Árboles y Grafos, 2024-2

Para entregar el martes 24 de septiembre de 2024
A las 23:59 en la arena de programación

Instrucciones para la entrega

- Para esta tarea y todas las tareas futuras en la arena de programación, la entrega de soluciones es *individual*. Por favor escriba claramente su nombre, código de estudiante y sección en cada archivo de código (a modo de comentario). Adicionalmente, cite cualquier fuente de información que utilizó. Los códigos fuente que suba a la arena de programación deben der de su completa autoría.
- En cada problema debe leer los datos de entrada de la forma en la que se indica en el enunciado y debe imprimir los resultados con el formato allí indicado. No debe agregar mensajes ni agregar o eliminar datos en el proceso de lectura. La omisión de esta indicación puede generar que su programa no sea aceptado en la arena de programación.
- Puede resolver los ejercicios en C/C++ y Python. Sin embargo, deben haber por los menos soluciones a dos problemas en cada lenguaje.
- Debe enviar sus soluciones a través de la arena. Antes de subir sus soluciones asegurese de realizar pruebas con los casos de pruebas proporcionados para verificar que el programa finalice y no se quede en un ciclo infinito.
- El primer criterío de evaluación será la aceptación del problema en la arena cumpliendo los requisitos indicados en los enunciados de los ejercicios y en este documento. El segundo criterio de evaluación será la complejidad computacional de la solución y el uso de los temas vistos en clase. Es necesario incluir en la cabecera del archivo comentarios que expliquen la complejidad de la solución del problema para cada caso.

  En adición a lo anterior, para efectos de la calificación se tendrán en cuenta aspectos de estilo como no usar `break` ni `continue` y que las funciones deben tener únicamente una instrucción `return` que debe estar en la última línea.

Problemas prácticos

Hay cuatro problemas prácticos cuyos enunciados aparecen a partir de la siguiente página. Debe resolver 4 de los 5 problemas. El problema restante se calificará como bonificación.

A - Problem A

Source file name: `capital.cpp`
Time limit: 2 seconds

There are N cities in Flatland connected with M unidirectional roads. The cities are numbered from 1 to N. The Flat Circle of Flatland (FCF) wants to set up a new capital city for his kingdom. For security reasons, *the capital must be reachable from all other cities* of Flatland. FCF needs the list of *all* candidate cities. You are the chief programmer at FACM (Flat Association for Computing Machinery) responsible for providing the list to FCF as soon as possible.

Input

The first line of each test case contains two integers: $1 \leq N \leq 100000$ and $1 \leq M \leq 200000$. Each of the following M lines contains two integers $1 \leq A, B \leq N$ denoting a road from A to B.

The input must be read from standard input.

Output

The output for each test case contains an integer denoting the number of candidate cities followed by the list of candidate cities in increasing order.

The output must be written to standard output.

Sample Input	**Sample Output**

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

B - Problem B

Source file name: `equivalences.cpp`
Time limit: 1 second

Consider the following exercise, found in a generic linear algebra textbook.

Let A be an $n \times n$ matrix. Prove that the following statements are equivalent:

(a) A is invertible.

(b) $Ax = b$ has exactly one solution for every $n \times 1$ matrix b.

(c) $Ax = b$ is consistent for every $n \times 1$ matrix b.

(d) $Ax = 0$ has only the trivial solution $x = 0$.

The typical way to solve such an exercise is to show a series of implications. For instance, one can proceed by showing that (a) implies (b), that (b) implies (c), that (c) implies (d), and finally that (d) implies (a). These four implications show that the four statements are equivalent.

Another way would be to show that (a) is equivalent to (b) (by proving that (a) implies (b) and that (b) implies (a)), that (b) is equivalent to (c), and that (c) is equivalent to (d). However, this way requires proving six implications, which is clearly a lot more work than just proving four implications!

I have been given some similar tasks, and have already started proving some implications. Now I wonder, how many more implications do I have to prove? Can you help me determine this?

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line containing two integers n ($1 \leq n \leq 20000$) and m ($0 \leq m \leq 50000$): the number of statements and the number of implications that have already been proved.
- m lines with two integers s_1 and s_2 ($1 \leq s1, s2 \leq n$ and $s_1 \neq s_2$) each, indicating that it has been proved that statement s_1 implies statement s_2.

The input must be read from standard input.

Output

Per testcase:

- One line with the minimum number of additional implications that need to be proved in order to prove that all statements are equivalent.

The output must be written to standard output.

Sample Input	Sample Output

2
4 0
3 2
1 2
1 3
	4
	2

C - Problem C

Source file name: `graffs.cpp`
Time limit: 5 seconds

King Graff, the ruler of the land of Feerie, has a problem — his nation is under attack! Luckily, he has an army at his disposal, composed of a whopping S soldiers (where $S = 2$).

