2007 Canadian Computing Competition
Day 2, Question 1

Problem D – Gerrymandering

Input file: `gerry.in`
Output file: to standard output
Source file: `gerry.{c, cpp, pas}`

Politicians like to get elected. They will do just about anything to get elected. Including changing the rules of the voting: who can vote, where you can vote, when you can vote, etc. One very common practice is called *gerrymandering*, where the boundaries of "ridings" are redrawn to favour a particular candidate (the one doing the redrawing, of course).

Your task is to help determine how to do some simple, linear gerrymandering.

You will be given the information about N ridings ($2 \leq N \leq 100000$) where there are candidates from p ($2 \leq p \leq 10$) different parties. These N ridings are linear, in the sense that they are side-by-side; there are two ridings (on the ends) that have only one adjacent riding, with the rest of the ridings having two adjacent ridings. A picture is shown below for $N = 4$ and $p = 2$ (which is also the sample data):

	Riding 1	Riding 2	Riding 3	Riding 4
Votes for Party 1	1	4	1	6
Votes for Party 2	5	3	2	1

Note that Riding 1 and Riding 2 are adjacent, Riding 2 and 3 are adjacent, Riding 3 and 4 are adjacent. No other ridings are adjacent.

You have some financial backing that will let you bribe the people in charge of setting the boundaries of ridings: in particular, there is a fixed rate to merge two adjacent boundaries. When you merge two ridings, the votes of the ridings merge together, in the sense that the number of votes of party 1 is the sum of the votes of party 1 in each riding, and likewise for all other parties.

Your task is to merge the minimum number of regions such that the first party (your party!) has a majority of the ridings. Note that to win a riding, the party must have more votes than any other party in that riding. Also note that to have a majority of ridings, if there are Q ridings (where $Q \leq N$), then your party has won at least $\lfloor \frac{Q}{2} \rfloor + 1$ of the ridings.

Input

The first line of input will consist of the integer N. The second line of input will consist of the integer p. The next N lines will each contain p non-negative integers (where each integer is at most 10000), separated by one space character. Specifically, the p integers on each line are v_1 v_2 ... v_p where v_1 is the number of votes that party 1 will receive in this riding, v_2 is the number of votes that party 2 will receive in this riding, etc.

You may also assume that the total number of voters is less than 2 billion.

Output

One line, consisting of an integer, which gives the minimum number of ridings that need to be merged in order for the first party to win a majority of ridings. If the first party cannot win, even with any number of mergers, output -1.

Sample Input 1

```
4
2
1 5
4 3
1 2
6 1
```

Output for Sample Input 1

```
1
```

Sample Input 2

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

Output for Sample Input 2

```
-1
```

2007 Canadian Computing Competition
Day 2, Question 2

Problem E – Particle Catcher

Input file: none (see below)
Output: none (see below)
Source file: `particle.{c, cpp, pas}`

You may have heard that you cannot observe *both* the speed and location of a subatomic particle, simultaneously. At least, not until today, since computer scientists can ignore physicists whenever we want.

Your task is to determine both the speed and location of a particle that is moving around a nucleus. Of course, we will assume that the particle is moving in a circular orbit and moving at a constant velocity.

You can use your high-tech measuring device (which is, in fact, the synthesis of a spoon and a Commodore-64) to perform the following 3 different queries:

- `GetSize()` – this method returns the integer representing the number of different positions in one orbit of the particle. You may assume that if this function returns N, the positions are labelled $0, ..., N-1$. You may assume $2 \leq N \leq 100000$.

- `Look(a,b)` – returns true if the point is in the range from $a..b$ (where $0 \leq a \leq b \leq N-1$) at this moment and false otherwise. This method will be called repeatedly, but subsequent calls cause the particle to move v positions from its current position. In other words, the particle is moving at v units per query. The first call to this method will query the initial position of the point (i.e., the `GetSize()` method does not advance the point).

- `Guess(v,p)` – will record the final velocity `v` and final position `p` (where 0 ≤`v, p`≤ $N-1$). There is exactly one call to this method, which will terminate the high-tech measuring device (i.e., no other calls may be made) and the high-tech device will output a statement indicating the success or failure of the guess (whether the point is at position p and velocity v when the call to `Guess` is made. Note that the particle is v units away from where it was at the last query statement.

Testing Your Program

The query module reads the description of a test case from the file `query.in`. This file contains the values N (on the first line), v (on the second line) and p (on the third line). You can modify this file by hand in order to test your program. Of course, your program must not read directly from this file, since we will change what file the query module reads from (and then you will get zero).

The query module outputs a file named `query.log` which describes the calls that your program made to the module.

Instructions for Pascal Programmers

To access the library, your program must contain the statement `uses query;`

For your information, here are the function and procedure declarations:

```
function GetN:integer;
function Look(a:integer, b:integer):integer;
procedure Guess(a:integer, b:integer):integer;
```

Instructions for C/C++ Programmers

Use the instruction `#include "query.h"` in your source code. You should have the file `query.o` in your home directory (it was put there while you slept). To compile your solution, type in (at the command prompt):

```
g++ -o particle query.o particle.cpp
```

Make sure the library is in the same directory as your source code.

To execute your program, type in (at the command prompt):

```
./particle
```

Additional Information

Your program is not allowed to read from or write to any files.

2007 Canadian Computing Competition
Day 2, Question 3

Problem F – Road Construction

Input file: `road.in`
Output file: to standard output
Source file: `road.{c, cpp, pas}`

It's almost summer time, and that means that it's almost summer construction time! This year, the good people who are in charge of the roads on the tropical island paradise of Remote Island would like to repair and upgrade the various roads that lead between the various tourist attractions on the island.

The roads themselves are also rather interesting. Due to the strange customs of the island, the roads are arranged so that they never meet at intersections, but rather pass over or under each other using bridges and tunnels. In this way, each road runs between two specific tourist attractions, so that the tourists do not become irreparably lost.

Unfortunately, given the nature of the repairs and upgrades needed on each road, when the construction company works on a particular road, it is unusable in either direction. This could cause a problem if it becomes impossible to travel between two tourist attractions, even if the construction company works on only one road at any particular time.

So, the Road Department of Remote Island has decided to call upon your consulting services to help remedy this problem. It has been decided that new roads will have to be built between the various attractions in such a way that in the final configuration, if any one road is undergoing construction, it would still be possible to travel between any two tourist attractions using the remaining roads. Your task is to find the minimum number of new roads necessary.

Input

The first line of input will consist of positive integers n and r, separated by a space, where $3 \leq n \leq 1000$ is the number of tourist attractions on the island, and $2 \leq r \leq 1000$ is the number of roads. The tourist attractions are conveniently labelled from 1 to n.

Each of the following r lines will consist of two integers, v and w, separated by a space, indicating that a road exists between the attractions labelled v and w. Note that you may travel in either direction down each road, and any pair of tourist attractions will have at most one road directly between them. Also, you are assured that in the current configuration, it is possible to travel between any two tourist attractions.

Output

One line, consisting of an integer, which gives the minimum number of roads that we need to add.

Sample Input 1

```
10 12
1 2
1 3
1 4
2 5
2 6
5 6
3 7
3 8
7 8
4 9
4 10
9 10
```

Output for Sample Input 1

```
2
```

Sample Input 2

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

Output for Sample Input 2

```
0
```