Árboles y Grafos, 2024-2

Para entregar el lunes 2 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. Además, 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.

A - Problem A

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

Eloy is a hard worker man, however, he is constantly bullied by his superiors, molested by this, one day he was wondering in what "rank" you are, so you can bully the people with lower ranks, also to discover who can really bully Eloy!.

Now, given the number of employees and the number of relations between them, Eloy need you to output the "rank" which employee is in, being 1 the "boss" (not bullied by anybody) and the employee who are in these ranks.

Input

There will be an integer T denoting the number of test cases, then, T test cases will follow. Each test case starts with two integers N ($1 \leq N \leq 1000$) and R ($1 \leq R \leq 10000$), the number of employees and the number of relations between them. The next R lines consists of two integers R_1 and R_2, meaning that "employee R_1 is lower than employee R_2's rank".

The input must be read from standard input.

Output

You will output for each test case the string `''Scenario #i:''` where `i` is the test case you are analyzing, after that, you will print N lines, for each line you will output the rank of the employee and the employee itself, if there is the same rank for several employees, then output them lexicographically ordered (the first is the lower.)

The output must be written to standard output.

Sample Input	**Sample Output**
2 5 6 2 0 2 4 1 4 1 2 3 2 4 0 5 4 1 0 2 0 3 2 4 2	Scenario #1: 1 0 2 4 3 2 4 1 4 3 Scenario #2: 1 0 2 1 2 2 3 3 3 4

B - Problem B

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

Isaringa and Karina are participating in the Incredible Calrare Programming Contest (ICPC), in which they must solve a set of problems. If they succeed, they will be able to enjoy a nice coffee with Calrare and talk about cycle invariants. Calrare is an evil being and for this competition he has selected problems from two categories, mathematical problems (Level 5 and up) and DP problems (Level 8 and up). As if this were not enough, Calrare has decided that some problems depend on others, each problem can depend on one or more other problems, and there are no directed circular dependencies between problems. A problem can only begin to be solved if all the problems it depends on have already been solved.

Isaringa is an expert at solving mathematical problems and gets angry easily because she is not very patient, while Karina has an incredible ability for Dynamic Programming (DP) problems and is very calm. Both are able to solve problems only in the subject they are proficient in. During the competition they have only one copy of the problem set so while one of them is reading and solving the problems the other one must wait. Karina is the one who always starts reading the problem set and when she finds mathematical problems then she can call Isaringa to solve them, but Karina must be careful because Isaringa can explode with anger and destroy the copies.

Find the minimal number of times Karina has to call Isaringa to solve all the problems without Karina's well-being being in danger.

Input

The first line contains a single integer C that stands for the number of test cases. Each test case starts with a line containing two space-separated integers N ($1 \leq N \leq 10^5$) — the total number of tasks given, and M ($0 \leq M \leq 10^5$) — the total number of dependencies between problems.

The next line contains N space-separated integers $E_i \in \{0, 1\}$. If $E_i = 0$, problem i is related to Dynamic Programming, otherwise it is mathematical problem.

The next M lines describe the dependencies between problems. Each line contains two space-separated integers P_1 and P_2 and means that problem P_1 depends on problem P_2 ($P_1 \neq P_2$). Problems are indexed from 0 to $N - 1$. All M pairs (P_1, P_2) are distinct. It is guaranteed that there are no circular dependencies between tasks.

The input must be read from standard input.

Output

For each test case, output one line containing an integer — the minimal number of times Isaringa must be called.

Explanation: In the first test case, problem 1 and 3 are mathematical problmes and they can only be solved by Isaringa. The dependency graph is linear, so the problems must be solved in order $3- > 2- > 1- > 0$. Karina has to call Isaringa twice: first she calls her for problem 3, then Isaringa solves problem 2, then she calls her for task 1, and finally she solves problem 0.

In the second test case, problems 0, 1 and 2 can only be solved by Isaringa. Problems 1 and 2 have no dependencies, and problem 0 depends on problems 1 and 2, so all three problems 0, 1 and 2 can be assigned in one Isaringa call. After that problem 3 is solved by Karina.

The output must be written to standard output.

Sample Input	**Sample Output**
2 4 3 0 1 0 1 0 1 1 2 2 3 4 3 1 1 1 0 0 1 0 2 3 0	2 1

C - Problem C

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

One day Zlatan got a package in a Mercadolibre box. Zlatan knows that when the employees of Mercadolibre send a package directly from city A to city B, they mark the box with A B, or B A. Unfortunately, often it is impossible to send a package directly from the city of the sender to the city of the receiver, that's why the package is sent via some intermediate cities. Mercadolibre employees never send a package in such a way that the route of this package contains some city more than once. Zlatan is sure that the employees mark the boxes accurately.

There are n marks on the box of Zlatan's package. He understands that the possible routes of this box are only two. But there are a lot of marks, and Zlatan can't determine himself none of these routes. That's why he asks you to help him. Find one of the possible routes of the box.

Input

The first line contains integer n ($1 \le n \le 10^5$) — amount of marks on the box. Then there follow n lines with two integers each — description of the marks. Each marks is described with indexes of the cities between which a package is sent. The indexes of cities are integers from 1 to 10^9. Indexes of all the cities are different. Every time the package is sent from one city to another, exactly one mark is put on the box. It is guaranteed that the given marks correspond to some valid route from some city to some other city. The input is terminated when $n = 0$.

The input must be read from standard input.

Output

For each test case, output $n + 1$ numbers — indexes of cities in the route of the package starting in the city with the least index.

The output must be written to standard output.

Sample Input	**Sample Output**
2 1 100 100 2 3 3 1 100 2 3 2 0	1 100 2 1 3 2 100

D - Problem D

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

You might have noticed that English and Spanish are spoken in many areas all over the world. Now it would be nice to rank all languages according to the number of states where they are spoken.

You're given a map which shows the states and the languages where they are spoken. Look at the following map:

```
ttuuttdd
ttuuttdd
uuttuudd
uuttuudd
```

The map is read like this: Every letter stands for a language and states are defined as connected areas with the same letter. Two letters are "connected" if one is at left, at right, above or below the other one. So in the above map, there are three states where the language "t" is spoken, three where "u" is spoken and one state where people speak "d".

Your job is to determine the number of states for each language and print the results in a certain order.

Input

The first line contains the number of test cases N. Each test case consists of a line with two numbers H and W, which are the height and the width of the map. Then follow H lines with a string of W letters. Those letters will only be lowercase letters from `'a'` to `'z'`.

The input must be read from standard input.

Output

For each test case print `'World #n'`, where `n` is the number of the test case. After that print a line for each language that appears in the test case, which contains the language, a colon, a space and the number of states, where that language is spoken. These lines have to be ordered decreasingly by the number of states. If two languages are spoken in the same number of states, they have to appear alphabetically, which means language `'i'` comes before language `'q'`, for example.

The output must be written to standard output.

Sample Input	**Sample Output**
2 4 8 ttuuttdd ttuuttdd uuttuudd uuttuudd 9 9 bbbbbbbbb aaaaaaaab bbbbbbbab baaaaacab baccccab bacbbbcab baccccab baaaaaab bbbbbbbbb	World #1 t: 3 u: 3 d: 1 World #2 b: 2 a: 1 c: 1