Feerie consists of N ($1 \leq N \leq 100000$) towns (numbered $1 \ldots N$), and M ($1 \leq M \leq 500000$) roads. The ith road runs between distinct towns A_i and B_i, in both directions. No pair of towns is directly connected by more than one road, but every pair of towns is connected by at least one path of connected roads. King Graff would like to position his two soldiers in two different towns to prepare for the impending assault — however, since he's not much of a strategist, he'll choose the towns at complete random.

Graff's only real concern is with his enemies using a divide-and-conquer strategy. His soldiers will be susceptible to this type of attack if there exists any single road that, if blocked, will prevent them from reaching each other by any system of connected roads. As the royal computer scientist, your job is to determine the probability that King Graff will be defeated.

Input

Each test case is described using several lines. The first line contains two integers N and M, the number of town and rodas. Next M lines describes each road by means two integers A_i and B_i.

The input must be read from standard input.

Output

For each test case outputs a real number (rounded to 5 decimal places), the probability that the two towns chosen by Graff can be disconnected by the removal of any single road.

The output must be written to standard output.

Sample Input	**Sample Output**

```
4 4
1 2
1 3
2 4
4 1
```
	`0.50000`

D - Problem D

Source file name: `mazes.cpp`
Time limit: 1 second

The Queen of Nlogonia is a fan of mazes, and therefore the queendom's architects built several mazes around the Queen's palace. Every maze built for the Queen is made of rooms connected by corridors. Each corridor connects a different pair of distinct rooms and can be transversed in both directions.

The Queen loves to stroll through a maze's rooms and corridors in the late afternoon. Her servants choose a different challenge for every day, that consists of finding a simple path from a start room to an end room in a maze. A simple path is a sequence of distinct rooms such that each pair of consecutive rooms in the sequence is connected by a corridor. In this case the first room of the sequence must be the start room, and the last room of the sequence must be the end room. The Queen thinks that a challenge is good when, among the routes from the start room to the end room, exactly one of them is a simple path. Can you help the Queen's servants to choose a challenge that pleases the Queen?

For doing so, write a program that given the description of a maze and a list of queries defining the start and end rooms, determines for each query whether that choice of rooms is a good challenge or not.

Input

Each test case is described using several lines. The first line contains three integers R, C and Q representing respectively the number of rooms in a maze ($2 \leq R \leq 10^4$), the number of corridors ($1 \leq C \leq 10^5$), and the number of queries ($1 \leq Q \leq 1000$). Rooms are identified by different integers from 1 to R. Each of the next C lines describes a corridor using two distinct integers A and B, indicating that there is a corridor connecting rooms A and B ($1 \leq A < B \leq R$). After that, each of the next Q lines describes a query using two distinct integers S and T indicating respectively the start and end rooms of a challenge ($1 \leq S < T \leq R$). You may assume that within each test case there is at most one corridor connecting each pair of rooms, and no two queries are the same.

The last test case is followed by a line containing three zeros.

The input must be read from standard input.

Output

For each test case output $Q + 1$ lines. In the i-th line write the answer to the i-th query. If the rooms make a good challenge, then write the character '`Y`' (uppercase). Otherwise write the character '`N`' (uppercase). Print a line containing a single character '`-`' (hyphen) after each test case.

The output must be written to standard output.

Sample Input

```
6 5 3
1 2
2 3
2 4
2 5
4 5
1 3
1 5
2 6
4 2 3
1 2
2 3
1 4
1 3
1 2
0 0 0
```

Sample Output

```
Y
N
N
-
N
Y
Y
```

E - Problem E

Source file name: `water.cpp`
Time limit: 1 second

Technopark is a huge industrial park surrounded by small residential towns, inhabited by its workers. Technopark owns a dam, which provides electricity and clean drinking water to its facilities and some residential towns. The remaining towns get their water from dug wells.

Recently, some workers and their families have gotten a disease caused by bacteria found in some wells, so it has been decided to extend the water supply network to take dam water from Technopark to every residential town. Some pipelines connecting pairs of towns already exist but additional pipelines could possibly be needed. Pipelines are unidirectional, i.e., they only allow water transportation in one direction, and every pipeline connects exactly two towns.

Your task is to compute the minimum number of pipelines that must be installed to take dam water to every residential town.

Input

The input contains several test cases. The first line of each test case has two blank-separated integers N and M, where N is the number of residential towns ($1 \leq N \leq 1000$) and M is the number of existing pipelines ($0 \leq M \leq 100000$). Towns are numbered from 0 to N, being Technopark town 0. Each of the next M lines contains two blank-separated integers a and b ($0 \leq a \leq N$, $0 \leq b \leq N$) indicating that there is a pipeline taking water from town a to town b.

The input must be read from standard input.

Output

For each test case, print the minimum number of pipelines that must be built to take dam water to every residential town.

The output must be written to standard output.

Sample Input	**Sample Output**

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