Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI-2692 - Laboratorio de Algoritmos y Estructuras de Datos II
Trimestre Enero-Marzo 2024

Mergesort y visualización del tiempo de ejecución

1. Introducción

El primer objetivo de este laboratorio es el agregar a la librería de ordenamiento `Ordenamiento.kt`, dos versiones del algoritmo Mergesort. El segundo objetivo es el de modificar el programa cliente `PruebaOrdenamiento.kt`, para agregar a las pruebas las dos versiones de Mergesort, y la capacidad de graficar el tiempo de ejecución versus el tamaño de la secuencia a ordenar.

2. Actividades a realizar

En esta sección se presentan las actividades que debe realizar para cumplir los objetivos del laboratorio.

2.1. Modificación de la librería de ordenamiento

Se quiere que implemente dos versiones de Mergesort, siguiendo los pseudo códigos presentados en las referencias indicadas:

ms : Versión de *Mergesort* contenida en [1].

mi : *Mergesort-Iterativo* que es presentado en la página 183 de [2].

Debe agregar a la librería `Ordenamiento.kt` las funciones `mergesort` y `mergesortIterativo`. Ambas funciones reciben solo como entrada, un arreglo de elementos tipo `Number`, como el resto de los algoritmos de ordenamiento de la librería.

2.2. Modificación del programa de pruebas

Se quiere extender el programa de pruebas de los algoritmos de ordenamiento, agregando un nuevo parámetro a la línea de comandos y se modifica uno de los parámetros anteriores. La línea de comandos queda como:

```
>./probarAlOrd.sh [-t #num] [-a <algoritmos>] [-s <secuencia>] [-n #1,...,#m] [-g <figura>]
```

La semántica de los parámetros de entrada es la siguiente:

- **-s**: Clase de secuencia a realizar, en donde el parámetro *secuencia* es el identificador de la secuencia. Si se indica una secuencia que no existe, el programa muestra un mensaje de error y termina.

- **-a**: Algoritmos de ordenamiento a ejecutar. Si se escoge más de un algoritmo, los mismos deben estar separados por coma.
- **-t**: Número de intentos en el que se aplica el tipo de secuencia seleccionada.
- **-n**: Es una serie de números que indican el tamaño de las m secuencias que se van a generar para su ordenamiento. Los m números indicados deben ser diferentes, y deben estar en orden ascendente separadas por coma. En caso contrario, el programa debe abortar indicando un mensaje de error.
- **-g**: Indica que se debe mostrar un gráfico con el comportamiento del tiempo de ejecución de los algoritmos. Solo es válido si se indican dos o más tamaños de secuencia con la opción **-n**. En caso de haber indicado una sola secuencia, el programa aborta con un mensaje de error. El argumento *figura* es el nombre de la figura de formato .png que genera la librería de graficación, el cual no debe tener la extensión .png.

Todos los parámetros son obligatorios, excepto **-g**. En este caso obligatorios quiere decir, que si alguno de estos parámetros no es encuentra en la línea de comandos, entonces la ejecución se aborta con un mensaje de error. A continuación se presentan llamadas válidas del programa cliente:

```
>./probarAlOrd.sh -t 2 -s random -a is,ms -n 500,1000
```

Este comando ejecuta el programa cliente primero sobre una secuencia de 500, y luego sobre una de 1000. Las secuencias están compuestas de números enteros generados al azar, teniendo que los algoritmos Insertion Sort y Mergesort se ejecutan dos veces sobre cada una de las secuencias.

```
>./probarAlOrd.sh -s inv -n 20000,40000,50000 -t 3 -a ms,mi -g salidaInv
```

Al ejecutar este comando, el programa cliente aplica los algoritmos de ordenamiento Mergesort y Mergesort-Iterativo a tres secuencias de números enteros de tamaño 20000, 40000 y 50000, tres veces cada una. Luego genera un gráfico con los resultados de la corrida y el mismo queda guardado en el archivo `salidaInv.png`

La opción **-g** debe generar una gráfica donde el eje y es el tiempo de ejecución de los algoritmos de ordenamiento en segundos y el eje x corresponde al tamaño de las secuencias indicadas con la opción **-n**. La gráfica debe mostrar el comportamiento de todos los algoritmos de ordenamientos ejecutados. Para generar la gráfica se debe hacer uso de la librería de visualización **libPlotRuntime**, que fue publicada en clase. La figura 1 muestra un ejemplo de como puede ser graficado el comportamiento de varios algoritmos de ordenamiento.

