Árboles y Grafos, 2025-1

Para entregar el domingo 23 de marzo de 2025
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 es correcto, que finaliza 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 cinco problemas prácticos cuyos enunciados aparecen a partir de la siguiente página. El problema restante se calificará con **0.5 décimas de bonificación**. En caso de que la nota total de la tarea supere 5.0 el excedente será puesto en otra tarea del curso.

A - Problem A

Source file name: `critical.cpp`
Time limit: x seconds

In the Amicable, Great, Respected, and Advanced Empire of Zlatan, known as the AGRA Empire, train stations are connected by tracks where trains travel in both directions. Cities correspond to groups of stations that are interconnected, meaning that from any station, one can reach any other. Each station has an estimated daily passenger capacity.

Some stations are extremely important because if they were unavailable, the stations in the cities could become disconnected. These stations are called *critical* stations. For this reason, Zlatan wants to avoid inconvenients with critical stations, but ensuring their security would be very expensive, and unfortunately, the empire's treasury is not in the best condition at the moment. Correspondingly, Zlatan has decided to prioritize the security of a station such that the number of stations that would become disconnected due to its failure is maximized. There may be multiple critical stations with the maximum number of disconnected stations, and in such cases, the critical station to be prioritized will be the one with the highest estimated daily passenger count. If there is still a tie, the station with the lowest numerical identifier will be prioritized.

Your task is to help Zlatan determine which critical station should be prioritized and how many stations would become disconnected if that station were to fail.

Input

There will be multiple test cases in the input. For each test case, there will be a line with two numbers, n and m ($1 \leq n \leq 10000, 1 \leq m \leq 100000$), corresponding to the number of stations and the number of connections between stations in the AGRA Empire.

The next line will contain n positive integers v_i ($0 \leq i < n, 0 \leq v_i \leq 10000$) representing the estimated daily capacity of each station. Then, there will be m lines, each containing two numbers, u and v, indicating that there is a track between station u and station v.

The input must be read from standard input.

Output

For each test case, if there are critical stations in the empire then print a line with two numbers corresponding to the identifier of the critical station that will be prioritized and the number of stations that would become disconnected if this station were to fail. Otherwise, print `-1`.

The output must be written to standard output.

Sample Input

```
15 19
100 300 500 1000 200 2000 2200 250 400 2199 2201 330 50 5000 300
1 2
1 3
2 3
1 6
6 3
6 7
7 2
3 10
10 4
9 4
9 10
5 9
5 12
8 12
11 12
11 8
8 5
0 13
14 13
```

Sample Output

```
10 11
```

B - Problem B

Source file name: `dominos.cpp`
Time limit: x seconds

Dominos are lots of fun. Children like to stand the tiles on their side in long lines. When one domino falls, it knocks down the next one, which knocks down the one after that, all the way down the line. However, sometimes a domino fails to knock the next one down. In that case, we have to knock it down by hand to get the dominos falling again.

Your task is to determine, given the layout of some domino tiles, the minimum number of dominos that must be knocked down by hand in order for all of the dominos to fall.

Input

The first line of input contains one integer specifying the number of test cases to follow. Each test case begins with a line containing two integers, each no larger than 100000. The first integer n is the number of domino tiles and the second integer m is the number of lines to follow in the test case. The domino tiles are numbered from 1 to n.

Each of the following lines contains two integers x and y indicating that if domino number x falls, it will cause domino number y to fall as well.

The input must be read from standard input.

Output

For each test case, output a line containing one integer, the minimum number of dominos that must be knocked over by hand in order for all the dominos to fall.

The output must be written to standard output.

Sample Input	**Sample Output**
2 3 2 1 2 2 3 6 2 2 1 4 1	1 5

C - Problem C

Source file name: `impossible.cpp`
Time limit: x seconds

At your home university there are N research groups and M procedures to exchange information between them. Some procedures enable information to be sent from a specific group to another group. Other procedures are designed to share information between two or more groups (in either direction for each pair of them).

To be more precise, the research system of your university uses the following notation to identify the information sharing procedures:

- A simple procedure that enables information to be sent from the group I to the group J is specified as

$$1 \quad I \quad J$$

- A complex procedure that enables bi-directional information sharing between the k groups $I_1, I_2, ..., I_k$ is described as

$$k \quad I_1 \quad I_2 \quad \cdots \quad I_k$$

The president of your home university wants to make sure that the procedures to exchange information between research groups are sufficiently complete: he wants to make sure that any research group is able to send information to any other group (regardless of whether it is done directly or using intermediaries).

Please, help your president!

Input

The input consists of several test cases. The first line of a case contains two integers N and M indicating, respectively, the number of research groups ($2 \leq N \leq 50000$) and the number of information sharing procedures ($1 \leq M \leq 1000$). Each of the next M lines identifies one simple or one complex procedure, following the above-described notation. Research groups are represented with the integers $1, 2, ..., N$. You can assume that each complex procedure has at most 1 000 research groups. Single blanks are used to separate any pair of consecutive numbers.

The input must be read from standard input.

Output

For each test case output a single line with `YES` if the given set of procedures is sufficiently complete and `NO` otherwise.

The output must be written to standard output.

Sample Input	**Sample Output**
3 3 1 1 2 1 2 3 1 1 3 4 3 1 1 2 1 2 3 3 1 3 4	NO YES

D - Problem D

Source file name: `tingo-tingo-tango.cpp`
Time limit: x seconds

The game of *tingo-tingo-tango* is a very popular game at parties and gatherings, enjoyed by both children and even some adults. In the original game, an object is passed among the participants while a blindfolded person continuously says the word *tingo* until, at some point, she/he says *tango*. The person holding the object at that moment is eliminated from the game.

