Universidad de los Andes

Ingeniería de Sistemas y Computación
ISIS1206 – Estructura de Datos
Taller 4

Taller 4: Ordenamiento

Objetivos

- Entender los diferentes tipos de ordenamiento y como estos afectan la complejidad y tiempos de ejecución
- Aprender a implementar algoritmos de ordenamiento en un lenguaje de programación

Descripción General

Rendimiento y comparación práctica de algunos algoritmos.

Con el fin de poder comparar el tiempo de ejecución de diversos algoritmos de ordenamiento sujeto al tamaño de la entrada, la Asociación Internacional de Algoritmos Altamente Ineficientes (AIAAI) ha propuesto establecer una carrera internacional de algoritmos. La idea en general consiste **en la inscripción de diversos equipos;** cada equipo debe contar con la implementación de un algoritmo de ordenamiento. Este algoritmo debe someterse a prueba en diversas rondas de ordenamiento, durante las cuales, se **provee una lista de elementos** comparables de una longitud específica.

El equipo ganador se establece por medio de la posición promedio durante cada ronda. Esta posición se asigna de acuerdo al tiempo de ejecución del algoritmo en la ronda. Los algoritmos que tengan el menor tiempo de ejecución obtendrán posiciones superiores.

Ejecución del programa y configuración del entorno.

Compile y ejecute el proyecto.

```
----------------------------------------
-                                      -
-    Carrera: Orden y eficiencia       -
-                                      -
----------------------------------------
¡Bienvenido a la segunda carrera más importante de ordenamiento en el hemisferio occidental!

Menú principal:
-----------------
1. Iniciar pruebas y competencias
2. Configuración y ajustes
3. Salir

Seleccione una opcion:
```

Figura 1 Menú principal de la aplicación.

Tras ejecutar la aplicación, la terminal deberá tener como contenido un menú de selección similar al presentado en la Figura 1. Ahora proceda a interactuar con el menú de forma completa, con el fin de detallar y conocer las opciones ofrecidas en el Torneo. Es posible ejecutar una instancia del torneo con los equipos del sistema bajo la opción número 1 en el menú principal, seguida de la opción 2.

Definición y descripción de un equipo.

Por definición, un equipo se encuentra directamente relacionado con la implementación específica de un algoritmo de Ordenamiento. No obstante, todo equipo que participa en el torneo debe contar con parámetros comunes, tal como un nombre, un tiempo de ejecución de la rutina de ordenamiento y un porcentaje de efectividad dispuesto después de cada ejecución de la rutina de ordenamiento. Para este fin, se define una clase abstracta *AlgorithmTeam*, la cual además de declarar los atributos previamente enumerados, define el método abstracto *public abstract Comparable[] sort(Comparable[] list, TipoOrdenamiento orden);* método que debe ser implementado por la clase (Equipo) que extienda directamente la clase *AlgorithmTeam*. Es importante notar que el arreglo a ordenar contiene instancias de la clase Comparable. Es decir, cada elemento debe disponer de un método *compareTo* el cual permitirá comparar dos elementos.

Adicionalmente, el parámetro orden puede adquirir dos valores enumerados (como es posible apreciar en *AlgorithmTournament.java*), los cuales establecen el criterio de orden de los elementos. Este valor puede ser *TipoOrdenamiento.ASCENDENTE* o *TipoOrdenamiento.DESCENDENTE*, si se desea ordenar de forma Ascendente o Descendente, respectivamente.

En la carpeta *src/taller.mundo.temas* se encuentran alojados todos los equipos (algoritmos) que usted deberá desarrollar, como también algunos equipos que ya están listos para competir en el torneo. Uno de estos equipos (algoritmos) de complejidad $\mathcal{O}(n^2)$ es el **equipo TimSort**, el cual representa la implementación de *referencia Timsort* de la librearía estándar de Java.

Metodología y funcionamiento del torneo.

El torneo comprende una base de datos que contiene 4.000.000 números enteros que se encuentran en desorden, compilados en un conjunto de 80 archivos que contienen 50.000 elementos cada uno, con el fin de establecer una medida de referencia que permita describir la eficiencia temporal del archivo, se procede a definir un conjunto de rondas, durante cada ronda, se escoge un archivo de forma aleatoria así como el número máximo j de elementos a ordenar. Posteriormente se evalúa y se captura el tiempo de ejecución del algoritmo de ordenamiento dispuesto por cada equipo. Posteriormente, se muestra los resultados en pantalla ordenados de acuerdo al tiempo total de ejecución y se repite el proceso hasta que el número de rondas sea equivalente al límite impuesto por el usuario.

Lo que usted debe hacer

Parte 1 – Preparación (en casa)

Implementación de Bubble sort, Insertion sort y Selection Sort.

En la carpeta que contiene las clases que describen los equipos inscritos, es posible observar tres archivos: *BubbleSortTeam*, *InsertionSortTeam y SelectionSortTeam*. Así pues usted debe abrir estos archivos en Eclipse e **implementar** los tres equipos (algoritmos) **Bubble Sort**, **Insertion Sort** y **Selection Sort** respectivamente. Una vez haya finalizado proceda a **ejecutar** una instancia del torneo. (Opción 1 en el menú principal, seguida de la opción 2) y verifique el correcto funcionamiento en tiempo y eficiencia de estos tres algoritmos de ordenamiento con respecto a sus contrincantes.

Algoritmos Híbridos

Diseñar un algoritmo híbrido (un algoritmo que usa dos o más rutinas de ordenamiento) y agregarlo a la lista de equipos inscritos. Finalmente, iniciar una sesión de competencia y comparar.

Realice pruebas unitarias para la implementación de sus equipos de ordenamiento, estas pruebas deben verificar:

- Que los algoritmos realicen de forma correcta el proceso de ordenamiento.
- Que los elementos de la lista ordenada por cada algoritmo sean los mismos elementos existentes en la lista no ordenada.

Parte 2 – Trabajo en clase

Implementación de Quicksort y Merge Sort

Nuevamente en la carpeta que contiene las clases que describen los equipos inscritos, es posible observar dos archivos: *QuickSortTeam* y *MergeSortTeam*. Así pues usted debe abrir estos archivos en Eclipse e **implementar** los dos equipos (algoritmos) **Quicksort** y **Merge Sort** respectivamente. Una vez haya finalizado proceda a **ejecutar** una instancia del torneo. (Opción 1 en el menú principal, seguida de la opción 2) y verifique el correcto funcionamiento en tiempo y eficiencia de estos dos algoritmos de ordenamiento con respecto a sus contrincantes.

Entregable

A partir del desarrollo compare el desempeño de ejecución de todos los algoritmos sobre conjuntos comunes de datos; es decir pruebe cada algoritmo con los mismos conjuntos de datos.

Prepare un documento (archivo .xls) que incluya una tabla en la que se tenga para cada algoritmo el análisis temporal del peor caso y mejor caso, utilizando notación tilde (~) y la explicación cuando se presenta su mejor y su peor caso (como vienen los datos en cada caso).

Entrega

1. Verifique que su proyecto cumple con los requisitos de la entrega de talleres.
2. Adjunte el documento (archivo .xls) con la tabla comparativa de complejidades de los diferentes algoritmos en la carpeta docs del taller.
3. Entregue su taller por medio de **BitBucket**. Recuerde, si su repositorio no tiene el taller o está vacío, su taller no será calificado por más de que lo haya desarrollado.