COP 4531 Algorithms

Spring 2007

Assignment 2

Due: 22 Feb 2007

Problem: In this assignment, you will implement programs to sort a list of integers in five different ways.

Sorting methods: You should create programs that will sort a list of integers using the methods given below.

1. InsertionSort.
2. MergeSort.
3. HeapSort.
4. Quicksort.
5. RandomizedQuicksort.

Running the program: Your program will be run as follows:

```
<executable-name> <input-file-name> <output-file-name>
```

The input file will contain a list of integers (not necessarily distinct), one per line. The output file should contain the integers sorted in ascending order. Each line of the output file will contain an element from the input followed by its index in the original input, as shown below. When elements are repeated in the input, please make sure that the index corresponds to the actual permutation produced by the sorting algorithm.

Sample input file	Sample output file

23
-8
21
	-8 2
	21 3
	23 1

Files: You should create a directory called `COP4531/hw2` on `linprog`. This directory should contain the following files, implementing the algorithm indicated by the file name.

1. *insertionsort.c*
2. *mergesort.c*
3. *heapsort.c*
4. *quicksort.c*
5. *randomizedquicksort.c*
6. *Makefile*: The makefile for this assignment. Typing make should create five executables, *insertionsort*, *mergesort*, *heapsort*, *quicksort*, and *randomizedquicksort*, corresponding to each of the above algorithms.

Performance evaluation: Evaluate the performance of each implementation as a function of the number of elements sorted, and compare the algorithms. This part of the assignment is open ended, and you should think of suitable tests to perform, so that you can come to useful conclusions. For example, some algorithms may perform well on data that are close to being sorted, while others may do well on random data. Some algorithm may have good theoretical time complexity in terms of CPU operations, but poor memory access patterns, indicated by a large number of cache misses. Some algorithms may have good average time for a given value of input size, but a large variance. Your tests should be able to point out these characteristics. You should describe your test methodology, present quantitative results, and also give a summary of qualitative conclusions, comparing the algorithms.

Turning in your assignment: All your C code files should have a time stamp before midnight, 22 Feb 2007. Please have read and execute permissions unset until at least midnight 24 Feb 2007. Read permission should be set on by noon 25 Feb 2007. Please turn in a hardcopy of your performance result by 5 pm, 22 Feb 2007, by either placing it in my mailbox or slipping it under my office door.

Grading: You will be graded on the following (i) correctness of your codes, (ii) performance of your codes, especially relative to those of others in the class, and (iii) the quality of your performance evaluation document (both, presentation, and technical quality).