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

Taller 3

Técnicas de ordenación

Las distintas técnicas de ordenación son de aplicación frecuente en el desarrollo de aplicaciones. Dependiendo de la situación, cierta técnica puede ser más apropiada. Adicionalmente, en muchas situaciones la llave puede no estar pre-establecidad o se puede requerir cambiar de llave en un mismo conjunto de datos. En este ejercicios se explorarán todas estas consideraciones.

Ejercicios a desarrollar

Para los efectos de este ejercicio se desea trabajar sobre los datos de la colección de los libros más populares que se encuentra en este enlace. Cada reglón corresponde a un libro y los distintos atributos se encuentran separados por commas (Comma Separated Values – CSV).

1. Definir un ADT `Libro` que contenga los atributos de un libro. Considerar todos los atributos disponibles en la colección. Utilizar tipos de datos acorde con el respectivo atributo. Incluir una versión sobre-escrita del método `toString`.
2. Implementar la interface `Comparable` en el ADT `Libro`. Se deben comparar libros con base en su *average_rating*.
3. Definir un método estático `leerCSV` que lea la los datos de la colección y los retorne como un arreglo de objetos `Libro`.
4. Se desea obtener el listado en orden descendiente por `average_rating`. Definir el método `listarPorRating` para realizar esta tarea.
5. Si se desea utilizar campos distintos para efectos de la ordenación, la solución más apropiada es implementar la interface `Comparator` como una clase aparte (puede ser clase interna o clase anónima). Implementar tres Comparators para el ADT `Libro`: Uno por autor, otro por número de calificaciones (`ratings_count`) y otro por fecha de publicación (`publication_date`).
6. Implementar un método `listarPorComparador` que acepte como parámetros el arreglo de libros y el comparador a utilizar. Se debe imprimir el listado ordenado según el comparador indicado.

7. Se desea comparar la eficiencia de distintos métodos de ordenación. Para tal efecto se requiere un conjunto de métodos `medirTiempoALG` que acepte un comparador, ordene los datos y retorne el tiempo requerido (No realizar operaciondes de I/O durante la medición de los tiempos). Dar al menos tres implementaciones utilizando los algoritmos de Selection, Insertion, Shell, Merge o Quick.
8. Realizar mediciones de tiempo para los tres métodos seleccionados utilizando los tres Comparators previamente definidos. Realizar K≈20 mediciones por caso para obtener promedios más confiables. Tabular los resultados obtenidos en cada caso.
9. Realizar un análisis descriptivo concluyendo acerca de las diferencias que se observan entre métodos de ordenación y entre los comparadores utilizados. Como se explican las diferencias? Concuerdan con los esperado según los ordenes de crecimiento de cada algoritmo?

Notas:

- Cada numeral vale 1 punto.
- Reutilizar las implementaciones de los métodos de ordenación definidas en las bibliotecas del texto guía.
- Se puede incluir en el comparativo la implementación propia de las bibliotecas del Java, Arrays.sort. (Opcional +1 punto)

Entregables:

Implementación del ejercicio. Incluir las clases `Libro` y una clase `Taller3` que contenga el `main` que invoque los distintos métodos. No incluir la biblioteca `algs4.jar`.

Documento con los datos tabulados (numeral 8) y con el análisis (numeral 9).