ISIS 1105 Diseño y Análisis de Algoritmos
Semestre 2025-20. Proyecto – PARTE 3
Entrega: jueves, 4 de diciembre de 2025 6:00PM (por Bloque Neon)

0 OBJETIVOS

- Diseñar soluciones computacionales para problemas.
- Estimar costos de las soluciones planteadas.
- Implementar soluciones.

Se premiarán las mejores soluciones y se castigarán las peores, en cuanto a optimización y eficiencia en tiempo y espacio.

1 CONDICIONES GENERALES

El proyecto se divide en tres partes independientes entre sí. Este documento describe la PARTE III. Cada parte contiene un problema a resolver mediante soluciones implementadas en `Java` o `Python`.

Para cada problema se pide:
- Descripción de la solución.
- Análisis temporal y espacial.
- Una implementación en Java o Python

2 DESCRIPCIÓN DEL PROBLEMA

Don Jaime Velosa es un joven agricultor que sueña con tener la cosecha más grande de la temporada. Sin embargo, se ha enfrentado a un gran desafío: una plaga está afectando puntos específicos de su extenso campo. A partir de observaciones satelitales, Jaime ha logrado establecer unos puntos que corresponden a los focos exactos de reproducción de la plaga.

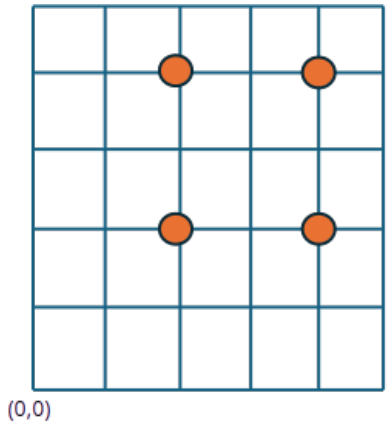
Para combatir la plaga, Jaime cuenta con dos procedimientos diferentes de control. Por una parte, puede contratar un servicio de aspersión aérea de plaguicida, por medio de drones. Sin embargo, debido a las condiciones atmosféricas del terreno, los drones solamente pueden volar en trayectorias rectas de norte a sur. Por otra parte, los resultados de una investigación reciente, concluyeron que una nueva especie de artrópodo se alimenta de la plaga y por lo tanto puede servir como control biológico. Sin embargo, el estudio también mostró que los artrópodos solamente pueden desplazarse en línea recta de oriente a occidente. Además de esto, un artrópodo no se puede cruzar con una trayectoria de aspersión en un punto diferente a un foco de infección porque no tendría los recursos suficientes para sobrevivir en las condiciones de la aspersión.

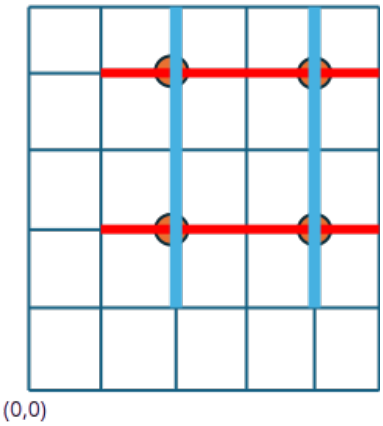
Problema

Jaime quiere saber cuál es la menor cantidad de aspersores y de artrópodos que debe utilizar bajo estas condiciones, de modo que los focos de reproducción de la plaga sean cubiertos tanto por aspersión como por artrópodos, pero ningún otro punto del cultivo sea cubierto de esta forma.

Ejemplo:

La parte izquierda de la siguiente figura ilustra un ejemplo del cultivo de Jaime con cuatro focos de reproducción. En este caso requiere como mínimo dos drones (trayectorias azules) y dos artrópodos (trayectorias rojas).





3 ENTRADA Y SALIDA DE DATOS

En todas las soluciones que se presenten, la lectura de los datos de entrada se hace por la entrada estándar; así mismo, la escritura de los resultados se hace por la salida estándar.

Puede suponer que ninguna línea de entrada tiene espacios al principio o al final, y que los datos que se listan en cada línea están separados por exactamente un espacio.

A continuación, se establecen parámetros que definen su tamaño y formato de lectura de los datos, tanto de entrada como de salida.

Descripción de la entrada

La primera línea de entrada especifica el número de casos de prueba que contiene el archivo. El programa debe terminar su ejecución, una vez termine de resolver la cantidad de casos de prueba dados por este número.

Cada caso está representado por una línea. El primer elemento de dicha línea contiene un entero n $(1 \leq n \leq 1000)$, el número de focos de infección. Seguido de n va a encontrar n parejas de números donde cada pareja representa las coordenadas de un foco de infección. Cada foco es un par x y y $(1 \leq x, y \leq 10^5)$. Todos los focos son distintos.

Descripción de la salida

Para cada caso de prueba, debe imprimir dos líneas.

