Árboles y Grafos, 2025-1

Para entregar el domingo 18 de mayo 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.

A - Problem A

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

The airline company NCPC Airways has flights to and from n cities, numbered from 1 to n, around the entire world. However, they only have $n - 1$ different flights (operating in both directions), so in order to travel between any two cities you might have to take several flights. In fact, since the management has made sure that it's possible to travel between any pair of cities, there is exactly one set of flights a passenger have to take in order to travel between two cities (assuming you want to use the same airline).

Recently many of NCPC Airways frequent flyers have complained that they have had to change flights too often to get to their final destination. Since NCPC Airways doesn't want to loose their customers to other airline companies, but still keep the nice property of their flights, they have decided to cancel one of their current flights and replace it with another flight. Help the company by writing a program which finds the best flight to cancel and the best new flight to add so that the maximum number of flight changes a passenger might have to make when travelling between any pair of cities in which NCPC Airways operates is minimized.

The input will be constructed so that it is always possible to improve the maximum number of flight changes needed.

Input

The first line of the input file contains an integer N which denotes the total number of test cases. The description of each test case is given below:

The first line of input for each test case contains the integer n ($4 \leq n \leq 10000$), the number of cities NCPC Airways operates in. Then follow $n - 1$ lines specifying the flights. Each flight is given as a pair of cities a and b ($1 \leq a, b \leq n$).

The input must be read from standard input.

Output

For each test case print one line of output, which should contain an integer, the minimum number of flights needed to take when travelling between any pair of cities after changing one of the flights.

The output must be written to standard output.

Sample Input	**Sample Output**
2 4 1 2 2 3 3 4 14 1 2 1 8 2 3 2 4 8 9 8 10 8 11 4 5 4 6 4 7 10 12 10 13 13 14	2 5

B - Problem B

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

Korea has many tourist attractions. One of them is an archipelago (Dadohae in Korean), a cluster of small islands scattered in the southern and western coasts of Korea. The Korea Tourism Organization (KTO) plans to promote a new tour program on these islands. For this, The KTO wants to designate two or more islands of the archipelago as a tour belt.

There are n islands in the archipelago. The KTO focuses on the synergy effect created when some islands are selected for a tour belt. Synergy effects are assigned to several pairs of islands. A synergy effect $\mathrm{SE}(u, v)$ or $\mathrm{SE}(v, u)$ between two islands u and v is a positive integer which represents a value anticipated when both u and v are included in a tour belt. The KTO wants to select two or more islands for the tour belt so that the economic benefit of the tour belt is as high as possible.

To be precise, we define a connected graph $G = (V, E)$, where V is a set of n vertices and E is a set of m edges. Each vertex of V represents an island in the archipelago, and an edge (u, v) of E exists if a synergy effect $\mathrm{SE}(u, v)$ is defined between two distinct vertices (islands) u and v of V. Let A be a subset consisting of at least two vertices in V. An edge (u, v) is an *inside edge* of A if both u and v are in A. An edge (u, v) is a *border edge* of A if one of u and v is in A and the other is not in A.

A vertex set B of a connected subgraph of G with $2 \le |B| \le n$ is called a *candidate* for the tour belt if the synergy effect of every inside edge of B is larger than the synergy effect of any border edge of B. A candiate will be chosen as the final tour belt by the KTO. There can be many possible candidates in G. Note that V itself is a candidate because there are no border edges. The graph in Figure 1(a) has three candidates $\{1, 2\}$, $\{3, 4\}$, and $\{1, 2, 3, 4\}$, but $\{2, 3, 4\}$ is not a candidate because there are inside edges whose synergy effects are not larger than those of some border edges. The graph in Figure 1(b) contains six candidates, $\{1, 2\}$, $\{3, 4\}$, $\{5, 6\}$, $\{7, 8\}$, $\{3, 4, 5, 6\}$, and $\{1, 2, 3, 4, 5, 6, 7, 8\}$. But $\{1, 2, 7, 8\}$ is not a candidate because it does not form a connected subgraph of G, i.e., there are no edges connecting $\{1, 2\}$ and $\{7, 8\}$.

