Analisis y Diseño de Algoritmos — Práctica 2

Definición del problema

Sea $\sum$ un alfabeto de símbolos, y sean s y r dos secuencias de cero o más símbolos de $\sum$ (es decir, $s,r \in \sum^*$). Sea la longitud de una cadena s (que escribiremos $|s|$) el número de símbolos que la forman. Por convenio, numeraremos los distintos caracteres de una cadena de uno en adelante ($s = s_1\ s_2\ \ldots\ s_{|n|}$). Decimos que la cadena s es una *supersecuencia* de la cadena r (lo cuál escribiremos $s \rangle\rangle r$), si todos los caracteres de r están presentes en s en el mismo orden, aunque no aparezcan de forma consecutiva. Formalmente:

$s \rangle\rangle \varepsilon = verdad$
$\varepsilon \rangle\rangle r = falso,$ si $r \neq \varepsilon$
$\alpha s \rangle\rangle \beta r = s \rangle\rangle r,$ si $\alpha = \beta$
$\alpha s \rangle\rangle \beta r = \alpha s \rangle\rangle r,$ si $\alpha \neq \beta$

donde ε denota la cadena vacía (con cero símbolos), y αs denota la cadena cuyo primer símbolo es α y cuyo resto de símbolos vienen dados por la cadena s.

Como ejemplo, consideremos el alfabeto $\sum = \{ a, b, c \}$. Podemos decir que *aabac* $\rangle\rangle$ *abc*, ya que, por ejemplo, los caracteres 2, 3 y 5 de la primera se corresponden con todos los de la segunda en el mismo orden. Sin embargo, para el mismo alfabeto, **no es cierto** que *acaba* $\rangle\rangle$ *abc*.

Sean $s,r,t \in \sum^*$ tres cadenas de símbolos. Diremos que la cadena s es supersecuencia común de las cadenas r y t, si $s \rangle\rangle r$ y $s \rangle\rangle t$. Obviamente, dadas dos cadenas, es fácil encontrar una supersecuencia común de ambas (serviría la cadena que se obtiene al concatenarlas). Sin embargo, estaremos interesados en encontrar una supersecuencia común los más corta posible. Éste es precisamente **el problema de la supersecuencia común más corta (SCMC).**

Resolución por programación dinámica

El problema SCMC, para dos cadenas r y t, puede ser resuelto usando programación dinámica. Para ello, observa que:

1) Si el último símbolo de las dos cadenas coincide, la supersecuencia común más corta de ambas debe acabar necesariamente con dicho símbolo.
2) En otro caso, la supersecuencia común más corta podrá finalizar bien con el último símbolo de la primera cadena, bien con el último símbolo de la segunda. De ambas opciones, tomaremos la que de lugar a una supersecuencia más corta.

Objetivos de la práctica

El alumno deberá:

a) Plantear la relación de recurrencias correspondiente al problema SCMC.

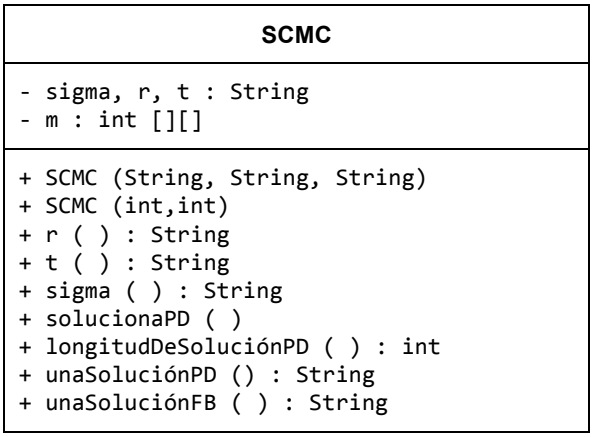
b) Implementar una solución por programación dinámica a este problema con complejidad $O(|r| \times |t|)$. Para ello, se rellenará una matriz $m[0..|r|][0..|t|]$ usando la relación de recurrencias del punto **a)**, de modo que la entrada $m[i][j]$ (para $0 \leq i \leq |r|$ y $0 \leq j \leq |t|$) indique la longitud de la SCMC de las cadenas $r_1\ r_2\ \ldots\ r_i$ y $t_1\ t_2\ \ldots\ t_j$.

c) Implementar el algoritmo que reconstruya, usando la tabla del punto **b)**, una supersecuencia común más corta (nótese que, aunque pueden existir varias soluciones óptimas para una instancia, estaremos interesados en obtener solo una).

d) Evaluar experimentalmente la corrección de su implementación, tal como se explica posteriormente.