En la primera línea tiene que imprimir: (i) el número de artrópodos a utilizar en trayectorias horizontales $(0 \leq h)$. (ii) las coordenadas de las h trayectorias representadas como 4 enteros x_1, y_1, x_2, y_2 , ie. las coordenadas de dos puntos (x_1, y_1) y (x_2, y_2) que determinan esta trayectoria.

La segunda línea es para las trayectorias verticales correspondientes a los drones $(0 \leq v)$. Se imprime igual que las trayectorias horizontales.

No debe haber dos trayectorias horizontales que compartan puntos entre si. Lo mismo aplica para las trayectorias verticales. El número total de trayectorias $(h + v)$ debe ser lo mínimo posible. Si hay múltiples respuestas posibles mínimas, imprima cualquiera.

Ejemplo de entrada / salida

Entrada	Salida
2 4 2 2 2 4 4 2 4 4 4 2 1 3 2 2 3 1 2	2 5 2 1 2 1 4 5 4 2 2 1 2 5 4 5 4 1 4 2 1 2 1 3 2 3 2 1 2 1 2 2 3 2 3 3 1 2 1 2 3 2 3 2 2 3 2 1

Nota: Se van a diseñar casos de prueba para valores de n, x, y mucho más grandes y dentro de los valores establecidos en el enunciado. Los casos mostrados en este documento son demostrativos de la estructura de entrada/salida esperada.

5 ENTREGABLES

El proyecto puede desarrollarse por grupos de hasta dos estudiantes de la misma sección. La entrega se hace por bloque neon (una sola entrega por grupo de trabajo).

El grupo debe entregar, por bloque neon, un archivo de nombre `proyectoDalgoP3.zip`. Este archivo es una carpeta de nombre `proyectoDalgoP3`, comprimida en formato `.zip`, dentro de la cual hay archivos fuente de soluciones propuestas y archivos que documentan cada una de las soluciones.

5.1 Archivos fuente de soluciones propuestas

Todos los programas implementados en *Java* o en Python

Para el problema:

- Entregar un archivo de código fuente en *Java* (`.java`) o python (`.py`) con su código fuente de la solución que se presenta.
- Incluir como encabezado de cada archivo fuente un comentario que identifique el (los) autor(es) de la solución.
- Denominar `ProblemaP3.java` o `ProblemaP3.py` el archivo de la solución que se presente.

Nótese que, si bien puede utilizarse un *IDE* como *Eclipse* o Spyder durante el desarrollo del proyecto, la entrega requiere incluir solo un archivo por cada solución. El archivo debe poderse compilar y ejecutar independientemente (sin depender de ninguna estructura de directorios, librerías no estándar, etc.).

5.2 Archivos que documentan la solución propuesta

La solución al problema debe acompañarse de un archivo de máximo 3 páginas que la documente, con extensión `.pdf`. El nombre del archivo debe ser el mismo del código correspondiente (`ProblemaP3.pdf`).

Un archivo de documentación debe contener los siguientes elementos:

0 Identificación
Nombre de autor(es)

1 Algoritmo de solución

Explicación del algoritmo elegido. Si hubo alternativas de implantación diferentes, explicar por qué se escogió la que se implementó. Generar al menos una gráfica que apoye la explicación del algoritmo implementado. No se debe copiar y pegar código fuente como parte de la explicación del algoritmo.

2 *Análisis de complejidades espacial y temporal*
Cálculo de complejidades y explicación de estas.

La nota del informe corresponde a un 50% de la nota total de la entrega del proyecto, solamente si se entrega el código fuente de la solución implementada. En caso de no entregar el código fuente, no se hará evaluación del informe y la nota del proyecto será cero.

Además de la pertinencia del texto como explicación de la solución implementada, se evaluará la calidad en la redacción del texto y en el diseño de las gráficas. Se evaluará también que la explicación del algoritmo integre los conceptos relacionados con las técnicas de diseño de algoritmos cubiertas en el curso.

Téngase en cuenta que los análisis de 2 tienen sentido en la medida que la explicación de 1 sea clara y correcta. No se está exigiendo formalismo a ultranza, pero sí que, como aplicación de lo estudiado en el curso, se pueda describir un algoritmo de manera correcta y comprensible.

5.3 Consideraciones sobre la entrega

Lea bien las recomendaciones de entrada/salida. Su proyecto será evaluado, en gran parte, por una máquina: si en la ejecución del programa el formato de salida no coincide perfectamente con lo requerido, la calificación de la ejecución será cero, sin importar la cantidad de código que haya desarrollado.

La forma en que se presenta la documentación debe respetarse, aunque su evaluación no sea tan automática como la del software. Hay que nombrar los archivos como se espera que se nombren, comprimirlos como se pide que se compriman, etc. Cualquier desviación en cuanto a lo que se pide produce deméritos en la calificación final.

El software se evalúa desde la línea de comandos de Linux. No hay problema en desarrollar en cualquier otro sistema (Mac, Unix, ...) siempre que se produzca código estándar en java o python. Cada problema se debe resolver en un (1) archivo.