ISIS 1105 Diseño y Análisis de Algoritmos
Semestre 2022-20. Proyecto – PARTE 1
Entrega: lunes, octubre 14 de 2022 6:00PM (por Bloque Neon)

0 OBJETIVOS

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

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

1 CONDICIONES GENERALES

El proyecto se divide en tres partes independientes entre sí. Este documento describe la PARTE I. Cada parte contiene un problema a resolver mediante soluciones implementadas en `Java` o `Python`.

Para cada problema se pide:
- Descripción de la solución.
- Análisis temporal y espacial.
- Una implementación en Java o Python

2 DESCRIPCIÓN DEL PROBLEMA

A Encontrando las ocurrencias de una subsecuencia

Definición: Una subsecuencia Y de una secuencia X dada es una secuencia que se puede derivar de X eliminando algunos o ningún elemento y sin cambiar el orden de los elementos restantes. Por ejemplo "*dlo*" es una subsecuencia de "*dalgo*".

Suponga que tiene dos secuencias de caracteres alfabéticos X y Y, donde $length(Y) < lenght(X)$. Como parte de un juego se le permite hacer movimientos de reemplazo en X. En un movimiento puede seleccionar un determinado carácter $x_i \in X$ (i determina la posición), y reemplazarlo por cualquier otro carácter del alfabeto. Así por ejemplo en la cadena "*dalgo*" en un movimiento puedo reemplazar el carácter $x_0 =$ "d" por el carácter "a" y la cadena resultante sería "*aalgo*".

Problema

Un mago de los strings lo reta a realizar a lo mucho m movimientos de reemplazo de tal forma que se maximice el número de ocurrencias de Y en X como subsecuencia.

Ejemplo:

Dado $X = "ccqq"$, $Y = "qc"$, y $m = 2$ el conjunto de movimientos de reemplazo:

- $x_0 =$ c reemplazarlo por q.
- $x_3 = q$ reemplazarlo por c.

Nos llevan a la cadena $"qcqc"$ que es la cadena que maximiza el número de ocurrencias de Y como subsecuencia. En total aparece 3 veces:

- $qcqc$
- $qcqc$
- $qcqc$

Nota: No es obligatorio realizar los m movimientos disponibles.

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

Descripción de la entrada

La primera línea de entrada especifica el número de cados de prueba que contiene el archivo. El programa debe terminar su ejecución, una vez termine de resolver la cantidad de casos de prueba dados por este número.

Cada caso está representado en una línea con la especificación de X, Y y m. Solo se van a contemplar caracteres en minúscula del alfabeto castellano.

$$2 \leq length(X) \leq 1000$$

$$length(Y) = 2$$

$$0 \leq m \leq length(X)$$

Descripción de la salida

Para cada caso de prueba, imprimir el máximo número posible de ocurrencias de Y en X como subsecuencia.

Ejemplo de entrada / salida

Entrada	Salida
1 ccqq qc 2	3

Tres casos de prueba:

Entrada	Salida
3 ccqq qc 2 axttxaf xt 3 ubucubu uu 2	3 10 15

Nota: Se van a diseñar casos de prueba para valores de $length(X)$, y m muchos más grandes y dentro de los valores establecidos en el enunciado. Los casos mostrados en este documento son demostrativos de la estructura de entrada/salida esperada.

4 COMPRENSIÓN DE PROBLEMAS ALGORITMICOS

A continuación, se presentan un conjunto de escenarios hipotéticos que cambian el problema original. Para cada escenario debe contestar[1]: (i) que nuevos retos presupone este nuevo escenario -si aplica-?, y (ii) que cambios -si aplica- le tendría que realizar a su solución para que se adapte a este nuevo escenario?

ESCENARIO 1: Suponga ahora que un movimiento no aplica solo a una posición, sino a todo X. Es decir, un movimiento cambiaría toda ocurrencia de un carácter x en X. Por ejemplo, para $X = "abacachi"$ un posible movimiento seria cambiar el carácter "a" por el carácter "u", dando como resultado $X = "ubucuchi"$.

ESCENARIO 2: Ya no hay límite de movimientos.

Nota: Los escenarios son independientes entre sí.

[1] NO tiene que implementar la solución a estos escenarios, el propósito es meramente analítico.

5 ENTREGABLES

El proyecto puede desarrollarse por grupos de hasta tres estudiantes de la misma sección. La entrega se hace por bloque neon (una sola entrega por grupo de trabajo).

El grupo debe entregar, por bloque neon, un archivo de nombre `proyectoDalgoP1.zip`. Este archivo es una carpeta de nombre `proyectoDalgoP1`, comprimida en formato `.zip`, dentro de la cual hay archivos fuente de soluciones propuestas y archivos que documentan cada una de las soluciones.

5.1 Archivos fuente de soluciones propuestas

Todos los programas implementados en *Java* o en Python

Para el problema:

- Entregar un archivo de código fuente en *Java* (`.java`) o python (`.py`) 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 `ProblemaP1.java` o `ProblemaP1.py` el archivo de la solución que se presente.

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

5.2 Archivos que documentan la solución propuesta

La solución al problema debe acompañarse de un archivo de máximo 3 páginas que la documente, con extensión `.pdf`. El nombre del archivo debe ser el mismo del código correspondiente (`ProblemaP1.pdf`).

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 estas. Debe realizarse un análisis para cada solución entregada.

3 Respuestas a los escenarios de comprensión de problemas algorítmicos.
Respuesta a las preguntas establecidas en cada escenario. NO tiene que implementar la solución a estos escenarios, el propósito es meramente analítico.

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.