Modelado del problema

Para resolver este problema, definiremos una clase `SCMC`, de modo que un objeto instancia de la clase representa una instancia del problema (dada por un alfabeto `sigma` y dos cadenas `r` y `t`):



La clase `SCMC` define dos constructores (ya implementados en el esqueleto proporcionado):

- **public** `SCMC (String sigma, String r, String t)`, que crea una instancia a partir de un alfabeto y dos cadenas, y
- **public** `SCMC (int longSigma, int maxLong)`, que crea una instancia aleatoria del problema, dados el tamaño del alfabeto y la longitud máxima de las cadenas.

Los métodos `r()`, `t()` y `sigma()` son selectores que permiten acceder a las distintas componentes de una instancia. Se suministra, ya implementado, el método `unaSoluciónFB()`, que devuelve una solución al problema usando el método de fuerza bruta descrito posteriormente.

La variable de instancia `m` es la matriz que se usará para resolver el problema por programación dinámica. El método `solucionaPD()` debe solucionar la instancia aplicando programación dinámica. Para ello, rellenará la matriz `m` (apartado **b)** de la práctica). Una vez resuelta una instancia, deberá poderse obtener la longitud de la supersecuencia común más corta al invocar el método `longitudDeSoluciónPD()`. El método `unaSoluciónPD()` debe devolver una supersecuencia común más corta para la instancia (apartado **c)** de la práctica), usando para ello la información almacenada en la tabla `m`. Por ejemplo, el programa:

```
public static void main(String[] args) {
 SCMC scmc = new SCMC("abc", "aab", "acbaa");
 scmc.solucionaPD();
 System.out.println("La longitud de la SCMC es "
                    +scmc.longitudDeSoluciónPD());
 System.out.println("Una SCMC es "+scmc.unaSoluciónPD());
}
```

produciría la siguiente salida:

```
La longitud de la SCMC es 6
Una SCMC es aacbaa
```

Con objeto de que el alumno compruebe experimentalmente el funcionamiento de su implementación (apartado **d)** de la práctica), se suministra la clase `TestsCorrección`, que comprueba que la implementación es correcta para un conjunto de instancias almacenadas en el fichero **"`tests.txt`"**. Para una implementación correcta, la salida por pantalla tras ejecutar este método debe ser:

```
Test de unaSolución correcto
```

La clase Utils

Se suministra también una clase `Utils`, con distintos métodos auxiliares (documentados en el código) que pueden ser útiles para resolver la práctica.

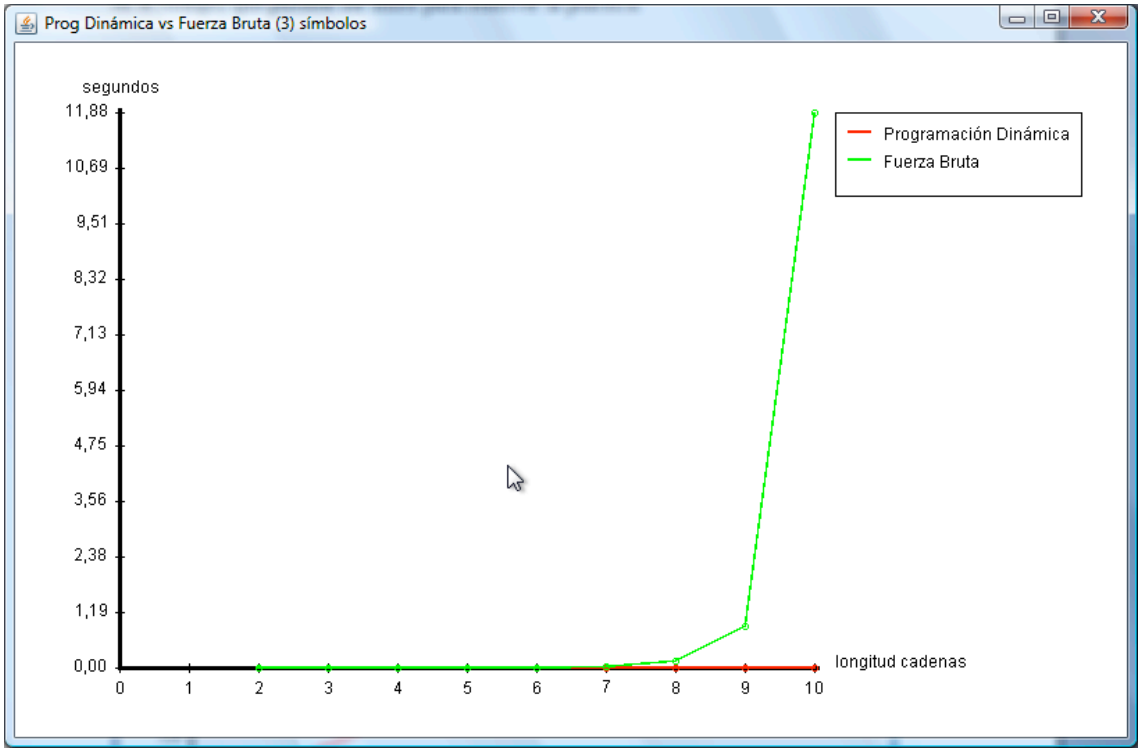
Comparación experimental con resolución por fuerza bruta

