Estructuras de Datos y Algoritmos

Práctica II - Curso 2020/21

Nombres más comunes – Avanzado (revisada)

1. Introducción

En esta segunda versión de la práctica se debe resolver exactamente el mismo problema planteado en la primera práctica pero en este caso usando estructuras de datos avanzadas.

La entrada de datos será la misma (nombre del fichero, número de nombres a mostrar, nombre del usuario a seguir, intervalo de fechas), así como la presentación de resultados (listado de nombres más comunes y ranking del nombre a seguir). Los cambios respecto a la primera práctica van a ser los siguientes:

1. Se van a utilizar estructuras de datos avanzadas para las fases críticas (inserción y extracción). En concreto, no se va a usar la clase APL ni el método de ordenación de la práctica anterior. Las estructuras de datos se pueden escoger de entre las siguientes, dependiendo del lenguaje de programación elegido:
   - **Java**: Todas las del paquete **java.util**
   - **Python**: Todos los tipos de datos estándar (listas, tuplas, conjuntos, diccionarios) más la librería estándar **heapq** y la librería externa **sortedcontainers**
2. No se va a medir el número de operaciones de las aplicaciones. Tan solo se medirá el tiempo total en segundos empleado por la aplicación en realizar los cálculos (es decir, el tiempo entre justo después de leer el fichero y justo antes de empezar a mostrar los resultados en pantalla).

A diferencia de lo indicado en el enunciado anterior de esta práctica, solo va a ser necesario el crear **una única** aplicación.

2. Detalles del algoritmo (atención: spoiler)

Es conveniente seguir utilizando (o crear, si no se hizo) la clase `Nombre` definida en la práctica anterior, que encapsula el nombre de pila (string) y el número de ocurrencias (entero).

Para la **fase de inserción** (que se realiza dentro de la fase de filtrado) sería necesaria una estructura de datos (**A**) en la que pudiera comprobarse, de la forma más eficiente posible, si un nombre de pila ya ha aparecido o no durante el procesado del array de personas. Y, en caso de ya hubiera aparecido, el que se pudiera acceder al objeto de clase `Nombre` correspondiente para incrementar el contador.

Para la **fase de extracción** el algoritmo **no se debe basar** en ordenar los objetos `Nombre`: Como el número de nombres a mostrar, *k*, típicamente va a ser muy pequeño, podemos usar la siguiente idea:

- Se recorren los objetos `Nombre` almacenados en **A** actualizando una estructura de datos (**B**) que contenga los *k* nombres más comunes encontrados hasta el momento. Para cada

nombre se comprobaría si su número de apariciones es mayor que **el peor** de los nombres almacenados en **B**, y si se da ese caso se reemplazaría ese peor por el que estamos tratando. Evidentemente hay que escoger la estructura **B** para que pueda realizar de forma eficiente esas operaciones.

- Al terminar simplemente se deben extraer en orden los *k* nombres almacenados en **B**, que serán los más comunes. **Nota**: Tampoco deben usarse métodos de ordenación en este último apartado.

Por último la **fase de seguimiento** ahora es diferente de la de la práctica anterior, ya que no disponemos de un array de objetos `Nombre` ordenados por número de apariciones, se debe idear una forma de calcular el ranking del nombre de seguimiento con los datos disponibles en las estructuras de datos anteriores. Ejemplo de salida del algoritmo:

```
Nombre del fichero: personas_va.txt
_|____|____|____|____|
**********************
Creación array: 0,01747 seg. Lectura fichero: 0,13702 seg.
n = 448560 personas en total.

*** OPCIONES GENERALES ***
Número de nombres a mostrar: 5
Nombre del usuario: CESAR
Intervalo fechas de nacimiento: 1/1/1974 31/12/1982

    1. DAVID 1.523
    2. JAVIER 943
    3. ALBERTO 827
    4. OSCAR 822
   56. CESAR 303

*** MEDIDAS ***
Tiempo de proceso: 0,023235 seg.
m = 59.338 personas cumplen las condiciones.
p = 1.039 nombres distintos.
d = 3.287 dias en el intervalo.
```

3. Presentación y Evaluación de la práctica

La práctica puede ser codificada en Java o Python, aunque se recomienda el uso de Java ya que esta práctica puede ser muy exigente respecto a los recursos (tiempo y espacio de almacenamiento).

Para una correcta evaluación de la práctica el alumno deberá:

1. Presentar electrónicamente (por el Aula Virtual de la Escuela o por correo electrónico), antes de la fecha límite (domingo **13 de diciembre de 2020**), un fichero comprimido que contenga el código fuente de la aplicación descrita en el enunciado. En el código fuente debe aparecer (como comentario) en las primeras líneas el nombre de quien ha realizado la práctica. Es posible que también sea necesario rellenar un formulario de entrega
2. Presentarse a la sesión de defensa que le corresponda según su grupo de laboratorio en los dias del **14 al 16 de diciembre de 2020**. En esta sesión se probará la aplicación y se pedirá una modificación sencilla.

Es en la defensa de la práctica donde se produce la evaluación, la presentación electrónica es simplemente un requisito previo para garantizar la equidad entre subgrupos y la comprobación preliminar de autoría.