Trabajo Práctico 1: Técnicas Algorítmicas

Compilado: 14 de septiembre de 2022

Fecha de entrega: 21 de septiembre de 2022 (antes de las 23:59)
Fecha de reentrega: 28 de noviembre de 2022 (antes de las 23:59)

Este trabajo práctico consta de tres ejercicios más o menos independientes. En dos de estos tres ejercicios, parte de la evaluación consistirá en el envío del código escrito a un juez online y pasar una batería de tests en el tiempo estipulado. Para la aprobación del trabajo práctico se debe entregar:

1. Un informe *de a lo sumo diez páginas* que describa la solución de los ejercicios y responda a todas las consignas del enunciado.

2. El código fuente con los programas que implementan las soluciones propuestas a los ejercicios.

3. El link a su página de usuario en https://onlinejudge.org, donde se vean las entregas aceptadas para los problemas 1 y 2[1].

El informe debe ser autocontenido, lo que significa que cualquier estudiante potencial de AED3 que conozca los temas debe poder leerlo sin necesidad de conocer cuál fue el enunciado. En particular, todos los conceptos y notaciones utilizados que no sean comunes a la materia deben definirse, incluso aquellos que se encuentren en el presente enunciado. No es necesario explicar aquellos conceptos propios de la materia ni la teoría detrás de las distintas técnicas algorítmicas o de demostración de propiedades.

El informe debe estar estructurado usando una sección independiente por cada ejercicio resuelto. Esto no impide que un ejercicio haga referencias a la solución de otro ejercicio; las comparaciones son muchas veces bienvenidas o parte de la consigna. Cada sección debe dar una respuesta cabal a todas las preguntas del enunciado. Sin embargo, el informe no es una sucesión de respuestas a las distintas consignas, sino una elaboración única y coherente donde se pueden encontrar las respuestas a las preguntas. La idea es que el informe sea de agradable lectura.

El informe **no debe incluir código fuente**, ya que quienes evaluamos tenemos acceso al código fuente del programa. En caso de utilizar *pseudocódigo*, el mismo debe ser de alto nivel, evitando construcciones innecesariamente complicadas para problemas simples.[2] Todo pseudocódigo debe estar acompañado de un texto coloquial que explique lo que el pseudocódigo hace en términos conceptuales.

El código fuente entregado debe venir acompañado de un documento que explique cómo compilar y ejecutar el programa. El código fuente debe compilar y debe resolver correctamente **todos** los casos de test que se le presenten en un tiempo que se corresponda a la complejidad esperada. En todos los ejercicios, la instancia se leerá de la entrada estándar y la solución se imprimirá en la salida estandar. Junto a este enunciado se incluyen algunos casos de test, mientras que la complejidad temporal en peor caso, de ser solicitada, forma parte del enunciado. Tener en cuenta que el programa puede ser probado en casos de test adicionales.

Para aprobar el trabajo práctico es necesario aprobar cada ejercicio en forma individual, ya sea en la primera entrega o en el recuperatorio. No es necesario reentregar aquellos ejercicios que sean

[1] Para generar esta página, primero se debe determinar el id de usuario propio, ese dato está en la sección de "My account" bajo el nombre *Online Judge ID*. Luego, se puede acceder por medio del link https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_authorstats&userid=<id>, reemplazando el "<id>" con el identificador obtenido.

[2] E.g., "tomar el máximo del vector V" vs. "poner max := V_0; para cada $i = 1, \ldots, n-1$: poner max := maximo(max, V_i);

aprobados en la primer entrega. Asimismo, para aprobar cada ejercicio es necesario que el informe describa correctamente la solución y que el programa propuesto sea correcto. La correcta escritura del informe forma parte de la evaluación.

Formato de entrega Para realizar la entrega se debe enviar un mail a la lista de docentes (algo3−doc) con el subject "Entrega TP1 2C2022". El cuerpo del mail debe contener el nombre y libreta de las **cuatro** personas que integren el grupo, más el link al perfil de usuario de UVa. El mail debe tener como adjuntos el pdf del informe y un archivo zip con el código de la solución de los tres problemas. El nombre de estos archivos debe ser el de los apellidos de las personas integrantes del grupo separados por guiones.

