Prifysgol Cymru
Aberystwyth
The University of Wales

Department of Computer Science

CS21120: Data Structures and Algorithm Analysis
Assignment 2 - **Sorting**

1 Background

For this assignment you are required to do three things:

1. Understand and document some existing code.
2. Write new classes to fit into the existing structure.
3. Write a report that discusses the results that you obtain from running the new code in the existing framework.

2 The Experiment

I want you to implement a QuickSort and time how long it takes to run. Then optimise the QuickSort by applying a cut-off to the recursion at which point the QuickSort will switch to another sort (non-recursive, like an insertion sort).

You should run various experiments to determine the cut-off point where the switch to the non-recursive sort should be done. This will involve running with different data files, different cut-off points and drawing graphs.

You should then write up your experiments in a report that uses the format that is provided.

3 The tasks in detail

3.1 Understand some existing code

Obtain the code from the web server at the location http://pcbo.dcs.aber.ac.uk/blog/teaching/cs21120.

The code consists a main class which runs experiments together with a factory class that creates sorting routines, and some classes that implement sorting algorithms.

The SortDemo class will not be suitable for the type of experiments you want to run, it was designed for a different experiment, so you will need to modify it, or create a new main class.

For QuickSort, you will *HAVE* to create new classes (you can't just modify existing code) so that they work within the same framework because I will want to use my own main classes. Your classes must be able to be instantiated from the factory class without modifying the factory class.

3.2 Write a new class to fit into the existing structure.

You must use your understanding of the code to write a new sort routine to add to the sorting system. You must provide two implementations of the "Quick Sort". I suggest that you use one of the existing classes as a starting point.

The first implementation should be a basic standard implementation of the Quick Sort algorithm.

The second implementation will be an optimized Quick Sort based on the first that uses another sort routine when the dataset to be sorted is below a (runtime) specified number of items. You may use any other sort routine that you wish, but simple selection sort is usually used, due to its non-recursive nature. It is important that you can alter the specified number of items at run time, as some of the experiments you will be performing will rely on this fact. The parameter should either be read from a file, or passed in in some other way that does not require interaction by the user at run time, and does not require recompilation.

3.3 Discuss the results that you obtain from running the new code in the existing framework.

You must then take the results of running SortDemo, after modifying it to cope with your sorts, and provide an analysis of the results that you obtain from running the different sorts on different sizes of data sets. The data-sets you use must be named as they are in the following list, and have the relevant number of data items specified by each name.

N.B. You will also have to adjust the amount of heap space available to the Java virtual machine for some of the larger sorts - use the -Xmx and -Xms directives to the Java virtual machine when running the program to specify this.

Data file names and numbers of items as generated by `MakeDataFile` with no parameters

`test3.dat`	`1000`
`test3a.dat`	`2000`
`test3b.dat`	`5000`
`test4.dat`	`10000`
`test4a.dat`	`20000`
`test4b.dat`	`50000`
`test5.dat`	`100000`
`test5a.dat`	`200000`
`test5b.dat`	`500000`
`test6.dat`	`1000000`
`test6a.dat`	`2000000`
`test6b.dat`	`5000000`

3.4 The Experiments and Report

The experiments I want you to perform are to do with the optimization of Quick Sort. You should run the basic quick sort and determine how long it takes for each size of data set. You should then run your "Optimized" Quick Sort with varying values for switching to an alternative sort, using, for example, the values 5, 10 and 15 for the change of sort.

You should then produce a report on the differences in run time between the different types of sort that are presented, including the ones you were initially given, making particular note of the size of data files at which one sort becomes quicker than another.

You will want to run the experiments multiple times to get averaged results, as we are measuring elapsed time for the sorts rather than cpu time, so make sure that the machine that you are using has similar conditions when you run all the experiments.

The report should be in the style of a scientific report, presenting methodology, results and conclusions, having an abstract on the first page, and an executive summary at the end. Use the ACM SIG Proceedings Template to format your report, which can be downloaded from their website http://www.acm.org/sigs/pubs/proceed/template.html. You can choose whether to use Word or LaTeX, but you might find it useful to get started with LaTeX for the first time on this assignment.

The report should be a **maximum of 6 pages**, including diagrams, but not including source code. The source code should not be part of the report, but must be included in the submission.

In your report you must address the following issues:

- How long does it take for each type of sort to run?
- What are the differences in sorting times between the different variations?
- Where do the break points occur?
- When are simple sorts faster?
- Does the size of the data-set make a difference?
- Which is the best general purpose sort?

I suggest that you make use of a graphing tool to present graphs of your results, as well as a textual discussion. If you copy the output from SortDemo into a file and save it as CSV, then a spreadsheet program will read it in and you will be able to draw graphs.

4 The Submission

There is no physical submission for this assignment, it is entirely electronic via Blackboard. Please read these guidelines very carefully.

You must submit a report detailing the techniques used and the results of your analysis in a formal style. You will note that from the marking scheme that the majority of the marks are for the report. You will have to produce the code in order to write the report, but you are expected to produce a quality scientific report for the majority of the marks. This report must be submitted as a PDF file.

You must submit all *your* Java source code for all **new and modified** classes, which should be well structured and commented. You do not have to submit copies of the source code that I have provided on the web site.

The PDF file and the code must be made into a **zip** file for submission. Please only use zip, and not rar or other formats as blackboard cannot handle those and it breaks everybody's submission *sigh*.

This assignment is exempt from anonymous marking, and as such should not be made anonymous. You should use @author tags in your source code and other places where appropriate.

This assignment is due on Wednesday 4th May 2011 by 5pm. The submission system will be open for the whole of the period that the assignment is set — you will be able to submit early if you want to do so. Late submissions, even one minute after 5pm will not be accepted, and will be treated in line with Faculty policy.

If your account is locked then you will not be able to submit via blackboard, in which case you must burn everything onto CD and hand it in to reception before the deadline.

5 The Marking Scheme

This assignment is worth 25% of the marks for the course CS21120, therefore you are expected to spend somewhere around 30 hours working on it.

Implementation of Quick Sort 30%

Report on experiments 70%

6 Feedback

It is our intention to provide you with individual feedback on your work by approximately mid May, but we can't guarantee that this will be available before you sit the exam for this module.

Thus, we attempt to provide an alternative form of feedback very early. Within two or three days of the submission deadline, once we have checked all material submitted, we plan to release a “sample solution” to the problem. Students will then be able to compare their own work to that of our solution. If students have struggled on any issues, they will be able to see how we have tackled things.

We believe this is a very positive approach to providing rapid feedback.