Todo el código debe usar la guía de estilo Kotlin indicada en clase. Asimismo, el código debe estar debidamente documentado.

3. Condiciones de entrega

La entrega debe contener los archivos `Ordenamiento.kt`, `PruebaOrdenamiento.kt`, `Makefiel` y `probarAlOrd.sh`, y la carpeta y `libPlotRuntime`. La versión final del código del laboratorio y la declaración de autenticidad firmada, deben estar contenidas en un archivo comprimido, con formato *tar.xz*, llamado *LabSem2_X_Y.tar.xz*, donde X y Y son el número de carné de los estudiante. La entrega debe hacerse por la plataforma Classroom, antes de las 8:50 del día viernes 02 de febrero de 2024.

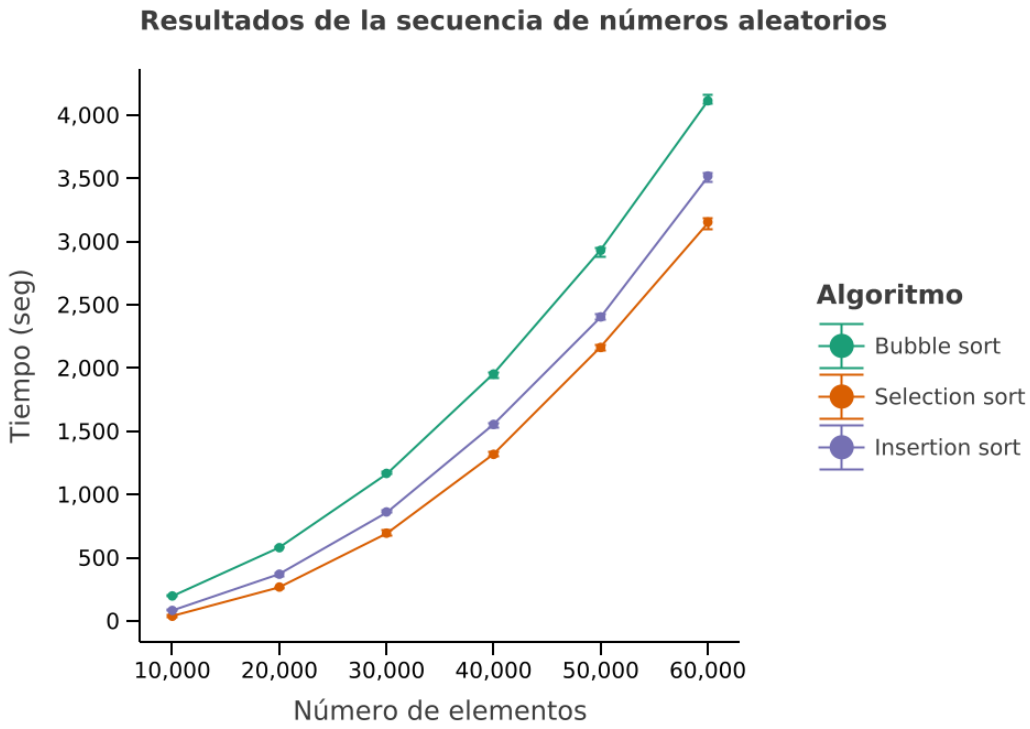

Figura 1: Ejemplo de una gráfica que muestra el comportamiento del tiempo de ejecución de tres algoritmos de ordenamiento, sobre seis secuencias de diferentes tamaño.

Referencias

[1] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to algorithms*. MIT press, 2022.

[2] Kaldewaij, A. *Programming: the derivation of algorithms*. Prentice-Hall, Inc., 1990.

Guillermo Palma / gvpalma@usb.ve / Enero 2024