ISIS 1105 Diseño y Análisis de Algoritmos
Semestre 2012-20. Proyecto
Prof. Alejandro Sotelo
Entrega: Viernes, Noviembre 30, 2012, 18:00 (por *Sicua+*)
Trabajo individual o por parejas (dentro de una misma sección)

0 OBJETIVOS

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

Por ejercicio, se premiarán las mejores soluciones y se castigarán las peores, en cuanto a eficiencia. De igual manera, serán penalizadas las soluciones ineficaces.

1 CONDICIONES GENERALES

Deben respetarse las siguientes reglas:
- El proyecto debe desarrollarse en grupos de máximo dos personas.
- El proyecto debe entregarse antes de la fecha de entrega, que es inaplazable.
- Toda comunicación con otro grupo debe ser informada en el documento entregado.
- El desarrollo (código fuente y documentación) debe ser de autoría propia de cada grupo.
- Cualquier comunicación con otro grupo o con otra persona debe ser informada explícitamente.

Hay diez problemas para resolver mediante soluciones implementadas en *Java*, que incluyen ocho ejercicios obligatorios y dos opcionales. Para cada uno de éstos se proporciona:
- Una *plantilla* `.java` que debe utilizarse para programar la solución entregada.
- Unos *casos de prueba de ejemplo* que sirven para probar la solución entregada.
- El *porcentaje* de participación dentro de la nota definitiva del proyecto.
- Un *umbral* de tiempo para la ejecución de la solución entregada.
- Una *complejidad temporal límite* que debe respetarse en la solución entregada.

#	Identificador	Porcentaje	Umbral	Complejidad temporal límite
A	`cookie`	10% [*bono*]	5000ms	$O(C{\cdot}K^3)$ por caso de prueba
B	`movie`	15%	4000ms	$O(M{\cdot}Q{\cdot}L^2)$, donde $L = max\{l_1, l_2, \ldots, l_M\}$.
C	`cinema`	15%	1000ms	$O(R{\cdot}C)$ por caso de prueba
D	`digsum`	15%	1000ms	$O(log_2(N))$ por caso de prueba
E	`lcd`	10%	4000ms	$O(M{\cdot}N)$ por caso de prueba
F	`parabox`	15%	2000ms	$O(N{\cdot}B)$ por caso de prueba
G	`garden`	10% [*bono*]	1000ms	$O(1)$ por caso de prueba
H	`teamface`	10%	6000ms	$O(T{\cdot}N^2)$ por caso de prueba
I	`imperial`	10%	3000ms	$O(N{\cdot}M)$ por caso de prueba, donde N,M es el tamaño de ambas cadenas
J	`lazyprof`	10%	4000ms	$O(N{\cdot}S + N{\cdot}log_2(N))$ por caso de prueba

Tabla 1: *Porcentaje, umbral de tiempo y complejidad temporal límite para cada ejercicio.*

Para cada ejercicio, la complejidad temporal límite se define con respecto a un tamaño de problema descrito usando las variables definidas en su enunciado. Cada ejercicio debe implementarse en un solo archivo con extensión `.java`, que debe compilar en JDK 6 o inferior. Para cada problema se debe entregar:
- Un archivo `.java` con la implementación escrita en Java, leyendo los datos desde la entrada estándar (`System.in`) e imprimiendo las respuestas en la salida estándar (`System.out`).
- Un documento `.docx` describiendo la solución y analizando su complejidad temporal y espacial.

