Trabajo práctico 2 (v. 1.1)

Fecha de entrega: viernes 8 de mayo, hasta las 18:00 hs.

Este trabajo práctico consta de varios problemas y para aprobar el trabajo se requiere aprobar todos los problemas. La nota final del trabajo será un promedio ponderado de las notas finales de los ejercicios y el trabajo práctico se aprobará con una nota de 5 (*cinco*) o superior. De ser necesario (o si el grupo lo desea) el trabajo podrá reentregarse una vez corregido por los docentes y en ese caso la reentrega deberá estar acompañada por un *informe de modificaciones*. Este informe deberá detallar brevemente las diferencias entre las dos entregas, especificando los cambios, agregados y/o partes eliminadas del trabajo. Cualquier cambio que no se encuentre en dicho informe podrá no ser tenido en cuenta en la corrección de la reentrega. Para la reentrega del trabajo **podrían pedirse ejercicios adicionales**.

Para cada ejercicio se pide encontrar una solución algorítmica al problema propuesto y desarrollar los siguientes puntos:

1. Describir detalladamente el problema a resolver dando ejemplos del mismo y sus soluciones.
2. Explicar de forma clara, sencilla, estructurada y concisa, las ideas desarrolladas para la resolución del problema. Para esto se pide utilizar pseudocódigo y lenguaje coloquial combinando adecuadamente ambas herramientas (**¡sin usar código fuente!**). Se debe también justificar por qué el procedimiento desarrollado resuelve efectivamente el problema.
3. Deducir una cota de complejidad temporal del algoritmo propuesto (en función de los parámetros que se consideren correctos) y justificar por qué el algoritmo desarrollado para la resolución del problema cumple la cota dada. Utilizar el modelo uniforme salvo que se explicite lo contrario.
4. Dar un código fuente claro que implemente la solución propuesta. El mismo no sólo debe ser correcto sino que además debe seguir las *buenas prácticas de la programación* (comentarios pertinentes, nombres de variables apropiados, estilo de indentación coherente, modularización adecuada, etc.). Se deben incluir las partes relevantes del código como apéndice del informe impreso entregado.
5. Realizar una experimentación computacional para medir la performance del programa implementado. Para ello se debe preparar un conjunto de casos de test que permitan observar los tiempos de ejecución en función de los parámetros de entrada. Deberán desarrollarse tanto experimentos con instancias aleatorias (detallando cómo fueron generadas) como experimentos con instancias particulares (de peor/mejor caso en tiempo de ejecución, por ejemplo). Se debe presentar **adecuadamente** en forma gráfica una comparación entre los tiempos medidos y la complejidad teórica calculada y extraer conclusiones de la experimentación.

Respecto de las implementaciones, se acepta cualquier lenguaje que permita el cálculo de complejidades según la forma vista en la materia. Además, debe poder compilarse y ejecutarse correctamente en las máquinas de los laboratorios del Departamento de Computación. La cátedra recomienda el uso de C++ o Java, y se sugiere consultar con los docentes la elección de otros lenguajes para la implementación.

La entrada y salida de los programas **deberá hacerse por medio de la entrada y salida estándar del sistema**. No se deben considerar los tiempos de lectura/escritura al medir los tiempos de ejecución de los programas implementados.

Deberá entregarse un informe impreso que desarrolle los puntos mencionados. Por otro lado, deberá entregarse el mismo informe en formato digital acompañado de los códigos fuentes desarrollados e instrucciones de compilación, de ser necesarias. Estos archivos deberán enviarse a la dirección `algo3.dc@gmail.com` con el asunto *"TP 2: Apellido_1, ..., Apellido_n"*, donde n es la cantidad de integrantes del grupo y *Apellido_i* es el apellido del i-ésimo integrante.

Problema 1: Dakkar

Nos encontramos entrenando para competir en una versión particular del Rally Dakkar. En este caso, la carrera consiste en atravesar varias etapas por distintos tipos de terreno y para cada etapa podemos optar por usar uno de tres vehículos disponibles: una BMX, una motocross o un buggy arenero. Obviamente, el vehículo que usemos determinará qué tan rápido concretamos la etapa en cuestión. Lamentablemente, hay un límite para la cantidad de veces que podemos elegir usar la moto y el buggy; no existe límite para el uso de la BMX. Lo bueno, por otro lado, es que sabemos de antemano cuánto tardaríamos en concretar cada etapa para cada uno de los tres posibles transportes.

Se pide escribir un algoritmo que tome los datos de una instancia de este problema e indique la mejor combinación de vehículos a utilizar para minimizar el tiempo de finalización total de la carrera. Siendo n la cantidad de etapas de la carrera y k_m y k_b la cantidad máxima de etapas en las que se puede optar por usar la moto y el buggy respectivamente, se pide que el algoritmo desarrollado tenga una complejidad **temporal** de $\mathbf{O}(n \cdot k_m \cdot k_b)$ y que su complejidad **espacial** sea del orden de $\mathbf{O}(n + k_m \cdot k_b)$. En caso de que haya más de una solución óptima, el algoritmo puede devolver cualquiera de ellas.

