

# COMP114 - Assessment 3

## Semester 2 – 2010

### Comparing Algorithms

#### Assessment Information

Assessment Number	3
Contribution to Overall Mark	30%
Submission deadline	<i>Friday 23rd April 2010, 16.00</i>

---

#### Relevant Learning Outcomes for this Assessment

- 3 | Understanding of the factors involved in constructing experiments.
- 4 | Ability to select and apply appropriate experimental methods.
- 5 | Ability objectively to assess, analyse, and present experimental results.

#### Assessment Description – Overview

The aim of this assessment is to evaluate and distinguish two different methods for the *same task* which involves processing a collection of  $n$  integers. You are asked to carry out the following:

- a. Decide which of these methods is fastest and which slowest as assessed by a number of different criteria.
- b. To present evidence for the ranking presented under (a).

#### Assessment Description – Detail

The module web pages provide the following Java class which should be copied to your local filestore.

Algorithms.class

**N.B.** The *source* code is **not** provided.

This class contains four different methods,

```
static int[] AlgorithmA(int[] S, int n)
static int[] AlgorithmB(int[] S, int n)
```

each of which carries out the following,

Given a  $n$ -element array,  $S$ , whose value are *distinct* integers, the statement

```
T = Algorithms.AlgorithmX(S,n);
```

(where  $X$  is either  $A$  or  $B$ ) will return an  $n + 1$  element integer array in which  $\langle T[1], T[2], \dots, T[n] \rangle$  hold all of the elements in  $S$ : note that  $Algorithms.AlgorithmA(S, n)[i]$  and  $Algorithms.AlgorithmB(S, n)[i]$  will be **identical** whenever  $i$  satisfies  $1 \leq i \leq n$  since  $AlgorithmA$  and  $AlgorithmB$  perform the *same task* (but using *different* approaches).

The element  $T[0]$  will contain a count of the number of steps (i.e. runtime) taken by  $AlgorithmX$  in order to generate its result. The run-times of each method, i.e. the value that is stored in  $T[0]$  given different arrays of size  $n$ , varies quite considerably.

In this assignment you are asked to design and conduct an experimental study of these methods whose aim is to provide an assessment of their speed, i.e. your experiment should draw conclusions about which method is the “fastest”, and which the “slowest”. Your experimental study must also provide reasonable justification for the conclusions you reach.

## Hints and Suggestions

1. In order to produce input data with which to compare the different methods, you should use the random permutation generation method provided on the module resource page, i.e. the class *Permutation*.
2. In order to make reasonable conclusions about how the different methods behave, you will have to consider their behaviour with different *lengths of array* (i.e. different values of  $n$ ) and with several trials for each value.
3. There are a number of different measures you should consider use. The include (but are not limited to) the following:
  - a. *Worst-case* time, i.e. the **maximum** number of steps each method takes (as a function of the array length).
  - b. *Average case* time, i.e. for a given number of tests,  $M$ , sum of the different  $T[0]$  values for a given method divided by  $M$ .
4. **Other issues to consider**
  - a. Which do you think is more reasonable: to evaluate the methods with the *same* collection of permutations or to generate a new permutation for each method with each experiment?

- b. (Non-trivial) are you able to make any estimates of how the various methods perform as a function of the array size being considered?, e.g. do any of the methods seem to take “about”  $n$  or  $n \log n$ , or  $n^2$  steps, etc?
5. You will have to consider the following:
- 1. What range of  $n$  (the size of arrays/permutations) to use (smallest value?, highest value?, gap between successive values?)
  - 2. How many permutations should be generated for each  $n$  and with each method. (10? 20?, 50?, some function of  $n$  itself?)

### What should be Submitted

- a. The *source* code of the Java program through which your experiments are carried out.
- b. A summary of your experimental findings which should include your conclusions about the overall ranking of the two methods.
- c. A summary of the measures examined – i.e. average/worst/best case –; the range of sizes and number of trials carried out with each size.
- d. A report (in the form of a table and/or graphical plot) presenting your findings. These tables should be organised in the form of Table 1 below.

Table 1: Measure e.g. Average Steps

$n$ (Array length)	AlgorithmA	AlgorithmB
--------------------	------------	------------

- e. Any further observations arising from your experimental study.
- f. A completed and signed *Declaration On Plagiarism and Collusion Form*.

### How the work should be submitted

Items (a–f) should handed in to the *Student Office*. A cover sheet should indicate all of the following information:

- a. The Assessment *number*.
- b. Your **name** and University **e-mail address**
- c. Your lab group.
- d. The name of the *demonstrator* responsible for this group.
- e. Your degree programme, e.g. G400 Computer Science, G500 Computer Information Systems.