Zlatan is a child in an adult's body and frequently throws parties, inviting his little friends, including Ronaldinho and Cavani. To be original, Zlatan invented a new way to play tingo-tingo-tango. In this new version, groups of people of arbitrary sizes participate, and the number of people in each group is not necessarily the same. Thus, one group may have only one person, while others may consist of many people. Each person can pass the object to certain people within their own group, and some members of a group can pass the object to people in other groups. Whenever there is a sequence of people that allows the object to be passed from person a to person b and vice versa, a and b will belong to the same group.

As usual, a blindfolded person will keep saying *tingo* until she/he eventually says *tango*. When this happens, the group to which the person holding the object belongs is eliminated.

Zlatan is a bit egocentric and does not like losing, so he always makes sure to be in the group where the game starts in order to, according to him, minimize his chances of losing. Additionally, in each round, he chooses a person whose group he wants to be eliminated. Zlatan would like to know what is the minimum amount of groups the object must pass through before reaching the group of the chosen person.

Given the configuration of any round of the game—that is, a description of who can pass the object to whom—your task is to help Zlatan achieve his peculiar goal.

Input

The first line of each test case will contain two numbers, n and m, which correspond to the number of people in the game and the number of possible passes of the game object between two people. The people in the game will be represented by numbers between 1 and n.

The next line will contain two numbers, z and x, which represent the number identifying Zlatan and the number identifying the person chosen by Zlatan.

Next, there will be m lines, each containing two numbers, p and q. Each of these lines indicates that person p can pass the object to person q.

The input will end with a test case where $n = m = 0$, which should not be processed.

The input must be read from standard input.

Output

For each test case, print a message as follows:

`Zlatan will need the object to pass through t groups.`

Where `t` is the number of groups the object must pass through before reaching the group of the chosen person.

The output must be written to standard output.

Sample Input

```
13 18
2 12
8 6
1 2
12 10
7 8
4 1
9 10
5 6
6 7
12 13
3 9
3 4
10 11
3 5
11 12
10 12
2 3
5 2
13 10
0 0
```

Sample Output

```
Zlatan will need the object to pass through 2 groups.
```

E - Problem E

Source file name: `tourism.cpp`
Time limit: x seconds

In the Amicable, Great, Respected, and Advanced Empire of Zlatan, known as the AGRA Empire, cities are connected by modern highways that run in a single direction. The political organization of the empire dictates that any pair of cities a and b, where there is a sequence of highways allowing travel from a to b and vice versa, must belong to the same state. The highways that connect cities within the same state are known as *intra-state* highways. Similarly, some highways allow travel from a city c to a city d such that c and d do not belong to the same state. These types of highways are known as *interstate* highways.

Each city in the empire has a founding date, and highways between cities were built over the years after the cities were founded. Each state has a capital city, which is the oldest city in that state. In cases where multiple cities in a state share the same founding date, the city with the lexicographically smaller name will be chosen as the capital.

As part of a tourism campaign called "Come to Zlatan's Homeland", the goal is to allow tourists to visit as many cities in the empire as possible. However, it is not feasible for tourists to visit every city, so Emperor Zlatan wants them to focus on visiting the capital cities.

This plan would be quite interesting, but Emperor Zlatan, known for his obsessive rigidity, insists that the capital cities be visited in a specific order. Specifically, if there are highways allowing travel from a city x to a city y, then x must be visited before y. However, if there is no sequence of highways connecting x to y or viceversa, then the city with the earlier founding date must come first. If both cities have the same founding date, the city with the lexicographically smaller name will come first.

You have been hired by Zlatan to determine the order in which the capital cities should be visited.

Input

The first line of the input will contain a number t, representing the number of test cases to be processed.

For each test case, a line with the numbers n and m will be read, corresponding to the number of cities in the empire and the number of highways. In the next n lines, the names c_i of each city and their founding year a_i will be provided, for $0 \leq i < n$. The name c_i is a string without spaces. Finally, the next m lines will contain the description of each highway. This description will consist of the name c_1 of the origin city of the highway followed by the name of the destination city c_2.

The input must be read from standard input.

Output

For each test case, print one line with the names of cities in the order to be visited according to the Zlatan's requirements.

The output must be written to standard output.

Sample Input

```
1
16 24
Bb 1855
Ffffff 1985
Aaaa 1900
Hhhhh 1960
Ccccc 1820
Kkkk 1940
Qqq 1820
Dd 1920
Ggg 1960
Mmmmmmmm 1980
Eee 1980
Ii 1950
Jjjjjjjj 1940
Ll 1980
N 1970
Pppppppp 1800
Ggg Hhhhh
Pppppppp Eee
Mmmmmmm N
Ii Jjjjjjj
Hhhhh Ffffff
Ggg Ii
N Ll
Bb Dd
Bb Aaaa
Ccccc Aaaa
Jjjjjjj Kkkk
Bb Ccccc
Aaaa Bb
Ll Mmmmmmm
Pppppppp Qqq
Eee Ffffff
Kkkk Ii
N Dd
Dd Eee
Mmmmmmm Ll
Ffffff Ggg
Eee Hhhhh
Hhhhh Eee
Qqq Pppppppp
```

Sample Output

```
Pppppppp Ccccc N Dd Ggg Jjjjjjj
```