ADA - Análisis y Diseño de Algoritmos, 2025-1
Tarea 3: Semanas 7 y 8

Para entregar el domingo 16 de marzo de 2025

Problemas conceptuales a las 23:59 por BrightSpace

Problemas prácticos a las 23:59 en la arena de programación

Tanto los ejercicios como los problemas deben ser resueltos, pero únicamente las soluciones de los problemas deben ser entregadas. La intención de los ejercicios es entrenarlo para que domine el material del curso; a pesar de que no debe entregar soluciones a los ejercicios, usted es responsable del material cubierto en ellos.

Instrucciones para la entrega

Para esta tarea y todas las tareas futuras, la entrega de soluciones es *individual*. Por favor escriba claramente su nombre, código de estudiante y sección en cada hoja impresa entregada o en cada archivo de código (a modo de comentario). Adicionalmente, agregue la información de fecha y nombres de compañeros con los que colaboró; igualmente cite cualquier fuente de información que utilizó.

¿Cómo describir un algoritmo?

En algunos ejercicios y problemas se pide "dar un algoritmo" para resolver un problema. Una solución debe tomar la forma de un pequeño ensayo (es decir, un par de párrafos). En particular, una solución debe resumir en un párrafo el problema y cuáles son los resultados de la solución. Además, se deben incluir párrafos con la siguiente información:

- una descripción del algoritmo en castellano y, si es útil, pseudo-código;
- por lo menos un diagrama o ejemplo que muestre cómo funciona el algoritmo;
- una demostración de la corrección del algoritmo; y
- un análisis de la complejidad temporal del algoritmo.

Recuerde que su objetivo es comunicar claramente un algoritmo. Las soluciones algorítmicas correctas y descritas *claramente* recibirán alta calificación; soluciones complejas, obtusas o mal presentadas recibirán baja calificación.

Problemas conceptuales

1. Ejercicio 14.1 (a,c,e,g): *Scheduling* (Erickson, página 177).
2. Ejercicio 23: *Climbing* (Erickson, página 184).

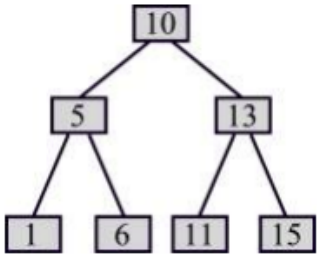
Problemas prácticos

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

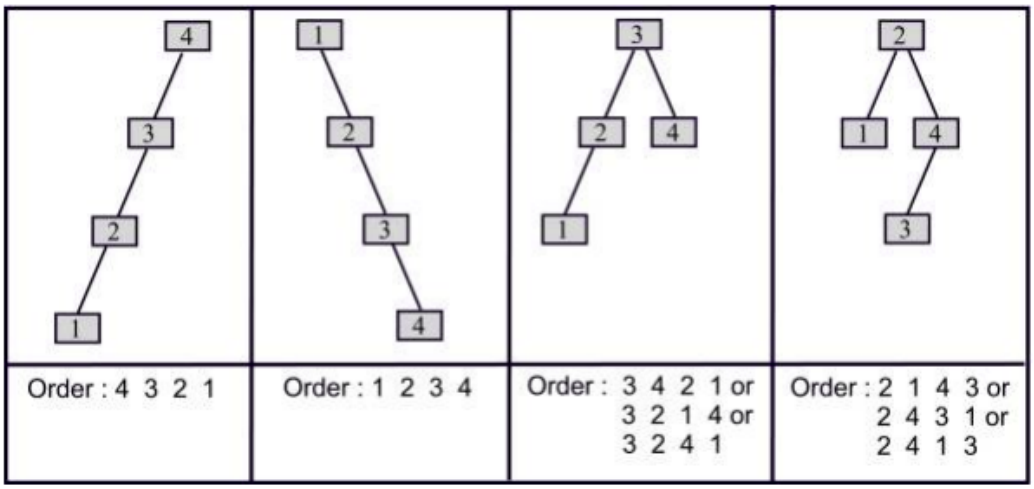
A - Constructing BST

Source file name: `bst.py`
Time limit: x ssecond

BST (Binary Search Tree) is an efficient data structure for searching. In a BST all the elements of the left sub-tree are smaller and those of right sub-tree are greater than the root. A typical example of BST is



Normally, we construct BST by successively inserting an element. In that case, the ordering of elements has great impact on the structure of the tree. Look at the following cases:



In this problem, you have to find the order of 1 to N integers such that the BST constructed by them has height of at most H. The height of a BST is defined by the following relation

1. BST having no node has height 0.
2. Otherwise, it is equal to the maximum of the height of the left sub-tree and right sub-tree plus 1.

