Trabajo Práctico 1: Técnicas Algorítmicas

Compilado: 23 de agosto de 2023

Fecha de entrega: 13 de septiembre de 2023 (antes de las 23:59)

Fecha de reentrega: 4 de diciembre de 2023 (antes de las 23:59)

Este trabajo práctico consta de cuatro ejercicios que pueden ser resueltos mediante la aplicación de las técnicas algorítmicas vistas en clase. Para cada uno de ellos se deberá diseñar e implementar un algoritmo que lo resuelva, y es necesario que dicha implementación supere una prueba de eficiencia (tanto temporal como espacial) en un juez online.

En todos los ejercicios la instancia se leerá de la entrada estándar y la solución se imprimirá en la salida estándar. Junto a los enunciados se incluyen algunos casos de test que sirven como ejemplo tanto del formato de salida como del problema en cuestión. Está **terminantemente prohibida** la copia de código, ya sea de otro grupo o de internet. La detección de plagio puede resultar en la desaprobación directa de la materia o en otras medidas más graves.

El trabajo práctico se debe realizar en grupos de a 3. Para aprobarlo 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 aprobados en la primera entrega. Asimismo, para aprobar cada ejercicio es necesario que la implementación del algoritmo propuesto pase la prueba del juez online y que este sea entregado a los docentes para su revisión.

Formato de entrega La entrega se realizará a través del campus. Se deberá entregar una carpeta comprimida con el código de la solución de los cuatro problemas. El nombre de estos archivos debe ser el de los apellidos de las personas integrantes del grupo separados por guiones.

Ejercicio 1: Sendero

Tuki recibió como regalo de cumpleaños un tablero blanco de N filas y M columnas, que contiene 4 tipos distintos de piezas:

- Piezas vacías, que notaremos como #.
- Piezas con forma de I.
- Piezas con forma de L.
- Piezas con forma de +.

Cada pieza está fija en su posición, y puede rotarse en múltiplos de 90°.

Como Tuki no entiende bien el objetivo del tablero, decide jugar a rotar las piezas para formar caminos. Por ejemplo, si considera la siguiente porción del tablero

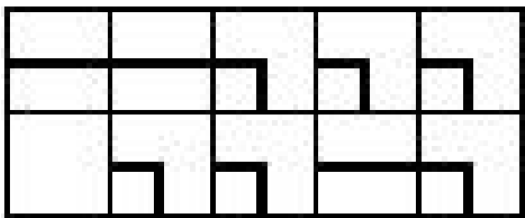

Figura 1: Un tablero de 2 filas y 5 columnas ($N = 2$ y $M = 5$).

puede rotar 180° la pieza en forma de L ubicada en la segunda fila y tercera columna para formar un camino desde la esquina superior izquierda a la inferior derecha:

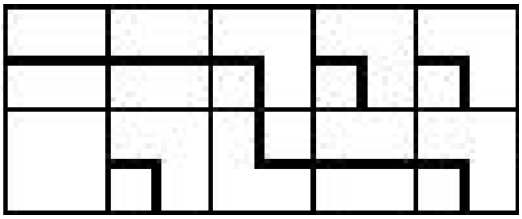

Figura 2: El mismo tablero que en la figura 1, pero tras rotar la pieza en forma de L ubicada en la fila 2 y la columna 3 se obtiene un camino desde la esquina superior izquierda a la esquina inferior derecha.

Tuki se pregunta: ¿Será posible hacer esto mismo, pero para todo el tablero? Es decir, ¿Se pueden girar las piezas de forma tal de obtener un camino que vaya de la esquina superior izquierda a la esquina inferior derecha del tablero? ¿Qué tan corto y largo puede ser ese camino?

Descripción de una instancia

La entrada contiene múltiples casos de test. La primera línea consiste de un número c indicando su cantidad.

Luego siguen c casos de test. Cada uno comienza con dos enteros N y M, denotando la cantidad de filas y la cantidad de columnas del tablero, respectivamente. Se garantiza que $N \times M \leq 120$.

Después sigue la descripción del tablero de N filas y M columnas. En cada posición del mismo habrá un símbolo indicando el tipo de pieza. Se garantiza que la pieza ubicada en la esquina superior izquierda no es de la forma #.

