ALGORITHMICS

Bachelor of Software Engineering – Academic year 2018-2019

LAB GUIDE. SESSION 2

GOALS:

- **Sorting algorithms and their comparative study**

1. THREE BAD SORTING ALGORITHMS

In the Java files associated with this lab you have three sorting algorithms already studied in class (*insertion, direct selection* and *bubble*). **The specific code of the three methods should be included in the files by the student.**

There are bad algorithms because there are quadratic – $O(n^2)$ – in their best, worst and average case (except the insertion algorithm, which in the best case is linear O(n)).

To prove that all methods work correctly, a `SortingTests` class is provided. It has an argument `n` that is the size of the problem.

Try to understand in detail the operation of all the algorithms by analyzing the times for different sizes of the problem. It is provided a class `SortingMeasurements` that you should parameterize to correctly measure the respective times in the different cases.

The way to execute the codes may be the following if you don't use an IDE:

- `javac *.java`
- `java labs.lab2sorting.SortingMeasurements Type //Type is the type of vector (sorted, inverse, random)`

2. A BETTER SORTING ALGORITHM: QUICKSORT

In this case, you are going to study the *Quicksort* sorting algorithm. You should study it in detail, as it is a much more elaborate algorithm than others. **Complete the code** when necessary and analyze the times for different sizes of the problem. Finally, conclude whether the times obtained are the expected from the complexity in each case.

- **QuicksortMedianOfThree.java** → It has an argument `n`, that is the size of the problem. It is the one we saw in class.
- **QuicksortFateful.java** → It has an argument `n`, that is the size of the problem (it uses a bad pivot). It is usually a very bad choice.
- **QuicksortCentralElement.java** → It has an argument `n`, that is the size of the problem. In this case, instead of the median of three we use as the pivot just the central element. In addition to complete the file, you should create a file called **QuicksortCentralElementTimes.java** to measure times for this algorithm.

3. WORK TO BE DONE

To sum up, you should do the following:

- An Eclipse package. The content of the package should be:

  - All the files that were given with the instructions for this session but completing the fragments that were incomplete in **Bubble.java, Selection.java, Insertion.java** and **QuicksortCentralElement.java**

- A PDF document. The content of the document should be the following:
  - Four tables with times for each of the algorithms (Insertion, Selection, Bubble and Quicksort with the central element as the pivot). An example of one of the tables is below:
  -

n	*sorted(t)*	*inverse(t)*	*random(t)*
10000	…..	…..	…..
20000	…..	…..	…..
40000	…..	…..	…..
80000	…..	…..	…..
160000	…..	…..	…..
320000	…..	…..	…..
640000	…..	…..	…..
1280000	…..	…..	…..
…	…..	…..	…..
Until an exception is thrown			

  - A brief explanation for each of the tables to conclude whether the values make sense regarding the theoretical complexity.

*You should include in the ZIP file called **session02.<YOUR_UO>.zip** the following:*

- *The Eclipse package called **lab2** in the **<YOUR_UO>** Eclipse project.*
- *The PDF document called **lab2.pdf**.*

The deadline is one day before the next session of your lab group.