Árboles y Grafos, 2025-2

Para entregar el sábado 22 de noviembre 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.

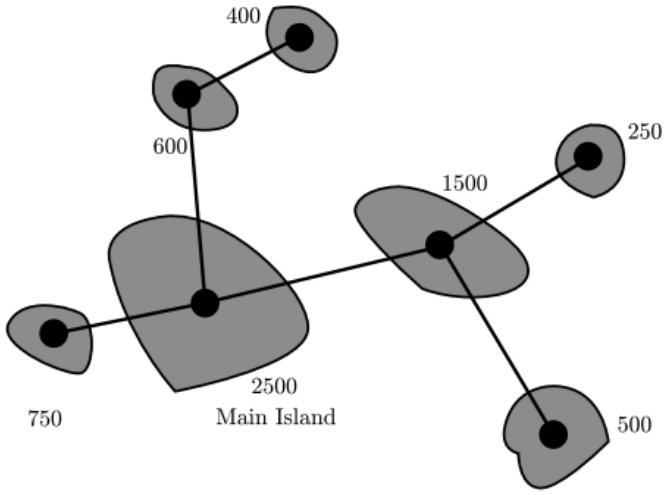
Problemas prácticos

Hay cinco problemas prácticos cuyos enunciados aparecen a partir de la siguiente página.

A - Problem A

Source file name: `island.cpp, island.py`
Time limit: x seconds

The company Pacific Island Net (PIN) has identified several small island groups in the Pacific that do not have a fast internet connection. PIN plans to tap this potential market by offering internet service to the island inhabitants. Each groups of islands already has a deep-sea cable that connects the main island to the closest internet hub on the mainland (be it America, Australia or Asia). All that remains to be done is to connect the islands in a group to each other. You must write a program to help them determine a connection procedure.



For each island, you are given the position of its router and the number of island inhabitants. In the figure, the dark dots are the routers and the numbers are the numbers of inhabitants. PIN will build connections between pairs of routers such that every router has a path to the main island. PIN has decided to build the network such that the total amount of cable used is minimal. Under this restriction, there may be several optimal networks. However, it does not matter to PIN which of the optimal networks is built.

PIN is interested in the average time required for new customers to access the internet, based on the assumption that construction on all cable links in the network begins at the same time. Cable links can be constructed at a rate of one kilometer of cable per day. As a result, shorter cable links are completed before the longer links. An island will have internet access as soon as there is a path from the island to the main island along completed cable links. If m_i is the number of inhabitants of the i^{th} island and t_i is the time when the island is connected to the internet, then the average connection time is:

$$\frac{\sum t_i * m_i}{\sum m_i}$$

Input

The input consists of several descriptions of groups of islands. The first line of each description contains a single positive integer n, the number of islands in the group ($n \leq 50$). Each of the next n lines has three integers x_i, y_i, m_i, giving the position of the router (x_i, y_i) and number of inhabitants m_i ($m_i > 0$) of the islands. Coordinates are measured in kilometers. The first island in this sequence is the main island.

The input is terminated by the number zero on a line by itself.

The input must be read from standard input.

Output

For each group of islands in the input, output the sequence number of the group and the average number of days until the inhabitants are connected to the internet. The number of days should have two digits to the right of the decimal point. Use the output format in the sample given below.

Place a blank line after the output of each test case.

The output must be written to standard output.

Sample Input	**Sample Output**
7 11 12 2500 14 17 1500 9 9 750 7 15 600 19 16 500 8 18 400 15 21 250 0	Island Group: 1 Average 3.20

B - Problem B

Source file name: `money.cpp, money.py`
Time limit: x seconds

Our sad tale begins with a tight clique of friends. Together they went on a trip to the picturesque country of Molvania. During their stay, various events which are too horrible to mention occurred. The net result was that the last evening of the trip ended with a momentous exchange of "I never want to see you again!". A quick calculation tells you it may have been said almost 50 million times!

Back home in Scandinavia, our group of ex-friends realize that they haven't split the costs incurred during the trip evenly. Some people may be out several thousand crowns. Settling the debts turns out to be a bit more problematic than it ought to be, as many in the group no longer wish to speak to one another, and even less to give each other money.

Naturally, you want to help out, so you ask each person to tell you how much money she owes or is owed, and whom she is still friends with. Given this information, you're sure you can figure out if it's possible for everyone to get even, and with money only being given between persons who are still friends.

Input

The first line of the input file contains an integer N denoting the total number of test cases. Each test case begins with a line containing two integers, n ($2 \leq n \leq 10000$), and m ($0 \leq m \leq 50000$), the number of friends and the number of remaining friendships. Then n lines follow, each containing an integer o ($-10000 \leq o \leq 10000$) indicating how much each person owes (or is owed if $o < 0$). The sum of these values is zero. After this comes m lines giving the remaining friendships, each line containing two integers x, y ($0 \leq x < y \leq n - 1$) indicating that persons x and y are still friends.