Descripción de la salida

Para cada caso de test se debe indicar si es posible formar un camino desde la esquina superior izquierda a la inferior derecha rotando las piezas. En caso de que sea posible se debe imprimir el mensaje `POSIBLE` y a continuación la mínima y máxima longitud del camino, teniendo en cuenta que este no puede repetir piezas. En caso de que sea imposible se debe imprimir el mensaje `IMPOSIBLE`.

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

Entrada de ejemplo	Salida esperada de ejemplo
3 3 3 ILI #IL +L+ 2 2 +# #L 4 4 LI+L ++IL I#+# LI++	POSIBLE 4 4 IMPOSIBLE POSIBLE 6 12

Ejercicio 2: Correcciones pendientes

Llegado el día de entrega de los parciales de Algo3 Carolina se percata, durante la misma entrega, de que algunos exámenes habían quedado al fondo de su mochila y que, por lo tanto, no están corregidos.

Hay N parciales sin revisar, y tras una breve inspección de los mismos dedujo una estimación M_i de los minutos que le toma corregir cada uno. Aparte, conociendo a los estudiantes, infirió un *coeficiente de descontento* asociado a cada uno de ellos: por cada minuto que al estudiante i no se le entrega su parcial, este incrementa su descontento en C_i unidades.

Teniendo en cuenta que Carolina puede corregir los parciales en cualquier orden, debemos indicarle cual es el mínimo valor de descontento general (es decir, de la suma de los descontentos de cada uno de los alumnos) que puede alcanzar.

Descripción de una instancia

La entrada contiene múltiples casos de test. La primera línea consiste de un número c indicando su cantidad.

Cada caso de test comienza con una línea con un entero N con $1 \leq N \leq 10^7$, indicando la cantidad de parciales sin corregir. Luego sigue una nueva línea con los N valores M_i, con $1 \leq M_i \leq 10^7$. Después hay otra línea más con los N valores C_i, también con $1 \leq C_i \leq 10^7$.

Descripción de la salida

Para cada caso de test se debe imprimir un único entero indicando el mínimo valor de descontento general que se puede alcanzar. Como este valor puede llegar a ser muy alto se debe imprimir únicamente su valor módulo $10^9 + 7$.

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

Entrada de ejemplo	Salida esperada de ejemplo
3 3 1 2 3 2 2 2 4 2 1 3 2 1 2 2 3 2 100000 100000 10000 20000	20 31 999999979

En el primer caso de test se puede demostrar que la estrategia óptima consisten en corregir primero el primer parcial, luego el segundo, y finalmente el tercero. En ese caso, el descontento general se obtiene sumando:

- El descontento del primer alumno, que es 2 (pues solo esperó un minuto y su coeficiente de descontento es 2).
- El descontento del segundo alumno, que es 6 (pues espera 3 minutos).
- El descontento del tercer alumno, que es 12 (pues espera 6 minutos).

Por lo tanto, el menor descontento general alcanzable es 20.

En el tercer caso de test la estrategia óptima consiste en corregir primero el segundo parcial. El descontento general asociado a este orden es $100000 \times 20000 + (100000 + 100000) \times 10000 = 4000000000 = 999999979 \mod (10^9 + 7)$.

Ejercicio 3: Saldos sospechosos

En el marco de un proyecto de colaboración con la AFIP se le presenta al equipo de algoritmos del DC la siguiente situación: una empresa decidió, con el objetivo de evitar declarar su patrimonio, borrar los signos de los valores en su libro de cuentas. Es decir, si en el libro de cuentas figuraban las entradas `500, -700` indicando una venta por 500 pesos y una compra por 700, ahora solo se lee `500, 700`, perdiendo luego la distinción entre gasto y venta que acarreaba el signo.

No obstante, la empresa olvidó borrar el signo de los saldos finales. Por lo tanto, en el ejemplo anterior, se conoce que la suma de las entradas es `-200`. Notemos que con esta información es posible identificar que el número `500` se corresponde con una venta, y que el número `700` se corresponde con un gasto.

Se le pide al DC desarrollar un programa que permita indicar, para cada una de las entradas del libro de cuentas, si esta es inequívocamente un gasto o una venta, o si bien puede ser cualquiera de las dos.

Descripción de una instancia

La entrada contiene múltiples casos de test. La primera línea consiste de un número c indicando su cantidad.

