Computer Science 2300: Lab 2

Due: Tuesday February 16, 2010

The purpose of this lab is to give you practical experience with “divide-and-conquer” approaches by implementing and analyzing the *merge sort* and *quick sort* algorithms.

1 Implementing the Merge Sort and Quicksort Algorithms

Write two separate programs that implement the recursive version of merge sort and quick sort, respectively. The programs should take as input a file of integers (which will be provided to you) and a bit that determines if the results should be printed or not. For example, consider the following text file:

```
<num.txt>
2
44
1
34
```

If the program is run as follows:

```
$:~ ./merge-sort num.txt 1  (or ./quicksort num.txt 1)
```

The result should be:

```
original list: [2] [44] [1] [34]
sorted list: [1] [2] [34] [44]
```

However, if the program is run with a 0 bit instead, no results should be outputted:

```
$:~ ./merge-sort num.txt 0 (or ./quicksort num.txt 0)
$:~
```

You can use the provided text files (4.txt, 8.txt, 16.txt) in lab2-files.tar.gz (on the course web-page) to test your programs.

2 Analysis and Comparison against Selection Sort

Mergesort has a run-time of $O(n \log n)$. Quicksort has complexity $O(n^2)$ in the worst case, but is also $O(n \log n)$ on average. However, quicksort has significantly less overhead. Selection sort is an algorithm that runs in O(n^2) time. In the tar file, there is a file called `selection-sort.cpp` and other text files, numbered from `400.txt - 50000.txt`. Run your merge sort and quick sort implementations on these files (do not print out the results when you do this!). Additionally, compile `selection-sort.cpp` and run it with the aforementioned files as input. Record the running times of each algorithm and plot the results. Show this to your TA to receive full credit.