Se espera que cada solución tenga una complejidad mejor o igual que la establecida como límite, que arroje las respuestas esperadas, y que ejecute en un tiempo menor o igual que el umbral definido en la máquina oficial utilizada para los juzgamientos. Para estimar el factor de conversión entre la máquina oficial y su propia máquina, puede usar el siguiente programa:

```
public class Fibonacci {
  public static void main(String[] args) {
    for (int n=0; n<=50; n++) {
      long start=System.nanoTime();
      fib(n);
      System.out.println(n+"\t"+(System.nanoTime()-start)/1000000L+"ms");
    }
  }
  static long fib(int n) {return n<=1?n:fib(n-2)+fib(n-1);}
}
```

Output:

```
...
41     2222ms
42     3590ms
43     5824ms
44     9441ms
45    16565ms
46    25400ms
47    40243ms
48    64914ms
49   105447ms
50   170519ms
```

2 PROBLEMAS

Los problemas son los de la *XXVI Maratón Nacional de Programación ACIS REDIS 2012*, llevada a cabo el 20 de octubre de 2012. La tabla de puntuaciones de la competencia puede consultarse en:

http://www.acis.org.co/index.php?id=556

Los enunciados están publicados en la página del Juez Online de la Universidad de Valladolid (*UVa*):

http://uva.onlinejudge.org > *Contests* > *Past Contests* > (*Id=311*) *XXVI Colombian Programming Contest*

También es posible registrarse en la página de la *UVa* para el juzgamiento automático de las soluciones que se vayan desarrollando (esto es voluntario y no se evaluará). Los ejercicios de la competencia nacional de programación aparecen registrados en la *UVa* bajo la siguiente ruta:

http://uva.onlinejudge.org > *Browse Problems* > *Contest Volumes* > *Volume CXXV* > *Ejercicios 12514 a 12523*

Para probar las soluciones en el juez automático de la *UVa* hay que seguir ciertas indicaciones, ligeramente diferentes a las descritas en este enunciado:

http://uva.onlinejudge.org > *Additional Information* > *Submission Specification*

Así pues, las soluciones en *UVa* se envían en el formato de *UVa*, y las soluciones a entregar para este proyecto se envían en el formato descrito en el presente documento.

3 INSTRUCCIONES

Las soluciones que se construyan para los problemas deben soportarse en el material que sirve de referencia para éstas. Más exactamente, para cada uno de los problemas se suministran los siguientes archivos (suponiendo que *id* es el identificador del problema):

- *id*.java: Archivo *Java* con la plantilla que debe usarse para implementar el código fuente de la solución.
- *id*.in: Archivo en texto plano con los casos de entrada de ejemplo.
- *id*.out: Archivo en texto plano con las respuestas de los casos de entrada de ejemplo.
- *id*.docx: Archivo en formato .docx que debe ser diligenciado con la documentación de la solución.

			Casos de prueba de ejemplo	
#	Identificador	Plantilla	Entrada	Salida
A	cookie	cookie.java	cookie.in	cookie.out
B	movie	movie.java	movie.in	movie.out
C	cinema	cinema.java	cinema.in	cinema.out
D	digsum	digsum.java	digsum.in	digsum.out
E	lcd	lcd.java	lcd.in	lcd.out
F	parabox	parabox.java	parabox.in	parabox.out
G	garden	garden.java	garden.in	garden.out
H	teamface	teamface.java	teamface.in	teamface.out
I	imperial	imperial.java	imperial.in	imperial.out
J	lazyprof	lazyprof.java	lazyprof.in	lazyprof.out

Tabla 2: *Archivos de referencia provistos para cada ejercicio.*

Además, cada uno de los problemas tiene casos de prueba oficiales que serán usados para probar las distintas soluciones. Estos casos de prueba no son públicos, pero son similares a los dados como ejemplo. Con respecto a los casos de ejemplo, los oficiales podrían llegar a ser muchos más, podrían ser más grandes, y podrían contener casos de frontera no presentes en los de ejemplo. Por lo tanto, es posible que una solución resuelva satisfactoriamente los casos de ejemplo pero no los oficiales, antes de que se alcance el umbral de tiempo asignado para su ejecución. Esto pues, los casos de prueba sirven sólo de referencia.

Para resolver un ejercicio basta implementar su código fuente en el archivo *id*.java y diligenciar su documentación relacionada en el archivo *id*.docx. Cualquier otro archivo adicional será descartado, y por ende, no será tenido en cuenta en la calificación.