La primera línea de cada caso contiene dos números: un entero N indicando la cantidad de entradas del libro y el saldo final W, con $1 \leq N \leq 10^2$ y $-10^5 \leq W \leq 10^5$. En la siguiente línea se encuentran los N valores x_i de cada fila del libro de cuentas, con $1 \leq x_i \leq 10^4$. Cada x_i es un múltiplo de 100.

Descripción de la salida

Para cada caso de test se debe imprimir una línea con N caracteres indicando la situación de cada entrada. Si la i-ésima entrada del libro es un gasto, entonces el i-ésimo caracter debe ser un símbolo $-$, mientras que si es una venta entonces debe ser un símbolo $+$. Por otro lado, si puede ser tanto uno como lo otro, se debe imprimir el símbolo ?.

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

Entrada de ejemplo	Salida esperada de ejemplo
2 2 -200 500 700 4 400 500 700 700 100	+- +??-

Ejercicio 4: Choripanes

En el marco de otro proyecto de colaboración el Gobierno de la Ciudad le pide ayuda al DC para gestionar la provisión de materia prima a los puestos de choripanes de la costanera. Se tomó la decisión de modelar la misma mediante una línea recta, describiendo la posición de los N puestos a lo largo de esta con números naturales.

El Gobierno está interesado en colocar K proveedurías, y puede poner estas en cualquier posición de la costanera. Se define una *disposición* de las K proveedurías como una función $p : \{1, 2, \ldots, K\} \to \mathbb{R}$ donde $p(i)$ denota la posición en la que se coloca la i-ésima proveeduría. Se asume que $p(i) \leq p(i+1)$ para $1 \leq i < K$ (es decir, la $i+1$-ésima proveeduría se coloca por delante de la i-ésima). Para optimizar la colocación se define el costo asociado a una disposición p como

$$C(p) = \sum_{i=1}^{N} \min_{1 \leq j \leq K} |x_i - p(j)| \tag{1}$$

donde x_i denota la posición del i-ésimo puesto. Intuitivamente, el costo se define como la suma de las distancias de cada restaurante a su proveeduría más cercana.

Se quiere conocer qué tan bajo puede ser este costo $C(\cdot)$, y una de las disposiciones que alcance este valor óptimo. Puntualmente, se quiere aquella disposición óptima que sea menor lexicográficamente: dadas dos disposiciones p y p' decimos que p es lexicográficamente menor a p' si existe j tal que $p(j) < p'(j)$ y $p(i) = p'(i)$ para $1 \leq i < j$. Es decir, p es menor lexicográficamente a p' si la cadena que se obtiene escribiendo las disposiciones que indica p es menor lexicográficamente a la de p'.[1].

[1]Por ejemplo, si p coloca la primera proveeduría en la posición 1 y la segunda en la posición 9 ($p(1) = 3, p(2) = 9$) y p' coloca la primera en la posición 1 y la segunda en la 10 ($p'(1) = 1, p'(2) = 10$) entonces p es lexicográficamente menor que p', pues la cadena `1 9` es menor lexicográficamente que `1 10`

Descripción de una instancia

La entrada contiene múltiples casos de test. La primera línea consiste de un número c indicando su cantidad.

La primera línea de cada caso contiene dos enteros N y K, indicando respectivamente la cantidad de puestos y la cantidad de proveedurías que se quieren colocar. Vale que $1 \leq K \leq N \leq 10^2$. Luego sigue una línea con N enteros x_i, con $1 \leq x_i \leq 10^7$, denotando las posiciones de los distintos puestos. Esta lista de posiciones está ordenada.

Descripción de la salida

Para cada caso de test se debe imprimir una línea indicando el mínimo valor que puede alcanzar el costo de la distribución de las proveedurías (i.e. la expresión 1). Luego se debe imprimir otra línea, describiendo de forma ordenada la disposición que alcanza este valor y es lexicográficamente mínima.

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

Entrada de ejemplo	Salida esperada de ejemplo
3 4 2 1 5 15 20 5 1 4 5 6 7 8 4 1 4 5 6 7	9 1 15 6 6 4 5

En el primer caso de test el costo óptimo es 9, y la disposición óptima consiste en colocar la primera proveeduría en la posición 1 y la segunda en la posición 15. También se obtiene una disposición óptima colocando la segunda proveeduría en la posición 19, pero esta no es lexicográficamente mínima.

En el segundo caso de test la disposición óptima consiste en colocar la única proveeduría disponible en la posición 6, resultando en un costo de 6.