ADA - Análisis y Diseño de Algoritmos, 2019-1
Tarea 3: Semanas 6 y 7

Para entregar el martes 12 de marzo/lunes 11 de marzo de 2019

Problemas conceptuales a las 09:00 (12 de marzo) antes de clase

Problemas prácticos a las 23:59 (11 de marzo) 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.

Ejercicios

16.1-1, 16.1-2, 16.1-3, 16.1-4, 16.1-5 (página 422), 16.2-1, 16.2-3, 16.2-5, 16.2-7 (páginas 427 y 428).

Problemas conceptuales

1. Problema 16-1 (a-c): *Greedy Coin Change* (Cormen et al., página 446).
2. Ejercicio 4.3: *Trucking Company* (Kleinberg y Tardos, página 189).

Problemas prácticos

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

A - Gas Stations

Source file name: `gas.py`
Time limit: 1 second

G gas stations are authorized to operate over a road of length L. Each gas station is able to sell fuel over a specific area of influence, defined as the closed interval $[x - r, x + r]$, where x is the station's *location* on the road ($0 \leq x \leq L$) and r is its *radius of coverage* ($0 < r \leq L$). The points covered by a gas station are those within its radius of coverage.

It is clear that the areas of influence may interfere, causing disputes among the corresponding gas stations. It seems to be better to close some stations, trying to minimize such interferences without reducing the service availability along the road.

The owners have agreed to close some gas stations in order to avoid as many disputes as possible. You have been hired to write a program to determine the maximum number of gas stations that may be closed, so that every point on the road is in the area of influence of some remaining station. By the way, if some point on the road is not covered by any gas station, you must acknowledge the situation and inform about it.

Input

The input consists of several test cases. The first line of each test case contains two integer numbers L and G (separated by a blank), representing the length of the road and the number of gas stations, respectively ($1 \leq L \leq 10^8$, $1 \leq G \leq 10^4$). Each one of the next G lines contains two integer numbers x_i and r_i (separated by a blank) where x_i is the location and r_i is the radius of coverage of the i-th gas station ($0 \leq x_i \leq L$, $0 < r_i \leq L$). The last test case is followed by a line containing two zeros.

The input must be read from standard input.

Output

For each test case, print a line with the maximum number of gas stations that can be eliminated, so that every point on the road belongs to the area of influence of some not closed station. If some point on the road is not covered by any of the initial G gas stations, print '`-1`' as the answer for such a case

The output must be written to standard output.

Sample Input	Sample Output

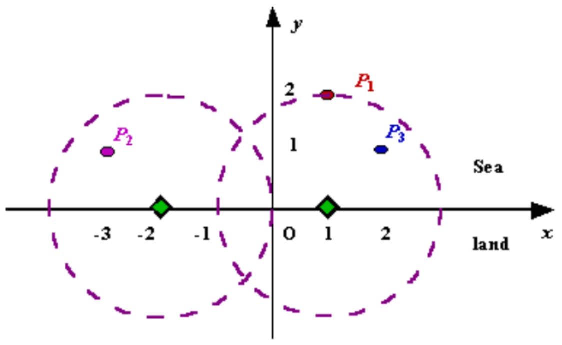
40 3
5 5
20 10
40 10
40 5
5 5
11 8
20 10
30 3
40 10
40 5
0 10
10 10
20 10
30 10
40 10
40 3
10 10
18 10
25 10
40 3
10 10
18 10
25 15
0 0
	0
	2
	3
	-1
	1

B - Radar Installation

Source file name: `install.py`
Time limit: 1 second

Assume the coasting is an in nite straight line. Land is in one side of coasting, sea in the other. Each small island is a point locating in the sea side. And any radar installation, locating on the coasting, can only cover d distance, so an island in the sea can be covered by a radius installation, if the distance between them is at most d.

We use Cartesian coordinate system, de ning the coasting is the x-axis. The sea side is above x-axis, and the land side below. Given the position of each island in the sea, and given the distance of the coverage of the radar installation, your task is to write a program to find the minimal number of radar installations to cover all the islands. Note that the position of an island is represented by its x-y coordinates.



Input

The input consists of several test cases. The first line of each case contains two integers n ($1 \leq n \leq 1\,000$) and d, where n is the number of islands in the sea and d is the distance of coverage of the radar installation. This is followed by n lines each containing two integers representing the coordinate of the position of each island. Then a blank line follows to separate the cases. The input is terminated by a line containing pair of zeros.

The input must be read from standard input.

Output

For each test case output one line consisting of the test case number followed by the minimal number of radar installations needed. '`-1`' installation means no solution for that case.

The output must be written to standard output.

Sample Input	Sample Output

```
3 2
1 2
-3 1
2 1

1 2
0 2

0 0
```
	```
	Case 1: 2
	Case 2: 1
	```