El archivo *id*.java debe contener la implementación de la solución del ejercicio con identificador *id*, con las siguientes consideraciones:

- Debe estar almacenado usando la codificación de caracteres ISO-8859-1 (ANSI).
- No debe tener ningún paquete declarado con alguna sentencia del estilo "package …;".
- Debe declarar una clase con la sentencia "public class *id*", tal y como aparece en la plantilla.
- Debe leer los datos desde entrada estándar (System.in) así esté dicho lo contrario en el enunciado.
- Debe escribir las respuestas en la salida estándar (System.out), con el separador de línea "\r\n".

Nótese que, si bien puede utilizarse un *IDE* como *Eclipse* durante el desarrollo del proyecto, la entrega requiere incluir sólo un archivo .java por cada solución. El formato presentado en la plantilla debe seguirse como modelo de presentación de la solución que se implemente, obligatoriamente.

Con respecto al documento *id*.docx:

- Debe estar debidamente marcado con el nombre completo de los integrantes y el *código único* del grupo (véase la sección que describe los entregables para la descripción del *código único*).
- Debe tener una sección donde se describa el algoritmo (técnicas usadas, estructuras de datos, etc.).
- Debe tener una sección donde se analice la complejidad temporal y espacial del algoritmo.

#	Identificador	Implementación	Documentación
A	cookie	cookie.java	cookie.docx
B	movie	movie.java	movie.docx
C	cinema	cinema.java	cinema.docx
D	digsum	digsum.java	digsum.docx
E	lcd	lcd.java	lcd.docx
F	parabox	parabox.java	parabox.docx
G	garden	garden.java	garden.docx
H	teamface	teamface.java	teamface.docx
I	imperial	imperial.java	imperial.docx
J	lazyprof	lazyprof.java	lazyprof.docx

Tabla 3: *Archivos a entregar para cada ejercicio.*

Es decir, cada solución implementada debe acompañarse de un archivo que la documente, cuya extensión sea .docx y cuyo nombre coincida con el del .java correspondiente. El contenido del documento debe respetar el formato definido en la plantilla .docx dada como referencia. A grandes rasgos, la descripción del algoritmo debe explicar informalmente su operación, las decisiones de diseño que se tuvieron en cuenta durante su desarrollo, las técnicas de programación que se usaron, y las estructuras de datos involucradas. El análisis de complejidad temporal y espacial debe ser minucioso, y debe estar dado en términos de las variables definidas en cada problema. Además de exhibir explícitamente cada una de las complejidades, debe explicarse detalladamente cómo se calcularon. Para el análisis de complejidad espacial tenga en cuenta el espacio ocupado por las variables que almacenan los datos de entrada.

Finalmente, con respecto a la solución implementada en el archivo *id*.java:

- Debe estar escrita en el lenguaje de programación *Java*.
- Debe tener un comentario al principio, indicando el nombre completo de los integrantes.
- Su código fuente debe compilar y ejecutar en *JDK 6* (*Java Version 1.6.0_35* o inferior).
- Su complejidad temporal (por cada caso de prueba) debe ser mejor o igual que la complejidad límite.
- Debe ejecutar en un tiempo menor o igual que el establecido como umbral.
- No debe presentar errores durante su ejecución sobre los casos de prueba oficiales.
- Debe pasar el 100% de los casos de prueba oficiales, presentando la misma salida que el .out.

Es necesario que se respeten al pie de la letra todas las reglas establecidas, pues un proceso semiautomático será el que determine la eficacia y eficiencia de cada una de las soluciones. En particular, es importante que el cambio de línea sea "\r\n" y no "\n". Serán penalizadas las soluciones que no tengan una complejidad temporal mejor o igual que el límite, que no arrojen las respuestas deseadas en el 100% de los casos de prueba oficiales, y/o que no ejecuten dentro del tiempo definido como umbral.