Ejercicio 1: *Backtracking*

Se pide resolver el problema 1098 de UVa llamado *Robots on Ice*[3]. Se provee una traducción de su enunciado:

> Inspirados por las esculturas de hielo de Harbin [4], las personas integrantes del equipo de programación de la Universidad del Ártico de Robots y Autómatas decidieron celebrar su propio festival del hielo. Su idea es recolectar bloques de hielo de un lago cercano cuando este se congele durante el invierno. Para facilitar el monitoreo del espesor del hielo, dibujarán una grilla en la superficie del lago y harán que un robot ligero viaje de celda en celda para medir el grosor del hielo en cada posición de la grilla. Hay tres posiciones que son establecidas como puntos de "check-in", desde ellas el robot debe transmitir por radio su reporte de avance cuando se encuentre a un cuarto, a la mitad y a tres cuartos del progreso de la inspección. Para evitar exigir por demás la superficie del lago, el robot debe comenzar el recorrido de la grilla en la esquina inferior izquierda, designada por coordenadas (fila,columna) como $(0, 0)$, visitar cada celda de la grilla exactamente una vez y terminar en la posición de fila 0 y columna 1. Es más, si existen múltiples recorridos que el robot puede hacer, debe realizar uno distinto cada día. El robot puede moverse sólo una celda por vez en una de las direcciones de los cuatro puntos cardinales: Norte, Sur, Este y Oeste.
>
> Se pide diseñar un programa que determine cuántos recorridos posibles tiene una grilla determinados su tamaño y la posición de los puntos de *check-in*. Por ejemplo, supongamos que el lago se encuentra divido en una grilla de 3×6 y que sus puntos de *check-in*, en orden de visita, son $(2, 1)$, $(2, 4)$, y $(0, 4)$. Entonces, el robot debe comenzar en $(0, 0)$ y terminar en $(0, 1)$ luego de pasar por 18 casilleros. Debe visitar la posición $(2, 1)$ en el paso 4 $(= \lfloor 18/4 \rfloor)$, la posición $(2, 4)$ en el paso 9 $(= \lfloor 18/2 \rfloor)$, y la posición $(0, 4)$ en el paso 13 $(= \lfloor 3 \times 18/4 \rfloor)$.
>
> Hay sólo dos formas de hacer este recorrido (ver Figura 1). Notar que cuando el tamaño de la grilla no es divisible por 4, se toma el piso de la división para obtener los momentos de *check-in*.

[3] https://onlinejudge.org/index.php?option=onlinejudge&Itemid=8&page=show_problem&problem=3539

[4] Harbin es una ciudad al noreste de China que lleva el apodo de "ciudad del hielo", y que anualmente celebra en Enero un festival en donde se hacen esculturas de este elemento.

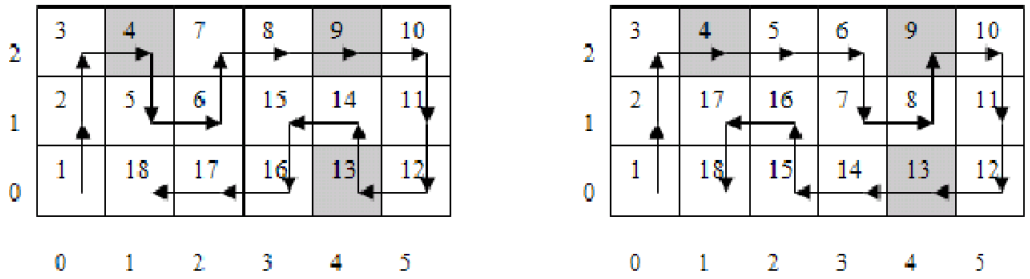

Figura 1: Posibles recorridos en una grilla de 3×6 con *check-ins* en $(2, 1)$, $(2, 4)$ y $(0, 4)$.

Descripción de una instancia

La entrada posee múltiples casos de test. Cada caso de test comienza con una línea que contiene dos enteros m y n, donde $2 \leq m$, $n \leq 8$, que especifican la cantidad de filas y columnas de la grilla, respectivamente. Esto es seguido por una línea que contiene 6 enteros r_1, c_1, r_2, c_2, y r_3, c_3, donde $0 \leq r_i < m$ y $0 \leq c_1 < n$ para $i = 1, 2, 3$. Luego del último caso de test hay una línea con dos ceros.

Descripción de la salida

Mostrar el número de caso, empezando de 1 notándolo como Case <numero>: ("Case" por caso en inglés), seguido del número de recorridos posibles que comienzan en la fila 0, columna 0 y terminan en la fila 0, columna 1, visitando la fila r_i, columna c_i en el paso $\lfloor i \times m \times n/4 \rfloor$ para $i = 1, 2, 3$. Seguir el formato de ejemplo.

Ejemplo de entrada y salida Se presenta un ejemplo de entrada y su correspondiente salida:

Entrada de ejemplo	Salida esperada de ejemplo
3 6 2 1 2 4 0 4 4 3 2 0 3 2 0 2 0 0	Case 1: 2 Case 2: 0

Se pide:

1. Diseñar un algoritmo eficiente para resolver este problema. Este algoritmo debe correr en el juez UVa en el límite de tiempo estipulado (9 segundos), por lo que será necesario diseñar podas en la búsqueda de caminos válidos.