The input must be read from standard input.

Output

For each test case your output should consist of a single line saying '`POSSIBLE`' or '`IMPOSSIBLE`'.

The output must be written to standard output.

Sample Input	**Sample Output**
2 5 3 100 -75 -25 -42 42 0 1 1 2 3 4 4 2 15 20 -10 -25 0 2 1 3	POSSIBLE IMPOSSIBLE

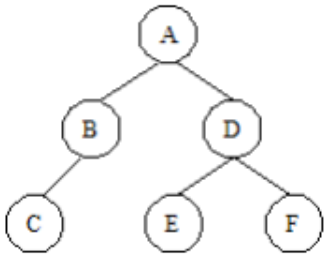
C - Problem V

Source file name: `preinpos.cpp, preinpos.py`
Time limit: x seconds

A common problem in data structures is to determine the traversal of a binary tree. There are three classic ways to do it:

- **Pre-order**: You must visit in sequence the root, left subtree and the right subtree.
- **In-order**: You must visit in sequence the left subtree, the root and the right subtree.
- **Post-order**: You must visit in sequence the left subtree, the right subtree and the root.

See the picture below:



The pre, in and post-order traversal are, respectively, ABCDEF, CBAEDF and CBEFDA. In this problem, you must compute the post-order traversal of a binary tree given its in-order and pre-order traversals.

Input

The input set consists of a positive number C, that gives the number of test cases and C lines, one for each test case. Each test case starts with a number $1 \leq N \leq 52$, the number of nodes in this binary tree. After, there will be two strings S_1 and S_2 that describe the pre-order and in-order traversal of the tree. The nodes of the tree are labeled with different characters in the range `'a' .. 'z'` and `'A' .. 'Z'`. The values of N, S_1 and S_2 are separeted by a blank space.

The input must be read from standard input.

Output

For each input set, you should output a line containing the post-order transversal for the current tree.

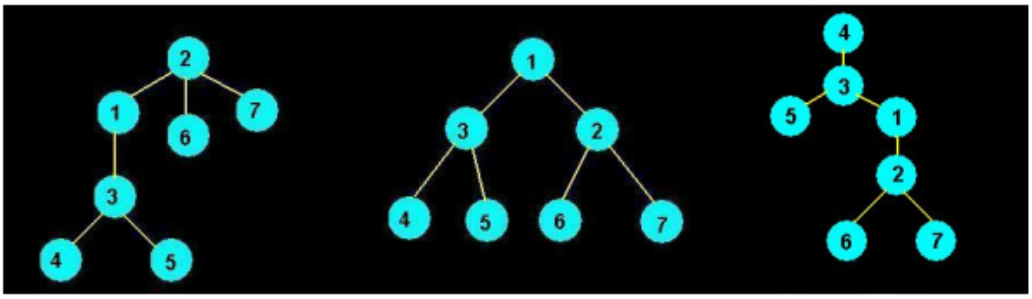
The output must be written to standard output.

Sample Input	**Sample Output**
3 3 xYz Yxz 3 abc cba 6 ABCDEF CBAEDF	Yzx cba CBEFDA

D - Problem D

Source file name: `root.cpp, root.py`
Time limit: x seconds

Tree is an important data structure. Searching is a basic operation in any data structure. In a tree searching mainly depends on its height. Consider the following three trees.



If you observe carefully, you will see that all trees are same except different nodes are used as roots. Here the height of the tree varies with the selection of the root. In the 1st tree root is '2' and height is 3. In 2nd one root is '1' and height is 2. And in last one root is '4' and height is 4. We will call '1' best root as it keeps the tree with the least possible height and '4' worst root for the opposite reason.

In this problem, you have to find out all best roots and worst roots for a given tree.

Input

Each dataset starts with a positive integer N ($3 \leq N \leq 5000$), which is the number of nodes in the tree. Each node in the tree has a unique id from 1 to N. Then successively for each i'th node there will be a positive integer $K[i]$ following id of $K[i]$ nodes which are adjacent to i. Input is terminated by EOF.

The input must be read from standard input.

Output

For each dataset print two lines. In the 1st line show all the best roots in ascending order and in next line show all worst roots in ascending order. See sample output for exact format.

The output must be written to standard output.

Sample Input	**Sample Output**
7 2 2 3 3 1 6 7 3 1 4 5 1 3 1 3 1 2 1 2	Best Roots : 1 Worst Roots : 4 5 6 7

E - Problem E

Source file name: `zlatan.cpp, zlatan.py`
Time limit: 7 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. The roads of the empire were designed in such a way that if it is possible to go from state A to state B, there is only one possible sequence of intermediate states that one must pass through.

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**
2 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 5 8 0 1 5 1 0 1 1 2 4 1 3 10 2 3 4 3 2 2 2 4 6 3 4 3	3 46 3 13