New Mexico Tech
Department of Computer Science

Programming Assignment 4

CS122 Algorithms and Data Structures

Due 11:00AM, Wednesday, Dec. 8th, 2004

Objectives:

1. To gain experience with common sorting algorithms by comparing the running time of a “naïve” O(n^2) sort to a more complicated O(n log n) sort.

2. Learn how to implement come of the most common sorting algorithms.

Specifications:

1. In this assignment, you will be required to implement two well known sorting algorithms, a naïve, inefficient one, and a more complicated, more efficient one. You will time the running time of each algorithm on lists of 1,000, 5,000, 10,000, 50,000, 100,000, 250,000, and 500,000 elements and record the results in a table. Specifically:

    - For the simple algorithm, you may choose to implement either bubblesort, insertion sort, or selection sort.

    - For the more complex algorithm, you may choose between mergesort, heapsort, or quicksort.

    - In both cases, you should design your algorithms so that they will sort an arbitrarily sized array of integers, and they should sort the list in ascending order.

    - Remember, you only need to implement ONE algorithm from each category.

    - A single run of your program should load the list to sort from a file, sort the list using the “simple” algorithm and record the time as it does so, and then reload/reset the list and then sort it with the more complex algorithm, again recording how long it takes. Finally, the program should print out the time each algorithm took.

- You will be provided with framework code for timing, loading the data from the file, and implementing these algorithms. You will also be provided with a program which will automatically generate test-cases for you. You do not need to use either of these, but it will likely make your life easier if you do.

- The "table" you create need not (and probably should not) be implemented in your code. Use a spreadsheet or graph program (such as Excel), or record your results by hand and draw out the table.

- The following websites may be helpful in terms of finding descriptions of the required sorting algorithms. Remember to document any external sources you use to help you complete this project, including these:

  http://en.wikipedia.org/wiki/Sorting_algorithm
  http://linux.wku.edu/~lamonml/algor/sort/sort.html

NOTE: In some cases, the maximum test size called for may be too large as it requires a big chunk of stack space, which some systems may not supply. If you are unable to run the program because of this (it will crash on Windows and segfault on Linux if the array used is too large for the stack), change the "#define MAX_LIST_SIZE 500000" line to use the size of the next smallest test case (250,000). In your README file be sure to note that the final test was omitted because you could not run it on your machine.

NOTE 2: Because of the nature of O(n^2) algorithms, if you find that your test case with 100,000 entries takes much more than about a minute to complete, do not run the 500,000 entry test as it will take an obscene amount of time. If you omit the final test for this reason, please make note of it. Also, if you happen to have an older system and your running time starts getting huge, you may omit all subsequent tests. As a general rule, if a test will take more than about 20 or 30 minutes to complete, DO NOT test that size or any larger sizes. Again, please note any tests that were omitted, along with the reason why they were omitted, in your README file. It may be helpful to know that the 250,000 entry test will take about 6.2 times as long as the 100,000 entry test, and the 500,000 entry test will take a full 25 times as long as the 100,000 entry test.

2. As always, coding and documentation guidelines must be followed, and a README file must be submitted along with your source code. At a minimum, the README file must include **compilation instructions**, and **instructions on how to use your program**. Additionally, a discussion of any design issues you ran into while implementing this project and a description of how the program works (or any parts that don't work) is also appropriate content for the README file.

3. Your table should be submitted along with your code, and should show the times each algorithm took on each list size.

4. Note that the user-friendliness of your program, which includes things like displaying clear, concise instructions to the user detailing what the program is and how it should be used when your program starts, and the formatting applied to input prompts and output data, **is** important and **does** count. This is another case where it is entirely possible that your program will never display a prompt to the user and read input, but the program should still print instructions describing what it is, and output should still be well formatted.

Submission Guidelines:

Use whatever archiving software you prefer to archive your source, README file, and table together (unless you drew your table by hand, in which case turn it in during class on or before the due-date). Also be sure to include your colored copy of the graph image included with this project. The archive should be called '<your name>.Homework4.<ext>'. You may include a directory structure in your archive if desired, but it is not required. E-mail your archive to aroth@nmt.edu with the subject line of "[CS 122] Programming Assignment 4" by the due-date posted on the website. In the body of your e-mail, be sure to include your name and student-id number

Grading Criteria:

A correct implementation of the sorting algorithms is worth 60 points.

A complete, well-formatted, easy to read table of results is worth 10 points.

A useful and descriptive README file is worth 15 points.

Adhering to coding style and documentation guidelines is worth 15 points.

Points will be deducted for assignments turned in after the due-date.

Extensions:

For 10 bonus points each (for a total of 20 points max), implement the remaining O(n log n) sorting algorithms. If you do this you must also include entries for these algorithms in your table as well.