Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI2692 - Laboratorio de Algoritmos y Estructuras de Datos II
Trimestre Abril-Junio 2023

Algoritmos de ordenamiento simples

1. Introducción

El objetivo de este laboratorio es la creación de una librería que contenga algoritmos de ordenamientos simples. También se quiere que cree un programa cliente que permita ejecutar los algoritmos de ordenamientos de la librería bajo una misma serie de pruebas. Con estas herramientas se podrá hacer un análisis del rendimiento de los algoritmos en diferentes condiciones.

2. Actividades a realizar

Este laboratorio tiene dos actividades. La primera es la creación de una librería de algoritmos de ordenamiento llamada `Sortlib.kt`. La librería debe implementar los siguientes algoritmos:

- Insertion Sort
- Selection Sort
- Shellsort
- Bubblesort

Los algoritmos de ordenamiento deben ser implementados siguiendo el pseudocódigo de [1]. Todos los algoritmos reciben como única entrada el arreglo a ordenar. Se le proporcionará el archivo `Sortlib.kt`, con la firma de los algoritmos a implementar.

La segunda actividad del laboratorio consiste en implementar un programa que sirva para evaluar los algoritmos de ordenamiento. El programa debe ser llamado `Main.kt`. Este programa cliente trabaja de la siguiente manera:

1. Genera una clase de secuencia con n elementos a ordenar.
2. Esta misma secuencia se aplica a todos los algoritmos de ordenamiento.
3. Cuando termina la ejecución de un algoritmo de ordenamiento, se toma el tiempo de ejecución del algoritmo.
4. El tiempo de ejecución de cada algoritmo, junto con el nombre del algoritmo, es mostrado por la salida estándar en el momento de finalizar el algoritmo.
5. Una vez terminado un algoritmo de ordenamiento, se debe verificar si los elementos del arreglo están ordenados correctamente.
6. En caso de que el algoritmo de ordenamiento no haya ordenado correctamente, se debe indicar un mensaje de error por la salida estándar y se aborta la ejecución del programa.

7. Los pasos anteriores, en donde se ordena un arreglo con un algoritmo de ordenamiento, lo llamamos intento. Se quiere que realice para todos los algoritmos de ordenamientos, un número de intentos t indicado por el usuario. Observe que cada intento se realiza con el mismo arreglo.

8. Si $t > 1$ entonces se debe indicar por la salida estándar el **tiempo promedio** junto con la **desviación estándar** de cada algoritmo de ordenamiento

Las clases de secuencia que debe generar el programa cliente, junto con su identificador, son los siguientes:

random: Secuencia de N elementos de tipo entero, generados aleatoriamente en el intervalo $[0\,..\,N]$.

sorted: Secuencia de N elementos de tipo entero, en donde los elementos están ordenados, tal que la secuencia tiene la forma $[1, 2, ..N]$.

inv: Secuencia de N elementos de tipo entero, en donde los elementos están ordenados de forma inversa, tal que la secuencia tiene la forma $[N, .., 2, 1]$.

zu: Secuencia de N elementos cero y uno, generados aleatoriamente.

media: Secuencia de N elementos de tipo entero, en donde los elementos de la secuenciaa tiene la forma $[1, 2, \ldots, \lfloor N/2 \rfloor, \lceil N/2 \rceil, \ldots, 2, 1]$.

La compilación del código del laboratorio debe generar un archivo llamado `TestSort.jar` para ser ejecutado con la JVM. Para la compilación puede crear un archivo `Makefile` si así lo desea. Si no crea un archivo `Makefile`, debe crear un archivo llamado `Leeme.txt`, en donde se muestre el comando a ejecutar para compilar su aplicación. La ejecución del programa cliente se va realizar por medio de un archivo ejecutable llamado `runSortlib.sh`. El `runSortlib.sh` debe ser un script de Bash ejecutable. La ejecución de `runSortlib.sh` se hace por medio de la siguiente línea de comando:

```
>./runSortlib.sh [-t #num] [-s <secuencia>] [-n #num]
```

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.
- **-t**: Número de intentos en el que se aplica el tipo de secuencia seleccionada
- **-n**: Número de elementos que contienen las secuencias a ordenar

Todos los argumentos son obligatorios, es decir, si falta alguno el programa de dar un mensaje de error por la salida estándar y terminar su ejecución. Los argumentos se pueden presentar en cualquier orden. A continuación se presentan llamadas válidas del programa cliente:

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

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

```
>./runSortlib.sh  -s inv -n 20  -t 3
```

Al ejecutar este comando, el programa cliente aplica los algoritmos de ordenamiento a una secuencia de 20 números enteros, tres veces.

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 *LabSem1_X_Y.tar.xz*, donde X y Y, son los números de carné de los estudiantes. La entrega del archivo *LabSem1_X_Y.tar.xz*, debe hacerse por la plataforma Classroom, antes de las 8:50 am del día viernes 19 de mayo de 2023.

Referencias

[1] Aho, A., Hopcroft, J., and Ullman, J. *Data structures and algorithms*. Addison-Wesley, 1983.

Guillermo Palma / gvpalma@usb.ve / Mayo 2023