Again, several orderings can satisfy the criterion. In that case we prefer the sequence where smaller numbers come first. For example, for $N = 4$, $H = 3$ we want the sequence 1 3 2 4 rather than 2 1 4 3 or 3 2 1 4.

Input

Each test case starts with two positive integers N ($1 \leq N \leq 10000$) and H ($1 \leq H \leq 30$). Input is terminated by $N = 0$, $H = 0$. This case should not be processed. There can be at most 30 test cases.

The input must be read from standard input.

Output

Output of each test case should consist of a line starting with '`Case #:`' where `#` is the test case number. It should be followed by the sequence of *N* integers in the same line. There must not be any trailing space at the end of the line. If it is not possible to construct such tree then print '`Impossible.`' (without the quotes).

The output must be written to standard output.

Sample Input	**Sample Output**

`4 3`
`4 1`
`6 3`
`0 0`
	`Case 1: 1 3 2 4`
	`Case 2: Impossible.`
	`Case 3: 3 1 2 5 4 6`

B - Invitation Cards

Source file name: `cards.py`
Time limit: x seconds

In the age of television, not many people attend theater performances. Antique Comedians of Malidinesia are aware of this fact. They want to propagate theater and, most of all, Antique Comedies. They have printed invitation cards with all the necessary information and with the programme. A lot of students were hired to distribute these invitations among the people. Each student volunteer has assigned exactly one bus stop and he or she stays there the whole day and gives invitation to people travelling by bus. A special course was taken where students learned how to influence people and what is the difference between influencing and robbery.

The transport system is very special: all lines are unidirectional and connect exactly two stops. Buses leave the originating stop with passengers each half an hour. After reaching the destination stop they return empty to the originating stop, where they wait until the next full half an hour, e.g. X:00 or X:30, where 'X' denotes the hour. The fee for transport between two stops is given by special tables and is payable on the spot. The lines are planned in such a way, that each round trip (i.e., a journey starting and finishing at the same stop) passes through a Central Checkpoint Stop (CCS) where each passenger has to pass a thorough check including body scan.

All the ACM student members leave the CCS each morning. Each volunteer is to move to one predetermined stop to invite passengers. There are as many volunteers as stops. At the end of the day, all students travel back to CCS. You are to write a computer program that helps ACM to minimize the amount of money to pay every day for the transport of their employees.

Input

The input consists of N cases. The first line of the input contains only positive integer N. Then follow the cases. Each case begins with a line containing exactly two integers P and Q, $1 \leq P, Q \leq 1\,000\,000$. P is the number of stops including CCS and Q the number of bus lines. Then there are Q lines, each describing one bus line. Each of the lines contains exactly three numbers —the originating stop, the destination stop, and the price. The CCS is designated by number 1. Prices are positive integers the sum of which is smaller than 1 000 000 000. You can also assume it is always possible to get from any stop to any other stop.

The input must be read from standard input.

Output

For each case, print one line containing the minimum amount of money to be paid each day by ACM for the travel costs of its volunteers.

The output must be written to standard output.

Sample Input	**Sample Output**
2 2 2 1 2 13 2 1 33 4 6 1 2 10 2 1 60 1 3 20 3 4 10 2 4 5 4 1 50	46 210

C - Keep the Customer Satisfied

Source file name: `customer.py`
Time limit: x seconds

Simon and Garfunkel Corporation (SG Corp.) is a large steel-making company with thousand of customers. Keeping the customer satisfied is one of the major objective of Paul and Art, the managers.

Customers issue orders that are characterized by two integer values q, the amount of steel required (in tons), and d, the due date (a calender date converted in seconds). The due date has to be met if SG Corp. accepts the order. Stated another way, when an order is accepted, the corresponding amount of steel has to be produced before its due date. Of course, the factory can process no more than one order at a time.

Although the manufacturing process is rather complex, it can be seen as a single production line with a constant throughput. In the following, we assume that producing q tons of steel takes exactly q seconds (i.e., throughput is 1). The factory runs on a monthly production plan. Before the beginning of the month, all customers' orders are collected and Paul and Art determine which of them are going to be accepted and which ones are to be rejected in the next production period. A production schedule is then designed. To keep customers satisfied, Paul and Art want to minimize the total number of orders that are rejected. In the following, we assume that the beginning of the next production plan (i.e., the first day of the next month) corresponds to date 0.

