

CS 2420 Fall 2008
Written 5 – 10 Points
Due October 27, 2008
Sort Detective

Objective

The primary objective of this assignment is for you to apply your theoretical knowledge of sorting algorithms to solve a problem of poor user interface design. More specifically, you will be given a program which is designed to measure comparisons and data movements for six sorting algorithms discussed in class. Unfortunately, the designer of the program did not label the buttons properly. You must apply your understanding of the general properties of the algorithms (and in some cases of the code used to implement them) to determine the proper labeling of the buttons.

The secondary objective of this lab is for you to gain experience writing a concise, but complete analysis of a system.

Background

As you know from class, if you double the size of the data set that you give to a quadratic algorithm, it does four times the work; by contrast, an $O(n \log n)$ algorithm does a bit more than twice as much; and a linear algorithm does only twice as much work. As you also know, the characteristics of the input data set can affect the expected performance of many of our sorting algorithms. Before you begin the lab, you should review the expected performance of the algorithms on various data sets.

The sorting algorithms under study include HeapSort, MergeSort, QuickSort, SelectionSort, ShellSort, and “other”.

Instructions – Warning: read all of the instructions before beginning!

1. Begin by copying the SortDetective application from the course web page. You have only the executable so you will need to run it on a windows machine. Execute it and play with it a bit. Notice that the button names (A-F) do not give any indication which sort they execute. Notice also, that you need to create a list before you can sort it. Notice also that there is an ‘Non-Random’ button that generates a list that is not random. The “worst possible input” creates data that is almost reversed of what you want. Important characteristics of sorts include: stability, oblivious, run time, and amount of data moved.
2. Devise a plan which enables you to match the particular algorithms to the button names. Hint: It may make sense to try to divide the sorts into initial groups and then to work on each group separately. Storing the counts in a spreadsheet may help you organize your attempts. Divide and conquer: it works for algorithms and it can work here, too!
3. Execute your plan, taking careful notes as you go.
4. Describe the results of your experiment in a summary document. Begin with a summary of the matching and then show the rationalization process that justifies it.

A Note on Writing

There is no coding in this lab. Thus, you should expect that a **significant** portion of the grade for this assignment will be determined by the quality of the writing of the report. This includes the completeness of the report, the clarity (and grammar) of the writing, and general presentation.

Some of the sorts are very difficult to distinguish. A carefully outlined experiment may compensate for an error in these cases if the writing makes it clear that your conclusions/guesses are substantiated by the data.

Finally, remember that your report needn't detail every experiment you ran. Rather, it should give sufficient information to justify your conclusions. It is possible to write a very short report that is completely correct if your experiments are well-chosen. After you learn the matching, you might consider whether there was a shorter way to arrive at your conclusion!

Be aware that the work is counted loosely – so while counts within the same sort are comparable, work counts between sorts are not.

Notes

- When completed, turn in your written assignment using eagle.cs.usu.edu in one of the following formats: .doc, .odp, .pdf.