ISIS 1105 Diseño y Análisis de Algoritmos
Semestre 2018-20. Proyecto
Profs. R. Cardoso / J. Duitama / N. Sánchez
Entrega: Martes, Diciembre 11 de 2018, 8:00 (por Sicua+)
Trabajo individual o por parejas (de la misma sección)

0 OBJETIVOS

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

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

1 CONDICIONES GENERALES

Hay tres problemas para resolver mediante soluciones implementadas en *Java*.

Para cada problema se pide:
- Análisis temporal y espacial.
- Una solución java.

2 PROBLEMAS

A Subarreglo casi ascendente más largo

Para un arreglo `a[0..n-1]:int`, un *subarreglo casi ascendente* de longitud `q-p` es un subarreglo `a[p..q-1]`, $0 \leq p < q \leq n$, para el cual existe a lo sumo un valor `i` en el rango $p < i < q$, que satisface `a[i-1]>a[i]`.

Problema

Se quiere desarrollar un programa que determine la longitud máxima dentro de todos los subarreglos casi ascendentes.

Ejemplo:

Para
```
a = [0,1,2,1,3,4,5,2,3,5]
n = 10
```

son casi ascendentes los subarreglos `a[0..0],a[0..3],a[0..6],a[1..6],a[3..9]` , entre otros. El subarreglo ascendente más largo mide `7`.

B Inversión en bolsa

Juan puede invertir en la bolsa A o en la B. Las inversiones se dan en un intervalo de tiempo (números naturales) $0..n-1$, con $n>0$. En el tiempo 0 dispone de un capital $C>0$. En cada instante de tiempo, Juan toma decisiones de inversión. Él tiene como regla invertir todo su capital en la bolsa que más rendimiento le dé y que, en todo caso, no le signifique pérdidas de capital (es decir, si se prevén pérdidas, Juan no invierte).

Un estudio de mercados ha establecido una rentabilidad esperada para las inversiones en cada una de las bolsas, de modo que Juan cuenta con dos arreglos r_A, r_B`:[0..n-1]:int` que estiman la rentabilidad en cada una de las bolsas. La idea es que si, al empezar el tiempo k, Juan invierte z en la bolsa X, al empezar el tiempo $k+1$ deberá tener $\lfloor z*(1+r_X/100) \rfloor$ como capital, $0 \leq k < n$, $z>0$, $X \in \{A,B\}$.

La bolsa A es especial, en el sentido de que tiene restricciones para hacer inversiones. De hecho, si en el tiempo k, $0 \leq k < n-1$ se ha invertido capital en la bolsa A, no se podrá invertir en el tiempo $k+1$ en esta bolsa.

Problema

Se quiere determinar cuál es el máximo capital que Juan podría esperar a tener en el tiempo n.

Ejemplo:

Supóngase $C = 1000$, $n=3$.

```
a = [-2,1,2]
b = [1,0,3]
```

Sean $z_A[k]$ y $z_B[k]$ las inversiones que se hacen en cada bolsa en el tiempo k. Las inversiones debieron hacerse en la siguiente forma (la columna C muestra la evolución del capital):

	C	r_A	r_B	z_a	z_b
0	1000	-2	1	0	1000
1	1010	1	0	1010	0
2	1020	2	3	0	1020
3	1050				

y, en el tiempo 3 el capital es 1050.

C Velocidad observada estimada

Para controlar el exceso de velocidad de los vehículos, la alcaldía de la ciudad compró una gran cantidad de sensores de baja precisión. Precisamente, por esta característica los sensores son muy baratos, y en un punto de control de velocidad se puede instalar una gran cantidad de ellos. La idea es que, a partir de todas las mediciones entregadas se estime la velocidad de los vehículos.

Una *estimación* es un par `(a,b)`**`:nat × nat`**, con $a \leq b$. La velocidad de un vehículo dado es medida por cada uno de $n>1$ sensores y el i-simo sensor, $1 \leq i \leq n$, entrega una estimación E_i. La *velocidad observada estimada* (*voe*) se define como un consenso de las estimaciones obtenidas por los sensores. Más exactamente, para $A \subseteq \{1,\ldots,n\}$, con $|A|>0$ sea

$$E_A = (\cap i \mid i \in A : E_i)$$

Si $E_A \neq \emptyset$, debe ser claro que E_A es también una estimación, digamos, (x_A, y_A). La voe se define como $\lfloor (x_{A^*}+y_{A^*})/2 \rfloor$, donde A^* es el subconjunto más grande de $\{1,\ldots,n\}$, con $|A^*|>0$, tal que $E_{A^*} \neq \emptyset$. Si hay empates, para definir A^*, se prefieren subconjuntos con menor valor de x_A y, de persistir el empate, se prefieren subconjuntos con menor valor de y_A.

Problema

Dado un conjunto de estimaciones E_i, $1 \leq i \leq n$, se quiere determinar la velocidad estimada observada (voe) correspondiente.

Ejemplos:

Dadas las estimaciones `E1 = {(12,16),(14,17),(11,15)}`. Obsérvese que la intersección de todas las estimaciones es no vacía y es igual a la estimación `(14,15)`. La voe es `14`.

