Lesson 7. Algorithms with arrays

Develop a program to test the execution time of the different sorting algorithms for different array sizes = {1000, 2000, …, 50000}. The program must run 6 times each algorithm for an array size with different initial values. (Use the provided *sorting.basic.Algorithms* class.)

The program must generate five text files (*bubble.txt*, *selection.txt*, *insertion.txt*, *quick.txt*, *heap.txt*) with these contents:

```
<array size> <average time> <best time> <worst time>
<array size> <average time> <best time> <worst time>
…
```

(To generate a text file, print the messages on the screen and copy&paste them into a text file)

Represent graphically the results (array size *vs.* average time) with Microsoft Excel. Use two graphs; one for basic algorithms and one for complex algorithms.

Upload the .java files and the .xls files created.

Program structure

1. Create and open output files
2. For *i* = 1.000 to *i* = 50.000 (step 2.000)
   - 2.1. Declare and allocate array
   - *// Bubble sort*
   - 2.2. Define best time, worst time, added time
   - 2.3. For *j* = 1 to *j* = 6 (step 1)
     - 2.3.1. Initialize *i* with random values (`sorting.basic.Algorithms.init`) *
     - 2.3.2. start = get system time (`System.nanoTime()`)
     - 2.3.3. **Run bubblesort** (`sorting.basic.Algorithms.bubbleSort`)
     - 2.3.4. end = get system time (`System.nanoTime()`)
     - 2.3.5. Update best, worst, and added time
   - 2.4. Store best, worst, and average time (separated with blank spaces or commas)
   - *// Selection sort*
   - 2.5. …
3. Close files

* Notice that each execution of a sorting algorithm is performed with different data

Programming – Grado en Ingeniería Informática

Authors

Of the English version:
Juan Gómez Romero

Based on the work by:
Ángel García Olaya
Manuel Pereira González
Silvia Castro García
Gustavo Fernández-Baillo Cañas
Daniel Pérez Pinillos
Javier Ortiz Laguna
Álvaro Torralba Arias de Reyna

Universidad
Carlos III de Madrid