Cada solución que se implemente debe leer y escribir datos por consola. Las siguientes dos líneas de comando se pueden utilizar para compilar y ejecutar la solución del ejercicio con identificador *id*, leyendo y escribiendo datos por la consola, suponiendo que el JDK está instalado y que el archivo que contiene el código fuente (`.java`) está ubicado en el directorio actual:

```
javac id.java
java –cp . id
```

Por otro lado, para leer la entrada desde el archivo `.in` y escribir en la consola, se pueden usar las siguientes dos líneas de comando, suponiendo que la entrada (`.in`) está ubicada en el directorio actual:

```
javac id.java
java –cp . id < id.in
```

Finalmente, para leer la entrada desde el archivo `.in` y escribir la salida en un archivo `.sol`:

```
javac id.java
java –cp . id < id.in > id.sol
```

Sería útil probar cada solución leyendo los datos desde el archivo `.in` y escribiendo las respuestas en un archivo `.sol`, para luego comprobar que el archivo de salida *id*`.sol` sea idéntico al archivo de referencia *id*`.out` (revisando el contenido manualmente, o comparando sus MD5s, por ejemplo). De esta manera, los archivos `.in` y `.out` suministrados pueden aprovecharse durante el desarrollo del proyecto para probar las nuevas soluciones, buscando su eficacia y su eficiencia. Sin embargo, la evaluación de las soluciones presentadas usará casos de prueba distintos para validar la eficacia y la eficiencia de las soluciones entregadas. Tales casos de prueba estarán especialmente diseñados para evaluar casos de borde y casos generales, donde el tamaño del problema satisface los umbrales definidos en cada enunciado.

5 ENTREGABLES

El proyecto debe desarrollarse en grupos de uno a dos estudiantes de una misma sección. Los grupos deben registrarse por correo electrónico antes del martes 13 de septiembre de 2012 a las 23:55 UTC -5. Para esto, se debe enviar un correo electrónico a `a-sotelo@uniandes.edu.co` cuyo asunto sea "`Inscripción grupo ISIS1105 2012-20`" y cuyo contenido indique el código de estudiante y el nombre completo de cada uno de los integrantes. Como respuesta, cada grupo recibirá un *código único* de la forma `GPS3_`*xx*, indicando que se está creando explícitamente un grupo con identificador *xx* en la sección `3` de la materia `ISIS1105` en el período `2012-20`, donde *xx* es un auto-numérico que va entregándose por orden de llegada. El código único de la forma `GPS3_`*xx* debe ponerse en el encabezado de cada uno de los archivos `.docx` entregados como documentación y al principio de cada uno de los archivos `.java`.

Como se dijo anteriormente, para resolver un ejercicio basta implementar su código fuente en el archivo *id*`.java` y diligenciar su documentación relacionada en el archivo *id*`.docx`. Los archivos `.java` y `.docx` correspondientes a los ejercicios que no se resuelvan deben ser eliminados completamente. De esta manera, se espera que el número de archivos entregados sea exactamente el doble del número de problemas resueltos, pues para cada problema resuelto con identificador *id*, debe haber exactamente un archivo con nombre *id*`.java` y un archivo con nombre *id*`.docx`.

Todos los archivos a entregar deben ser ubicados en un directorio con nombre `GPS3_`*xx*, donde *xx* está definido de acuerdo al código único entregado al grupo. Tal directorio debe ser empaquetado en formato `zip`, obteniendo un archivo con nombre `GPS3_`*xx*`.zip` que al descomprimir debe arrojar un directorio con nombre `GPS3_`*xx*. Finalmente, ese archivo es el que debe ser entregado en *Sicua*+ antes del vencimiento de la fecha de entrega. Uno solo de los integrantes del grupo debe subir el archivo a *Sicua*+.

Cualquier incumplimiento del formato descrito o de la fecha de entrega, será penalizada proporcionalmente a la gravedad de la falta.