is a long time friend of Nlogonia, he is allowed to choose a ticket and a city he wants to enter as the first city in the country, but after that he must obey the confederations rules. He wants to do the same routine he did before, visiting each city exactly once in Nlogonia, but now it's not easy for him to do this, even though he can choose where to start his journey.

For example, suppose there are four cities, labeled from 0 to 3. City 0 belongs to confederations *red* and *green*; city 1 belongs only to *red*; city 2 belongs to *green* and *yellow*; and city 3 belongs to *blue* and *red*. If Poly the Shoemaker chooses to start at city 0, he can enter it carrying either the *red* or the *green* ticket and leave receiving the other. Should he choose the *red* ticket, he will leave with a *green* ticket, and then there is only city 2 he can travel to. When leaving city 2 he receives the *yellow* ticket and now can't go anywhere else. If he had chosen the *green* ticket as the first he would receive the *red* one when leaving, and then he could travel to cities 1 or 3. If he chooses city 3, when leaving he will receive the *blue* ticket and again can't go anywhere else. If he chooses city 1, he receives the *red* ticket again when leaving (city 1 belongs only to the *red* confederation) and can only travel to city 3 and will never get to city 2. Thus, it is not possible to visit each city exactly once starting at city 0. It is possible, however, starting at city 2 with the *yellow* ticket, leaving the city with the *green* ticket, then visiting city 0, leaving with *red* ticket, then visiting city 1, leaving with *red* ticket again and, at last, visiting city 3.

As you can see, it got really difficult for Poly the Shoemaker to accomplish the task, so he asks you to help him. He wants to know if it's possible to choose a city to start such that he can travel all cities from Nlogonia exactly once.

Can you help Poly the Shoemaker?

Input

The input contains several test cases. The first line of a test case contains two integers N and C, separated by one space, indicating respectively the number of cities ($1 \leq N \leq 500$) and confederations ($1 \leq C \leq 100$) in the country. Each of the next C lines describes a confederation. It starts with one integer K ($0 \leq K \leq N$) and then K integers representing the cities which belong to

this confederation. All integers are separated by single spaces and cities are numbered from 0 to *N* -1. Each city will appear at least once and at most twice and no city will be repeated on the same confederation.

The end of input is indicated by a line containing two zeroes separated by a single space.

Output

For each test case in the input, your program must print a single line, containing the integer -1 if it's not possible to match the requirements or one integer representing the city where Poly the Shoemaker can start his journey. If there are multiple correct answers, print the smallest one.

Example

Input:

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

Output:

```
2
-1
1
```