COMPSCI.220.S1.C - Algorithms and Data Structures
Assignment 1 – Algorithm Analysis
Due: Sunday, 2007-08-12, 20:30.
Available from: www.cs.auckland.ac.nz/~mcw/Teaching/220/handouts/assignments/2007S2C/

This assignment is worth **100 marks** representing **8%** of your total course grade.

Requirements

1. (30 marks) You must analyse a given program with a rather poor user interface. More specifically, you will be given a program which is designed to measure resource use for seven sorting algorithms discussed in the textbook (bubblesort, heapsort, insertion sort, mergesort, quicksort, selection sort, Shellsort). Unfortunately, the designer of the program did not label the buttons properly. You must apply your understanding of the algorithms to determine the proper labelling.

   - Copy the Java jarfile `sd.jar` from the directory listed above, and run the application. Execute it and test it out. Notice that the button names do not give any indication which sort they will execute. Notice also, that if you create a small list, then that list is shown to you in the console window. In the unlikely event that a sort fails, a message will appear there as well.
   - Devise a plan of experiments which will enable you to match the particular algorithms to the button names. Execute your plan, taking careful notes as you go.
   - Describe the results of your experiment in a summary document in plain text or PDF format. Begin with a summary of the matching and then show the justification.

2. (70 marks) Devise an algorithm and implement it in a Java program to solve the following problem, similar to one often faced by an MP3 player. For our purposes, a song consists of the following data fields: title (a string), composer (which may be the empty string), running time (an integer). The user will call your program, giving a file containing a collection of n songs and an integer k with $1 \leq k \leq n$. Your program must find the k songs with longest running times, and output these songs in descending order of length of song. If songs have the same running time, then we use lexicographic order on the title, and then on the composer, to get a total ordering.

   - You should first make sure that your algorithm and implementation are correct! It is up to you to come up with test cases and the corresponding correct output. Usually, if a program contains an error, this can be detected on a small test case. You are allowed to share test cases with the class via the class forum.
   - Next you should consider efficiency. If k is close to n, then a different method may be better than the method used when k is small. So you may want to use a hybrid of some algorithms in the textbook. You are welcome to use the code available from the resources page of the course website in any way you choose.
   - Please see below for information on input and output formats.

Questions involving programming

- Java should be used for all programming questions unless special permission is obtained from the lecturer. You may freely use the classes available from the resources page of the course website. Template code may also be provided. You need not use it, nor stick to the template format, but it will probably save you time to do so.
- Your answer to each question should be a single .java file (containing all nonstandard classes you use). You may assume that the markers have access to all standard libraries and also the classes on the resource page, with the same directory structure as there.
- A sample input and output file for each question will be available. The markers may check the output with a text comparison program, so it must be in EXACTLY the right format.

Input and output format

For Q2, the following formats will be used. Sample input and output is available at the URL given above. The String.split() method may be useful in processing input.

Each line of the input file contains the song title (string), composer (string), running time in milliseconds (integers). These entries are separated by a single ampersand ("&") — this character does not appear in any of the data entries.

The output should be in exactly the same format (of course, the output will also be sorted).

Marking

- For Q1, of the 30 marks, 12 will be given for exposition and 18 for finding the correct answers.
- For Q2, of the 70 marks, 35 will be awarded for correctness and 35 for speed on a variety of input files to be devised by the lecturer. The pass mark for the correctness test will be at least 32. If you fail this test, you automatically score 0 for the speed test. The speed test marks are allocated according to the running time of the student answer compared to that given by the lecturer. Usually, we are not very tough - even a factor of 2 worse than the lecturer is sometimes given full marks in such questions. Correctness is more important than speed, but speed (or at least avoiding slowness) is also important if you want to get a high grade.

Submission

The due date is Sunday, 12 August 2007, 8:30 p.m. (ADB time) [then penalty linearly grows in time from 0% to 50% of marks on 5 August 2007, 8:30 p.m.; no submission afterwards]. Bonus also grows linearly in time from 0% to 10% of marks as time goes backwards by 5 days.

Submit your assignment to the Assignment Dropbox (ADB) system. You have to electronically hand in

- a file `summary.txt` or `summary.pdf` for Q1;
- a single Java program `ChooseSongs.java` for Q2.