Formato de entrada: La primera línea contiene tres enteros positivos, n, k_m y k_b, separados por un espacio, los cuales indican las cantidades de etapas de la carrera, el límite de usos para la moto y el límite de usos para el buggy, respectivamente. A esta línea le siguen n líneas, una para cada etapa, y cada una de ellas contiene tres enteros no negativos, separados por espacios, los cuales indican los tiempos de terminación de dicha etapa usando la BMX, la moto y el buggy, respectivamente.

Formato de salida: La salida debe comenzar con un entero no negativo indicando el tiempo total de finalización de la carrera. Luego de este valor, separado por un espacio, la salida debe contener n enteros, uno para cada etapa de la carrera indicando qué tipo de vehículo se utilizó. Cada uno de estos valores deberá ser 1, 2 o 3, en función de si se utilizó la BMX, la moto o el buggy, respectivamente.

Problema 2: Zombieland II

Afortunadamente, y gracias a sus algoritmos, el ataque coordinado contra los zombis de hace unas semanas fue todo un éxito. Hemos podido limpiar y recuperar la mayoría de las ciudades del país, aunque algunas otras están aun bajo la amenaza zombi. Si bien nuestros recursos bélicos ya casi están agotados, queda aun una luz de esperanza para exterminar de una vez por todas el virus Z (así es como han llamado al virus letal algunos sensacionalistas con mucho poder mediático).

Actualmente, un grupo de soldados se encuentra atrincherado en una de las últimas ciudades plagadas de zombis. Estos soldados se encuentran en una esquina de la ciudad defendiendo a un científico, quien es el único que posee el conocimiento para revertir los efectos del virus Z. El objetivo de la cuadrilla es escoltar al científico hasta un bunker militar ubicado en el centro de la ciudad. Lamentablemente, el equipo sólo puede moverse a pie atravesando las calles de la ciudad, las cuales se encuentran, de más está decirlo, infestadas de zombies. Peor aun, la cuadrilla ha agotado toda munición y los soldados están armados únicamente con sus cuchillos de combate.

Cada uno de los soldados de la cuadrilla militar está altamente entrenado y gracias a eso, el grupo puede atravesar sin ningún problema cualquier calle en la que haya hasta un zombie por soldado. En cambio, si el grupo atraviesa una calle con más de un zombie por soldado, no todo el grupo logrará atravesarla. Es decir, siendo s soldados, atravesar una calle con z zombies costará la muerte de $z - s$ soldados, siempre que $z > s$. Obviamente si $z - s \geq s$ esto significará la muerte de toda la cuadrilla, así también como la del científico y junto con ellos la muerte de toda esperanza de supervivencia de la raza humana. En el caso en que $z \leq s$, el grupo atravesará la calle sin ninguna baja.

Afortunadamente, gracias a nuestro majestuoso enlace satelital, podemos saber exáctamente cuántos zombies hay en cada una de las calles de la ciudad. Conociendo este dato, debería ser posible desarrollar un plan que permita al equipo alcanzar el bunker con la mayor cantidad de sobrevivientes posible (si es que existe algún camino para ello). Nuestro objetivo entonces es exactamente ese: ¡desarrollar un algoritmo que permita hallar dicho camino (si existe) y hacer esto en el menor tiempo posible!

Se sabe que la ciudad tiene la forma clásica de grilla rectangular; es decir, el trazado de la misma está compuesto por una grilla de n calles paralelas en forma horizontal y m calles paralelas en forma

vertical, dando así una grilla de manzanas cuadradas. Se conoce además la cantidad s de soldados que inicialmente posee el grupo y la cantidad de zombies que hay en cada una de las calles de la ciudad. Finalmente, se sabe en qué esquina están atrincherados los soldados y a qué esquina deben llegar para alcanzar la entrada del bunker. Se pide escribir un algoritmo que tome todos estos datos e indique (si es posible) un camino que permita alcanzar el bunker con la mayor cantidad de sobrevivientes posible. El algoritmo debe tener una complejidad temporal de peor caso de $\mathbf{O}(s \cdot n \cdot m)$. En caso de que haya más de una solución óptima, el algoritmo puede devolver cualquiera de ellas.