Resolución por fuerza bruta

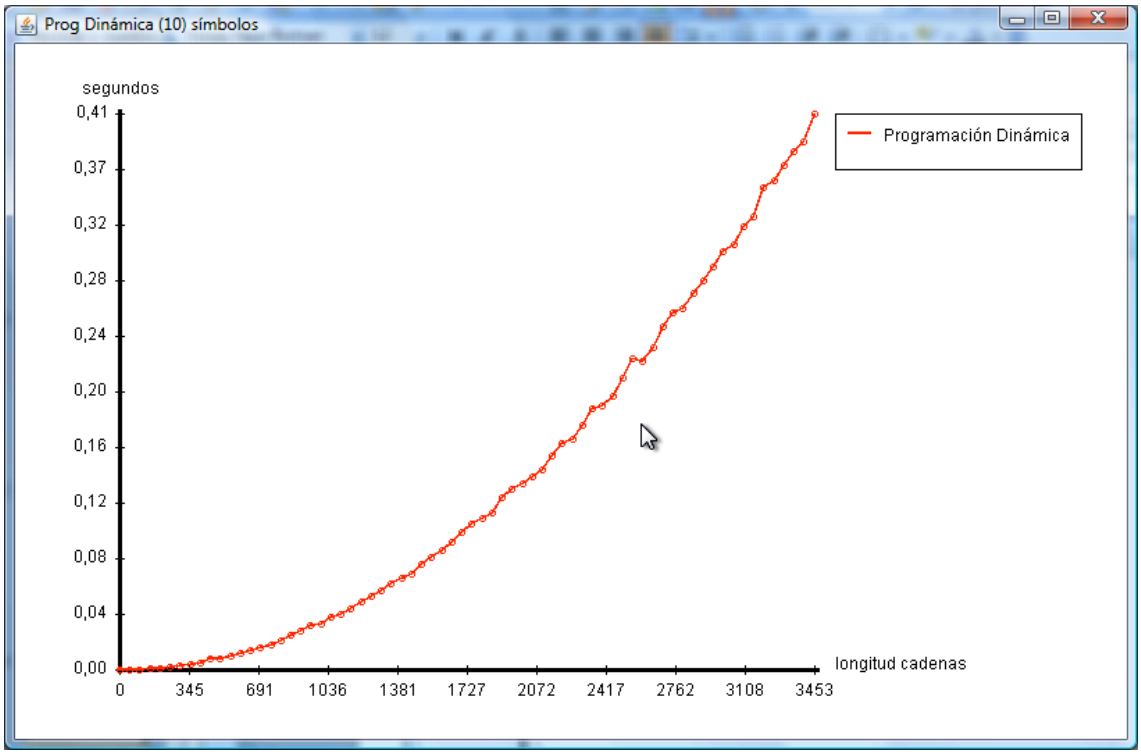
Obsérvese que si *s* es una supersecuencia común de las cadenas *r* y *t*, la longitud de *s* debe ser como mínimo, la de la más larga de las originales ($|s| \geq max\ (|r|,|t|)$). Un modo ineficiente de resolver el problema SCMC es generar todas las secuencias posibles de símbolos con longitud igual (o superior) a ésta, hasta encontrar una supersecuencia común. Este método de resolución, que llamaremos por fuerza bruta, está implementado en el método `unaSoluciónFB()`, suministrado con el esqueleto de la práctica. Dado que existen $|\sum|^n$ cadenas de longitud *n*, el número de cadenas a examinar crece exponencialmente con el tamaño de las cadenas consideradas, lo cuál hace esta solución muy ineficiente.

