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

Taller 3

Técnicas de ordenación aplicadas a listas

Para el estudio de las técnicas de ordenación se suelen utilizar arreglos por facilidad para entender los distintos algoritmos. Sin embargo, en muchas situaciones prácticas los datos a organizar se pueden encontrar en estructuras tipo lista enlazada. En este taller se explorarán dos posibles soluciones, sin necesidad de convertir la lista en un arreglo.

Selección del ejercicio a desarrollar

El ejercicio se desarrolla en equipos de máximo 3 personas. Para determinar que ejercicio le corresponde al equipo, obtener el hash SHA-256 de la concatenación de los ID de los integrantes, por ejemplo:

```
String concat = “0001233”+“0009234”+“0005567”;
```

Al string resultante, calcularle su hash SHA-256 utilizando páginas como esta o esta. Al valor hexadecimal resultante, determinar si es par o impar para realizar uno de los dos ejercicios que se proponen a continuación:

Hash de equipo par

Para este caso se implementará el algoritmo Mergesort sobre listas doblemente enlazadas. Para tal efecto desarrollar cada uno de los siguientes puntos:

1. Implementar los métodos `split()` y `merge()` de listas doblemente enlazadas. El método `split()` divide la lista en dos mitades y el método `merge()` toma dos listas ordenadas y las fusiona formando una sola lista ordenada.

   ```
   ListaDoble[] split() // devuelve dos listas doblemente enlazadas

   ListaDoble merge(ListaDoble l1, ListaDoble l2)
   ```

2. Implementar el algoritmo Mergesort de una lista doblemente enlazada utilizando los dos métodos anteriores.

   ```
   void mergesort(ListaDoble l) // ordena la lista l
   ```

3. Estimar el tiempo requerido por su implementación del método `split`, el

método `merge` y finalmente con estos resultados estimar el tiempo del `mergesort` de listas.

4. Evaluar experimentalmente el tiempo del mergesort implementado utilizando listas de nombres generadas aleatoriamente.

Hash de equipo impar

En este caso se implementará el algoritmo Quicksort sobre listas doblemente enlazadas. Para tal fin desarrollar cada uno de los siguientes puntos:

1. Desarrollar el método `partition` que tome como pivote un valor de una lista doblemente enlazada y retorne dos listas, una con los valores menores al pivote y otra con los valores mayores al pivote.
2. Implementar el algoritmo `Quicksort` de listas doblemente enlazadas, partiendo del método partition del punto anterior: Se particiona la lista de entrada, se ordenan recursivamente las listas izquierda y derecha, y luego se concatenan las listas izquierda, el pivote y la lista derecha, para obtener la lista ordenada.
3. Estimar el tiempo requerido por su versión del método `partition`. Con base en este resultado estimar el tiempo total requerido por el algoritmo `Quicksort`.
4. Evaluar experimentalmente el tiempo de quicksort implementado utilizando listas de nombres generadas aleatoriamente.

Entregables:

Implementación del algoritmo de ordenación de listas doblemente enlazadas según el caso que corresponda. No incluir la biblioteca `algs4.jar`. Análisis del tiempo requerido por el algoritmo (documento word o pdf). Hoja de Excel con los resultados tabulados y con la gráfica obtenida para el numeral 4.