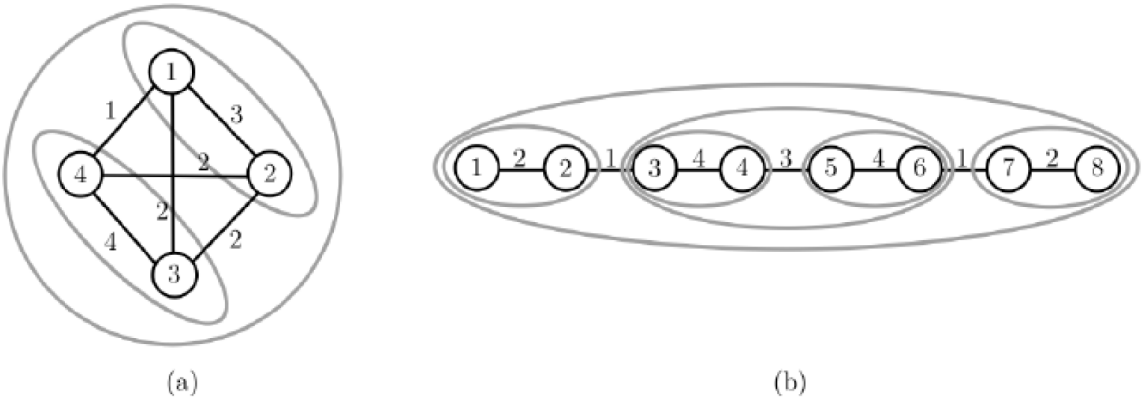

Figure 1. Graphs and their good subsets marked by gray ellipses.

The KTO will decide one candidate in G as the final tour belt. For this, the KTO asks you to find all candidates in G. You write a program to print the sum of the sizes of all candidates in a given graph G. For example, the graph in Figure 1(a) contains three candidates and the sum of their sizes is $2 + 2 + 4 = 8$, and the graph in Figure 1(b) contains six candidates and the sum of their sizes is $2 + 2 + 2 + 2 + 4 + 8 = 20$.

Input

Your program is to read input from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with a line containing two integers n ($2 \le n \le 5\,000$) and m ($1 \le m \le \frac{n(n-1)}{2}$), where n represents the number of vertices (islands) and m represents the number of edges of a connected graph G. Islands are numbered from 1 to n. In the following m lines, the synergy effects assigned to m edges are given; each line contains three integers, u, v, and k ($1 \le u \ne v \le n$, $1 \le k \le 10^5$), where k is the synergy effect between two distinct islands u and v,

i.e., $\text{SE}(u, v) = \text{SE}(v, u) = k$.

The input must be read from standard input.

Output

Your program is to write to standard output. Print exactly one line for each test case. Print the sum of the sizes of all candidates for a test case.

The output must be written to standard output.

Sample Input	**Sample Output**
2 4 6 1 2 3 2 3 2 4 3 4 1 4 1 2 4 2 1 3 2 8 7 1 2 2 2 3 1 3 4 4 4 5 3 5 6 4 6 7 1 7 8 2	8 20

C - Problem C

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

A war is being lead between two countries, *A* and *B*. As a loyal citizen of *C*, you decide to help your countrys espionage by attending the peace-talks taking place these days (incognito, of course). There are n people at the talks (not including you), but you do not know which person belongs to which country. You can see people talking to each other, and through observing their behaviour during their occasional one-to-one conversations, you can guess if they are friends or enemies. In fact what your country would need to know is whether certain pairs of people are from the same country, or they are enemies. You may receive such questions from Cs government even during the peace-talks, and you have to give replies on the basis of your observations so far. Fortunately nobody talks to you, as nobody pays attention to your humble appearance.