2. Describir en el informe su estrategia de resolución y las podas aplicadas.

Ejercicio 2: Algoritmo Greedy

Se pide resolver el problema 10382 de UVa llamado *Watering Grass*[5]. Se provee una traducción de su enunciado:

[5] https://onlinejudge.org/index.php?option=onlinejudge&Itemid=8&page=show_problem&problem=1323

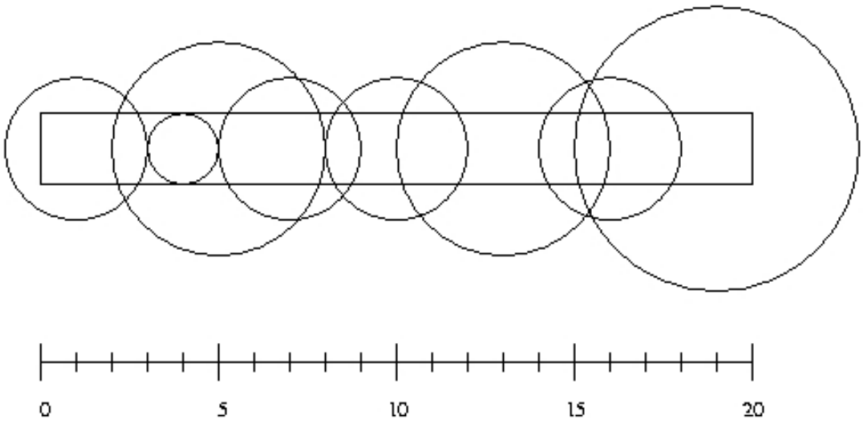

Figura 2: Rectángulo de pasto con 8 aspersores de distintos radios.

Hay n aspersores instalados en un terreno rectangular de ℓ metros de largo y w metros de ancho. Cada aspersor está instalado en el centro horizontal del rectángulo. Para cada aspersor se nos provee su posición como la distancia desde el extremo izquierdo del rectángulo y su radio de operación. ¿Cuál es el número mínimo de aspersores que debemos encender para asegurarnos de que cubrimos con agua todo el rectángulo?

Descripción de una instancia

El ínput contiene múltiples casos de test. La primera línea de cada caso contiene enteros n, ℓ y w con $n \leq 10000$. Las siguientes n líneas contienen dos enteros que dan la posición de un aspersor y su radio de operación (la Figura 2 ilustra el primer caso del input de ejemplo).

Descripción de la salida

Para cada caso de test imprimir el número mínimo de aspersores necesarios para regar todo el pasto del rectángulo. Si no es posible cubrirlo completamente, se debe imprimir -1.
Nota: El caso cuando ℓ vale 0 es posible, se debe imprimir 0.

Ejemplo de entrada y salida Se presenta un ejemplo de entrada y su correspondiente salida.

Entrada de ejemplo	Salida esperada de ejemplo
11 20 2 5 3 4 1 1 2 7 2 10 2 13 3 16 2 19 4 3 5 9 2 19 2 6 10 1 3 5 9 3 6 1 1 2 5 3 9 2 3 10 1 5 3 1 1 9 1	4 2 -1

Se pide:

1. Diseñar un algoritmo eficiente para resolver este problema. Este algoritmo debe correr en el juez UVa en el límite de tiempo estipulado (3 segundos), por lo que será necesario diseñar una estrategia golosa para resolverlo.

2. Describir en el informe su estrategia de resolución, argumentando su complejidad, y demostrar su correctitud.

Ejercicio 3: Programación Dinámica

Este ejercicio consiste en una variante del ejercicio 2. El enunciado es igual salvo que se agrega la siguiente variación:

Para cada aspersor, además de su distancia al extremo izquierdo y su radio de acción, tiene asociado un costo. Se pide devolver el *costo mínimo* necesario a pagar para cubrir toda la superficie.

Descripción de una instancia

Siguiendo la línea del Ejercicio 2, la entrada contiene múltiples casos de test, de los cuales la primera línea indica n, ℓ y w, siendo $n \leq 10000$ y ℓ y w valores que entran en un int de C++. Luego siguen n líneas que contienen tres enteros enteros: la posición de un aspersor, su radio de operación y su costo.

Descripción de la salida

En cada caso de test se debe imprimir el costo mínimo necesario a pagar para regar todo el pasto del rectángulo. Si no es posible cubrirlo completamente, se debe imprimir -1.

Ejemplo de entrada y salida Se presenta un ejemplo de entrada y su correspondiente salida.

Entrada de ejemplo	Salida esperada de ejemplo
11 20 2 5 3 1 4 1 1 1 2 1 7 2 2 10 2 1 13 3 2 16 2 1 19 4 3 3 5 5 9 2 2 19 2 1 6 10 1 3 5 4 9 3 3 6 1 1 1 2 1 5 3 1 9 2 2 3 10 1 5 3 1 1 1 2 9 1 1	8 4 -1

Se pide:

1. Dar un algoritmo de complejidad $O(n^2)$ para resolver el problema (Este no será enviado a ningún juez online pero será testeado por la cátedra).

2. Describir en el informe la estructura de la solución y argumentar su complejidad y corrección.