La clases Temporizador y Gráfica

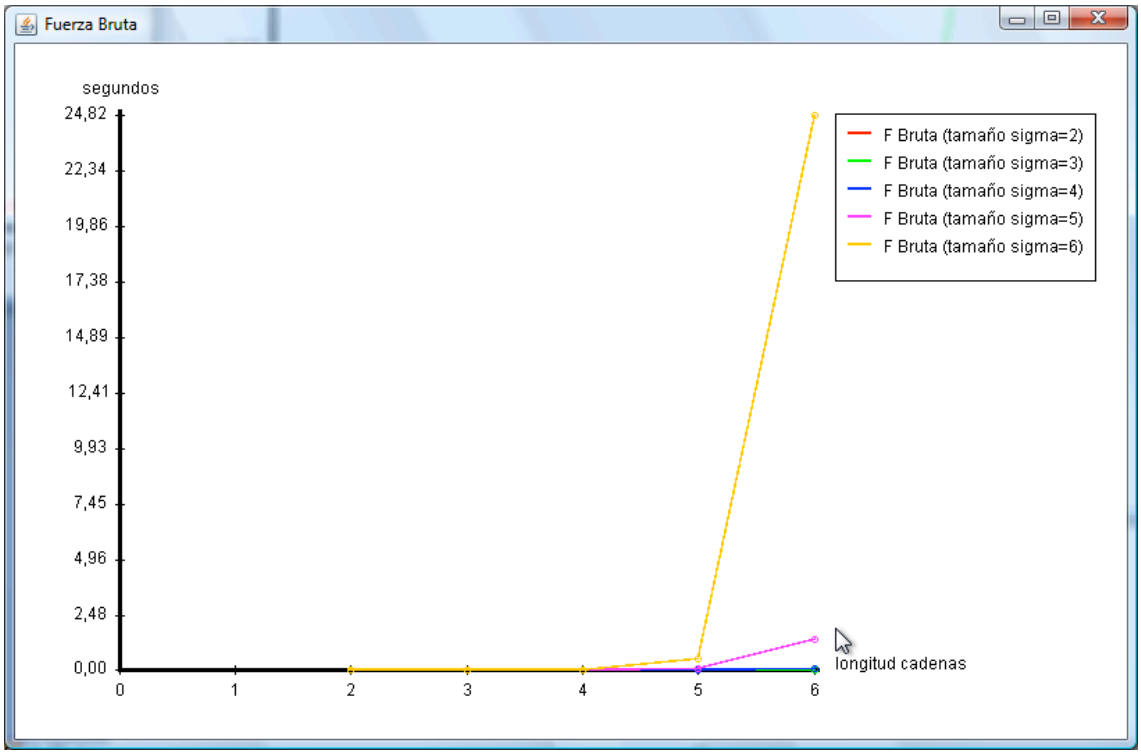
Con la clases `Temporizador` y `Gráfica` suministradas, podrás realizar gráficas para estudiar experimentalmente el comportamiento de tu implementación frente al método de fuerza bruta, como las siguientes, que se obtuvieron en un Pentium Core2 6300. Para ello, tendrás que ejecutar el método `main` de la clase `TestsTiempos` suministrada.



Gráfica que muestra el tiempo necesario para resolver instancias del problema con cadenas de longitud de hasta 10 símbolos con un alfabeto de 3 símbolos. Obsérvese que el tiempo necesario para resolver instancias de tamaño 10 mediante fuerza bruta es de casi 12 segundos, mientras que con programación dinámica se resuelve en un tiempo despreciable.



Gráfica que muestra que se pueden resolver instancias con cadenas de hasta 3500 símbolos (con un alfabeto de 10 símbolos) usando programación dinámica en menos de medio segundo. Obsérvese que el tiempo de ejecución crece de modo cuadrático aproximadamente, tal como se espera del análisis teórico del algoritmo.



Gráfica que muestra que el tiempo de resolución por fuerza bruta crece exponencialmente tanto con la longitud de las cadenas como con el tamaño del alfabeto, y que se necesita más de 24 segundos para resolver una instancia muy simple (cadenas de 6 símbolos con un alfabeto de tamaño 6).

Entrega

Una vez finalizada la practica, debe entregarse el programa realizado para su corrección de forma automática mediante el programa *Siette*. Para ello, seleccionar la actividad correspondiente en el Campus Virtual y enviar, una vez modificado, el fichero:

```
SCMC.java
```