Now, more formally, consider a black box with the following operations:

`setFriends`(x,y) shows that x and y are from the same country

`setEnemies`(x,y) shows that x and y are from different countries

`areFriends`(x,y) returns true if you are sure that x and y are friends

`areEnemies`(x,y) returns true if you are sure that x and y are enemies

The first two operations should signal an error if they contradict with your former knowledge. The two relations 'friends' (denoted by $\sim$) and 'enemies' (denoted by $*$) have the following properties:

$\sim$ is an equivalence relation, i.e.

1. If $x \sim y$ and $y \sim z$ then $x \sim z$ (The friends of my friends are my friends as well).
2. If $x \sim y$ then $y \sim x$ (Friendship is mutual).
3. $x \sim x$ (Everyone is a friend of himself).

$*$ is symmetric and irreflexive

1. If $x * y$ then $y * x$ (Hatred is mutual).
2. Not $x * x$ (Nobody is an enemy of himself).

Also

1. If $x * y$ and $y * z$ then $x \sim z$ (A common enemy makes two people friends).
2. If $x \sim y$ and $y * z$ then $x * z$ (An enemy of a friend is an enemy).

Operations `setFriends`(x,y) and `setEnemies`(x,y) must preserve these properties.

Input

The first line contains a single integer, n, the number of people.

Each of the following lines contains a triple of integers, c x y, where c is the code of the operation:

c = 1, `setFriends`

c = 2, `setEnemies`

c = 3, `areFriends`

c = 4, `areEnemies`

and x and y are its parameters, which are integers in the range $[0, n)$, identifying two (different) people. The last line contains `'0 0 0'`.

All integers in the input file are separated by at least one space or line break. The only constraint is $n < 10000$, the number of operations is unconstrained.

The input must be read from standard input.

Output

For every `areFriends` and `areEnemies` operation write `'0'` (meaning no) or `'1'` (meaning yes) to the output. Also for every `setFriends` or `setEnemies` operation which contradicts with previous knowledge, output a `'-1'` to the output; note that such an operation should produce no other effect and execution should continue. A successful `setFriends` or `setEnemies` gives no output.

All integers in the output file must be separated by one line break.

The output must be written to standard output.

Sample Input	**Sample Output**
10 1 0 1 1 1 2 2 0 5 3 0 2 3 8 9 4 1 5 4 1 2 4 8 9 1 8 9 1 5 2 3 5 2 0 0 0	1 0 1 0 0 -1 0

D - Problem D

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

Since Zlatan got tired of living in Europe, he decided to move to the paradisiacal city of Cali and bought a mansion in the humble neighborhood of Ciudad Jardín. To keep himself entertained, he frequently hosts parties where he invites his little friends to play different games. At a recent party, Zlatan wanted them to play a game called *Around the World*. This game is played in teams with N people, who are arranged in a room. The game involves a balloon, which is initially held by the team leader. The team leader can be any player in the team. The objective is to pass the balloon from person to person so that every player holds the balloon at least once but and finally the balloon ends up back in the hands of the leader with the restriction that the leader receives the balloon from the same person to whom he initially gave it. Each player can directly pass the balloon to certain other players, and this process takes a number of seconds depending on the agility of both players. This time is known beforehand. The act of directly passing the balloon from person x to person y is called a touch between x and y.

Zlatan is very competitive and always wants his team to win. That's why he thinks of strategies to help his team complete the balloon-passing task in the shortest time possible. He realizes that there may be multiple ways to achieve the objective optimally, and that there may be pairs of people such that in every optimal solution, a touch between them must occur. Zlatan calls these critical touches.

Additionally, Zlatan concludes that there may be pairs of people such that a touch between them may occur in some optimal solutions, but not necessarily in all of them. Zlatan refers to these as quasi-critical touches.