Formato de entrada: La primera línea contiene tres enteros positivos, n, m y s, separados por un espacio, los cuales indican las cantidaded de calles horizontales y verticales y la cantidad de soldados en la cuadrilla, respectivamente. Las calles horizontales se numeran de 1 a n y las verticales de 1 a m. Luego de esta línea hay otra con 4 enteros positivos I_h, I_v, B_h y B_v, separados por espacios. I_h e I_v son las calles horizontal y vertical, respectivamente, en cuyo cruce se encuentran inicialmente los soldados. B_h y B_v indican de la misma manera la esquina en la cual se encuentra el bunker. A continuación de esta línea, para cada calle horizontal i entre 1 y $n-1$, la entrada contiene un par de líneas con el siguiente formato:

$$\begin{array}{ccccccccc} & h_1^i & & h_2^i & & \dots & & h_{m-1}^i & \\ v_1^i & & v_2^i & & v_3^i & \dots & v_{m-1}^i & & v_m^i \end{array}$$

donde el valor h_j^i es la cantidad de zombies en la calle horizontal i entre las calles verticales j y $j+1$, y el valor v_j^i es la cantidad de zombies en la calle vertical j entre las calles horizontales i e $i+1$. Estos valores pueden estar separados por cualquier cantidad (positiva) de espacios blancos. La entrada finaliza con una línea con el siguiente formato:

$$h_1^n \qquad h_2^n \qquad \dots \qquad h_{m-1}^n$$

representando las cantidades de zombis de la calle horizontal n (con la misma nomenclatura que arriba).

Formato de salida: En caso de haber solución, la primera línea de la salida debe contener un entero positivo indicando la cantidad de soldados que llegaron vivos al bunker. A esta línea deben seguirle una línea por cada esquina por la que pase el camino dado como solución (incluyendo la esquina inicial y la final). Cada esquina se representa con dos enteros, separados por un espacio indicando la calle horizontal y vertical, respectivamente, del cruce en cuestión. En caso de no haber solución, la salida simplemente deberá contener el número 0.

Problema 3: Refinando petróleo

Una empresa petrolera posee pozos de extracción en algunas locaciones en la zona de *Toro Vivo*. El petróleo extraído debe enviarse a refinerías en la zona, aun no construídas. Las refinerías pueden construirse únicamente junto a alguno de los pozos disponibles. Además de las refinerías, pueden construirse largas tuberías para conectar los pozos entre sí, con el objetivo de transportar el petróleo extraído. Cada una de estas tuberías se constituyen de dos canales unidireccionales lo cual permite que el petróleo fluya en ambas direcciones, y no tienen un límite para el caudal transportado. Para evitar pérdidas en el transporte del petróleo, las tuberías no pueden bifurcarse a medio camino entre un pozo y otro.

La empresa necesita construir refinerías y tuberías con el objetivo de que todo el petróleo extraído de los pozos pueda ser refinado en alguna de las refinerías construídas. Es así que cada uno de los pozos debe tener acceso a una refinería, ya sea porque tiene una construída en su misma locación o bien porque existe un camino para transportar el petróleo desde el pozo hasta algún otro pozo que tenga una refinería en su locación (pasando eventualmente por pozos intermedios).

La construcción de una refinería tiene un costo fijo y no hay un límite para la cantidad de refinerías a construir. Por otro lado, el costo de construir una tubería que conecte dos pozos depende de qué pozos se quieran conectar y, dada la geografía del lugar, no todos los pares de pozos pueden conectarse entre sí por medio de tuberías directas.

Se pide escribir un algoritmo que determine un plan de construcción de refinerías y tuberías de forma tal de cumplir con los requisitos de la empresa y minimizando los gastos incurridos por el plan. El plan debe indicar en qué pozos se construirán refinerías y qué pares de pozos se conectarán con tuberías. Se pide que el algoritmo desarrollado tenga una complejidad temporal de peor caso **estrictamente mejor**

que $\mathbf{O}(n^3)$, donde n es la cantidad de pozos del problema. En caso de haber más de un plan óptimo, el algoritmo puede devolver cualquiera de ellos.

Formato de entrada: La primera línea contiene tres enteros positivos, n, m y C, separados por un espacio, donde n indica la cantidad de pozos de la empresa (numerados de 1 a n), m indica la cantidad de pares de pozos entre los cuales es posible construir una potencial tubería y C es el costo de construir una refinería (en cualquiera de los pozos). A esta línea le siguen m líneas, una para cada potencial tubería, y cada una de ellas contiene tres enteros no negativos, separados por espacios; los primeros dos identifican los pozos en cuestión y el tercero indica el costo de construir dicha tubería.

Formato de salida: La salida debe comenzar con una linea con tres enteros K, r y t. El valor K indica el costo total de la solución y los valores r y t indican las cantidades de refinerías y de tuberías a construir, respectivamente. A esta línea le sigue una línea con r enteros, separados por espacios, indicando los números de pozo en donde se construirán las r refinerías. A continuación deberá haber t líneas con dos enteros cada una (separados por un espacio) describiendo las t tuberías a construir. Para cada tubería, los enteros especifican los números de pozo entre los que se construirá dicha tubería.