Por otro lado, si se tienen las estimaciones `E2 = {(21,26),(25,30),(27,32)}`, la intersección entre las tres estimaciones es vacía, pero hay dos con intersección no vacía. Concretamente, las intersecciones no vacías de dos estimaciones son las estimaciones `(25,26)` y `(27,30)` y, de acuerdo con la convención, se prefiere `(25,26)`. La voe es `25`.

3 ENTRADA / 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, para cada problema, se establecen parámetros que definen su tamaño y formato de lectura de los datos, tanto de entrada como de salida.

3.1 Problema A: Subarreglo ascendente más largo

- Tamaño del problema: n
- Condiciones de los casos de prueba: $0<n<10^5$.

Descripción de la entrada

La entrada contiene varios casos de prueba. Cada caso de prueba se describe con una línea de texto de la forma

n

donde n es un número natural positivo, que determina el tamaño del arreglo a. Después hay una línea de la forma (cada a_k es un número entero, $0 \leq a_{k<} 10^6$):

$a_0\ a_0\ \ldots\ a_{n-1}$

El fin de la entrada se indica con una línea que contiene un `0`.

Descripción de la salida

Por cada caso de prueba para resolver, imprimir una línea de respuesta que contenga un número natural correspondiente al valor de la longitud del subarreglo casi ascendente más largo.

Ejemplos de entrada / salida

Entrada	Salida
4 1 4 2 1 10 0 1 2 1 3 4 5 2 3 5 0	3 7

3.2 Problema B: Inversión en bolsa

- Tamaño del problema: n
- Condiciones de los casos de prueba: $0<n<10^5$

Descripción de la entrada

La entrada contiene varios casos de prueba. Cada caso de prueba se describe con una línea de texto de la forma

n C

donde n es un número natural positivo, que determina el intervalo de tiempo de las inversiones y el tamaño de los arreglos r_A y r_B. Después hay dos líneas de la forma (cada a_k, b_k es un número entero, $-10^3 \leq a_k, b_k < 10^3$)

a_0 a_1 … a_{n-1}

b_0 b_1 … b_{n-1}

El fin de la entrada se indica con una línea de texto de la forma

0 0

Descripción de la salida

La salida debe tener exactamente una línea con un número entero que represente el máximo capital que se puede obtener en el tiempo n.

Ejemplo de entrada / salida

Entrada	Salida
3 1000 -2 1 2 1 0 3 3 1000 2 10 4 1 1 3 0 0	1050 1144

3.3 Problema C: Velocidad observada estimada

- Tamaño del problema: n
- Condiciones de los casos de prueba: $0<n<10^4$

Descripción de la entrada

La entrada contiene varios casos de prueba. Cada caso de prueba se describe con una línea de texto de la forma
n

donde n es un número natural positivo, que determina el número de sensores. Después hay una línea con $2n$ números naturales (cada a_k, b_k es un número natural, $0 \le a_k \le b_k < 500$):
$a_1\ b_1\ a_2\ b_2\ \ldots\ a_n\ b_n$

donde (a_k,b_k) es la estimación del k-simo sensor, $1 \le k \le n$.

El fin de la entrada se indica con una línea que contiene un 0.

Descripción de la salida

La salida debe contener una línea con un número natural que representa la voe correspondiente a los datos de la entrada.

Ejemplo de entrada / salida

Entrada	Salida
3 12 16 14 17 11 15 3 21 26 25 30 27 32 2 25 30 40 50 0	14 25 27

4 ENTREGABLES

El proyecto puede desarrollarse por grupos de uno o dos estudiantes de la misma sección. La entrega se hace por Sicua+ (una sola entrega por grupo de trabajo).

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

4.1 Archivos fuente de soluciones propuestas

Todos los programas implementados en *Java* deben compilar en *JDK 8*.

Para el problema `X`, siendo $X \in \{A,B,C\}$:

- Entregar un archivo *Java* (`.java`) 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 `ProblemaX.java` el archivo de la solución que se presente.

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

4.2 Archivos que documentan soluciones propuestas

Toda solución propuesta debe acompañarse de un archivo que la documente, con extensión `.doc`, `.docx` o `.pdf`. El nombre del archivo debe ser el mismo del código *Java* correspondiente. Por ejemplo, si incluyó un archivo `ProblemaB.java`, como solución para el problema B, debe incluirse un archivo `ProblemaB.docx` (o `ProblemaB.pdf`) que lo documente.

Un archivo de documentación debe contener los siguientes elementos:

0 Identificación
Nombre de autor(es)
Identificación 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.
Deseable:
Anotación (contexto, pre-, poscondición, …) para cada subrutina o método que se use.

2 Análisis de complejidades espacial y temporal
Cálculo de complejidades y explicación de las mismas. Debe realizarse un análisis para cada solución entregada.

3 Comentarios finales
Comentarios al desempeño observado de la solución.

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.

No describa un algoritmo con código GCL a menos que lo considere necesario para explicarlo con claridad. Y, si lo hace, asegúrese de incluir aserciones explicativas, fáciles de leer y de comprender.