Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI-2692 - Laboratorio de Algoritmos y Estructuras de Datos II
Trimestre Abril-Julio 2021

Mergesort y visualización del tiempo de ejecución

1. Introducción

El primer objetivo de este laboratorio es el agregar a la librería de ordenamiento `Sortlib.kt`, dos versiones del algoritmo Mergesort. El segundo objetivo es el de agregar al programa de pruebas de los algoritmos de ordenamiento, 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:

- *Mergesort-Insertion*: presentado en la página 229 de [1]. Cuando el número de elementos a ordenar es menor o igual a 10, entonces se ordena el arreglo con *Insertion-Sort*.
- *Mergesort-Iterativo*: introducido en la página 183 de [2].

Debe agregar a la librería `Sortlib.kt` las funciones `mergesortInsertion` y `mergesortIterativo`. Ambas funciones reciben como entrada un arreglo de números enteros, 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 `runSortlib.sh`, agregando dos nuevos parámetros obligatorios a la línea de comandos y se modifica una de los parámetros anteriores. 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. La línea de comandos queda como:

```
>./runSortlib.sh [-t #num] [-s <secuencia>] [-n #n_1 ... #n_m] [-a <alg>] [-g <figura>]
```

La semántica de los nuevos parámetros de entrada es la siguiente:

- **-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. 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. Este nombre no deberá tener la extensión .png.

- **-a**: Se escoge la familia de algoritmos a ejecutar. El parámetro toma el argumento *alg* que tiene tres opciones. La primera opción es `On2` con la cual se ejecutaran los algoritmos *Bubblesort*, *Insertion-Sort*, *Selection-Sort* y *Shellsort*. La segunda opción es `nlgn` con la que se ejecutarán todas las variantes de *Mergesort*. La tercera opción es `all` con la cual se ejecutarán todos los algoritmos de ordenamiento de la librería. En caso de que la entrada tenga una opción diferente, entonces el programa termina con un mensaje de error.

A continuación se presentan llamadas válidas del programa cliente:

```
>./runSortlib.sh -t 2 -s random -n 500 -a nlgn
```

Este comando ejecuta los dos algoritmos Mergesort con secuencias de 500 números enteros, generados al azar, y cada algoritmo ejecuta dos veces la secuencia.

```
>./runSortlib.sh  -s inv -n 20 40 50 -t 3 -a On2 -g salidaInv
```

Al ejecutar este comando, el programa cliente aplica los algoritmos de ordenamiento simples a tres secuencias de números enteros de tamaño 20, 40 y 50, 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 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. Cada tamaño de secuencia se ejecuta varias veces con la opción **-t**, lo cual debe verse reflejado en la gráfica. Para generar la gráfica se debe hacer uso de la librería de visualización **libPlotRuntime**, cuyo código fuente será publicado 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 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.tar.xz*, donde X es el número de carné del estudiante. La entrega del archivo *LabSem2_X.tar.xz*, debe hacerse por la plataforma Classroom, antes de las 8:00 am del día viernes 28 de mayo de 2021.

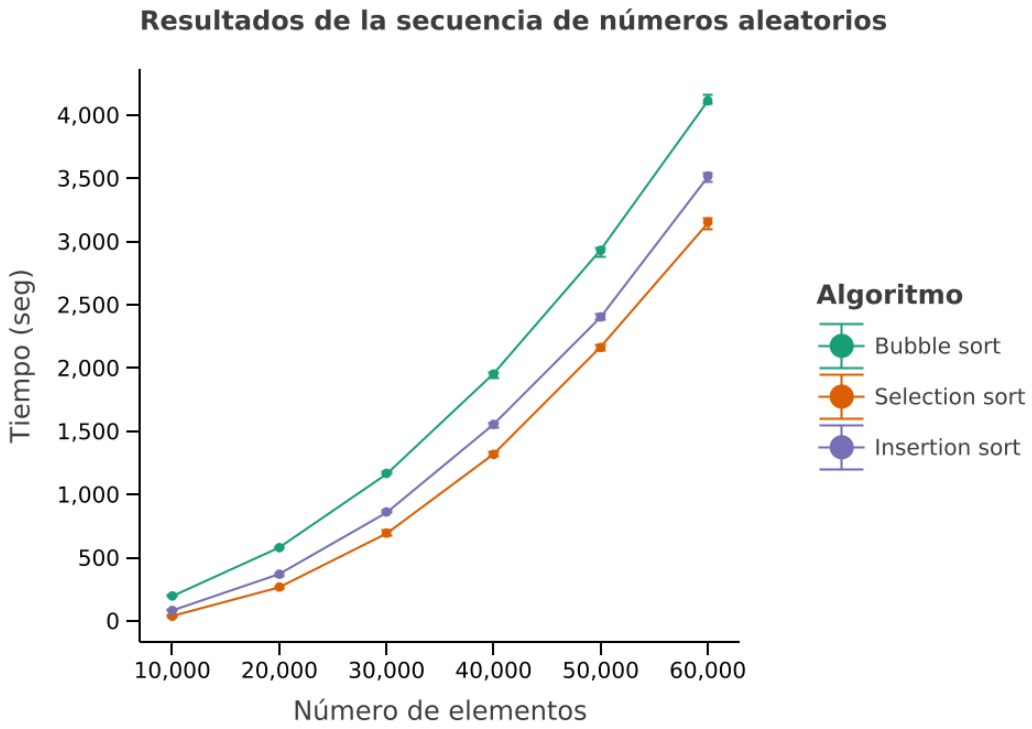

Figura 1: Ejemplo de una gráfica que muestra el comportamiento del tiempo de ejecución de tres algoritmos de ordenamiento, sobre seis secuencias de diferentes tamaño.

Referencias

[1] Brassard, G., and Bratley, P. *Fundamentals of Algorithmics*. Prentice Hall, 1996.

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

Guillermo Palma / gvpalma@usb.ve / Mayo 2021