Consider the following data set made of 6 orders $J_1, \ldots, J_6$. For a given order, J_j, q_j denotes the amount of steel required and d_j is the associated due date.

Order	q_j	d_j
J_1	6	8
J_2	4	9
J_3	7	15
J_4	8	20
J_5	3	21
J_6	5	22

You can check by hand that all orders cannot be accepted and it's very unlikely you could find a solution with less than two rejected orders. Here is an optimal solution: Reject J_1 and J_4, accept all other orders, and process them as follows.

Accepted Order	Starting Time	Completion Time
J_2	0	4
J_3	4	11
J_5	11	14
J_6	14	19

Note that the production line is never idle.

Hogdson and Moore have been appointed as Chief Scientific Officers and you are requested to help them to compute an optimal solution and to build a schedule of all accepted orders (starting time and completion time).

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs. In each test case, the first line contains the number n of orders (n can be as large as 800 000). It is followed by n lines. Each of which describes an order made of two integer values: the amount of steel (in tons) required for the order (lower than 1 000) and its due date (in seconds; lower than 2×10^6).

The input must be read from standard input.

Output

For each test case, you are required to compute an optimal solution and your program has to write the number of orders that are accepted. The outputs of two consecutive cases will be separated by a blank line.

The output must be written to standard output.

Sample Input	**Sample Output**
1 6 7 15 8 20 6 8 4 9 3 21 5 22	4

D - Packets

Source file name: `packets.py`
Time limit: x seconds

A factory produces products packed in square packets of the same height h and of the sizes 1×1, 2×2, 3×3, 4×4, 5×5, 6×6. These products are always delivered to customers in the square parcels of the same height h as the products have and of the size 6×6. Because of the expenses it is the interest of the factory as well as of the customer to minimize the number of parcels necessary to deliver the ordered products from the factory to the customer. A good program solving the problem of finding the minimal number of parcels necessary to deliver the given products according to an order would save a lot of money. You are asked to make such a program.

Input

The input file consists of several lines specifying orders. Each line specifies one order. Orders are described by six integers separated by one space representing successively the number of packets of individual size from the smallest size 1×1 to the biggest size 6×6. The end of the input file is indicated by the line containing six zeros.

The input must be read from standard input.

Output

The output file contains one line for each line in the input file. This line contains the minimal number of parcels into which the order from the corresponding line of the input file can be packed.

The output must be written to standard output.

Sample Input	**Sample Output**

`0 0 4 0 0 1`
`7 5 1 0 0 0`
`0 0 0 0 0 0`
	`2`
	`1`

E - Tour Belt

Source file name: `tour.py`
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 $\text{SE}(u, v)$ or $\text{SE}(v, u)$ 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 $\text{SE}(u, v)$ 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 $\{7, 8\}$.

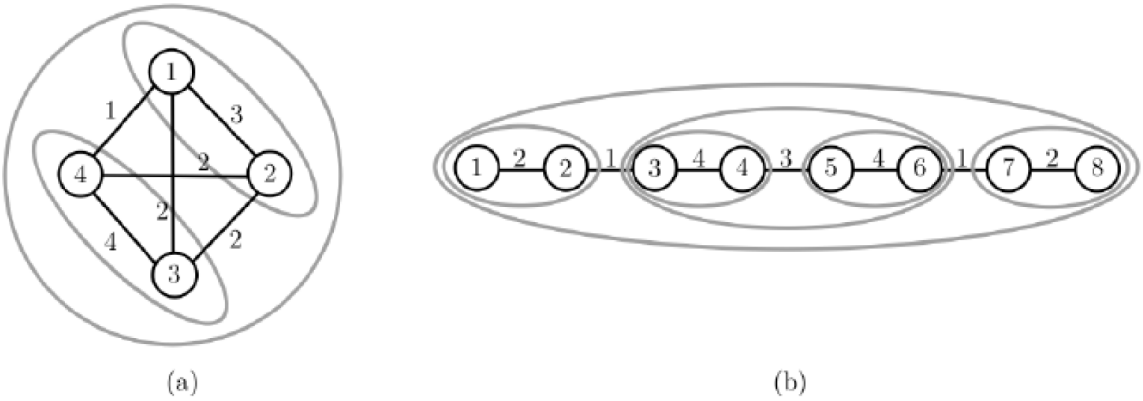


Figure 1. Graphs and their good subsets marked by gray ellipses.

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