Ing. de Sistemas e Informática
Estructuras de Datos y Algoritmos
Jorge Mario Londoño P.

Taller 5 - Recuperación

Ejercicio 1: Intersección de conjuntos ordenados

El método tradicional para la intersección de dos conjuntos requiere un tiempo cuadrático, lo cual es muy ineficiente si se tienen conjuntos grandes. Sin embargo, cuando los conjuntos están ordenados, la intersección se puede encontrar en un tiempo mucho menor. Para tal efecto:

1. Desarrollar un algoritmo que encuentre la intersección de dos arreglos de datos ordenados:

```
List<Comparable> interseccionOrdenados(Comparable[] a, Comparable[] b)
```

2. Estimar analíticamente el tiempo requerido por el método interseccionOrdenados.

3. Se propone que la intersección de conjuntos no ordenados se puede hacer con el método anterior si se ordenan previamente los conjuntos, esperando que el tiempo total no sea de orden cuadrático. Desarrollar el método que realiza este proceso a partir de dos conjuntos no ordenados:

```
List<Comparable> interseccionNoOrdenados(Comparable[] a, Comparable[] b)
```

4. Estimar analíticamente el tiempo requerido por el método interseccionNoOrdenados.

Ejercicio 2: Clasificación TopM

Se trabajo en clase sobre el problema de encontrar los TopM de un conjunto de N elementos.

Se quiere implementar una solución genérica de este problema para objetos que implementen la interface `ParejaLlaveValor`:

```
interface ParejaLlaveValor<Key, Value extends Comparable<Value>> implements Comparable<Value> {
     Key getKey();
     Value getValue();
}
```

5. Dar una implementación del algoritmo topM que encuentra los top-M elementos de una colección de objetos ParejaLlaveValor y los retorne como una cola de prioridad MinPQ:

```
MinPQ<ParejaLlaveValor> topM(List<ParejaLlaveValor> datos, int M)
```

6. En clase se propuso un algoritmo para contar la palabras en un texto. Adaptar este algoritmo para retornar el conjunto de palabras con el número de ocurrencias como objetos `ParejaLlaveValor`:

```
List<ParejaLlaveValor> contadorDePalabras(String rutaArchivo)
```

7. Implementar un método topMPalabras que obtenga las M palabras de mayor ocurrencia en un archivo de texto plano y las imprima en pantalla en orden decreciente por su número de ocurrencias:

```
void topMPalabras(String rutaArchivo, int M)
```

8. Estimar analíticamente el tiempo requerido por el algoritmo `topMPalabras`. Ignorar operaciones de I/O en el análisis.

9. Obtener las top 50 palabras de "El quijote de la Mancha" y anexar el pantallazo mostrando cuales con las 50 palabras y sus respectivas frecuencias en orden decreciente.

Entregables:

Remitir el código fuente de los métodos propuestos como funciones de biblioteca dentro de una clase Taller5.

Anexar documento (.doc, .pdf) con la solución de los ejercicios de análisis.

Nombrar el archivo comprimido ApellidoNombre-Taller5.zip (o .rar o .7z o .tgz)

Desarrollar los ejercicios en forma individual.