ISIS 1105 Diseño y Análisis de Algoritmos
Semestre 2022-10. Proyecto – PARTE 3
Entrega: viernes, junio 3 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 III. 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 Encriptación ingenua de contraseñas

Un personaje muy desconfiado llamado "Cripton" propone una estrategia para encriptar las contraseñas de sus diversas aplicaciones. Dada una cadena s que representa una contraseña y una cadena inicialmente vacía t, realiza las siguientes operaciones:

a. Concatena a la derecha de la cadena t la cadena s.
b. Selecciona un carácter arbitrario presente en s y se remueve todas sus ocurrencias. El carácter seleccionado debe estar presente en s al momento de realizar esta operación.

Las operaciones a y b son ejecutadas en orden estricto hasta que la cadena s queda vacia. La cadena t resultando de este proceso es igual a algún valor que depende del orden en que fueron removidos los caracteres, y es usado como encriptación.

A modo de ejemplo considere $s = \text{armar}$ y los resultados después de cada ronda de operaciones a/b:

- $t = \text{armar}$, el carácter r es seleccionado, se remueve y $s = ama$
- $t = \text{armarama}$, el carácter a es seleccionado, se remueve y $s = m$
- $t = \text{armaramam}$, el carácter m es seleccionada, se remueve y s es la cadena vacía.

Problema

Un estudiante de DALGO al conocer el método de encriptación, le indica a Cripton que su método es extremadamente inseguro. De acuerdo con el estudiante conociendo la cadena t es posible recuperar la contraseña s y el orden exacto de remoción de los caracteres. Cripton indignado desafía el estudiante a desencriptar sus contraseñas y le entrega diversos valores de t de prueba.

El problema consiste en (i) recuperar el estado inicial de la cadena s usando el estado final de la cadena t, y (ii) recuperar el orden en que fueron removidas las letras de la cadena s.

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 este compuesto de una línea con una cadena t. La cadena t está compuesta de letras del alfabeto (minúsculas o mayúsculas). El número total de letras que componen a t varían entre 1 y 10^4 $(1 \leq t < 10^4)$.

Descripción de la salida

Para cada caso de prueba, imprimir en una línea:

- NO EXISTE, si no existe una cadena s de respuesta.
- Dos cadenas separadas por un espacio. La primera contiene un posible valor inicial de s. La segunda debe contener la secuencia de eliminación de las letras de s.

Ejemplo de entrada / salida

Para el ejemplo especificado en A.

Entrada	Salida
1 rsrtrsrrrtrrt	rsrtrsr srt

Múltiples casos de prueba:

Entrada	Salida
6 rsrtrsrrrtrrt qweqeewew ama ppuxbdvdmmzbsbmrzjkdxkzxkbhpzpmmupbmz mfqmtptmqpitmqmtptmqpitmqmtptmqptqtptqptqttqtttt v	rsrtrsr srt NO EXISTE am ma NO EXISTE mfqmtptmqpit fimpqt v v

Nota: Se van a diseñar casos de prueba para valores de t 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: Se le pide generar cadenas t para las cuales es imposible obtener s y la secuencia de remoción de caracteres.

ESCENARIO 2: Se añade un nuevo paso a las operaciones de encriptación así:

a. Concatena a la derecha de la cadena t la cadena s.
b. Selecciona un carácter arbitrario presente en s y se remueve todas sus ocurrencias. El carácter seleccionado debe estar presente en s al momento de realizar esta operación.
c. Selecciona aleatoriamente un carácter no presente en s (en su versión inicial) y concatenarlo a la derecha de la cadena t.

Nota: Los escenarios son independientes entre sí.

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 `proyectoDalgoP3.zip`. Este archivo es una carpeta de nombre `proyectoDalgoP3`, comprimida en formato `.zip`, dentro de la

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

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 `ProblemaP3.java` o `ProblemaP3.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 (`ProblemaP3.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.