Your mission, should you choose to accept it, is to help Zlatan identify the critical and quasi-critical touches within the team he belongs to.

Input

There will be multiple test cases. For each test case, lines will be read containing the values $name_1$, $name_2$, and t, where $name_1$ and $name_2$ are alphabetic strings without spaces, and t is a positive integer. Each of these lines represents a touch between person $name_1$ and person $name_2$ with time t. The end of each test case is indicated by a blank line, except for the last test case, which is not followed by a blank line.

Each case will contain a maximum of 100 people and a maximum of 500 touches.

The input must be read from standard input.

Output

For each test case, print 2 lines. The **first line** must contain the **critical touches**, separated by commas and sorted lexicographically. The **second line** must contain the **quasi-critical touches**, also separated by commas and sorted lexicographically. In both lines, in each touch, the **lexicographically smaller name must appear first**.

The output must be written to standard output.

Sample Input

```
Carlos Zlatan 1
Zlatan Angelica 1
Angelica Scarlett 2
Carlos Scarlett 2
Carlos Jennifer 3
Scarlett Jennifer 3
Zlatan Jennifer 6

Carlos Zlatan 1
Angelica Zlatan 1
Scarlett Angelica 1
Scarlett Carlos 1
```

Sample Output

```
Angelica Zlatan,Carlos Zlatan
Angelica Scarlett,Carlos Jennifer,Carlos Scarlett,Jennifer Scarlett

Angelica Scarlett,Angelica Zlatan,Carlos Scarlett,Carlos Zlatan
```

E - Problem E

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

In the Amicable, Great, Respected, and Advanced Empire of Zlatan, known as the AGRA Empire, there are n cities identified with numbers from 0 to $n - 1$, connected by roads where vehicles travel in only one direction. Due to political decisions made by the emperor of the empire, Zlatan I, the states of the empire correspond to the largest possible groups of cities such that it is possible to travel from any city in the group to any other city within the same group. Roads that directly connect cities within the same state are known as *intra-state roads*, while those that connect cities from different states are known as *inter-state roads*. A toll is charged for every road, whether intra-state or inter-state.

In each state, there is a capital city, defined as the city in the state for which the average toll required to travel to any other city in the state is minimum. If multiple cities have the same minimum average toll, the capital is the city with the smallest identifier.

Clearly, the current road network—although it ensures that travel between cities within the same state is always possible—does not guarantee that one can travel from a city in one state to a city in another state. For this reason, Emperor Zlatan has decided to expand the inter-state roads to make them bidirectional. However, to finance this project, the toll charged for the new direction of travel will be twice the toll charged in the original direction.

After this change, the capital of the entire empire will be defined as the state capital such that, from that state, the maximum number of inter-state roads that need to be traversed to reach any other state is minimized. If multiple state capitals satisfy this condition, the capital of the entire AGRA empire will be the one with the smallest identifier among them. Zlatan would like to know the sum of tolls required to travel from this empire capital to each of the other state capitals, assuming the travel is done independently for each destination.

Input

There will be multiple test cases in the input. The first line contains an integer T, the number of test cases. For each test case, the first line contains two positive integers n and m ($2 \leq n \leq 5000, 0 < m \leq n * (n + 1)/2$), representing the number of cities in the empire and the number of original one-directional roads. Then, m lines follow, each containing three positive integers u, v, c ($0 \leq u, v < n, c > 0$), indicating the existence of a road from city u to city v with a toll c.

The input must be read from standard input.

Output

For each test case, print two integers u and r:

u is the identifier of the empire capital city, and

r is the total toll required to travel from city u to all other state capitals, assuming the routes are taken independently.

The output must be written to standard output.

Sample Input	**Sample Output**
1 11 14 0 1 1 1 2 2 2 0 3 1 3 4 3 4 2 4 3 5 3 8 3 8 9 6 9 10 7 10 9 5 4 5 5 5 6 3 6 7 4 7 5 2	3 46