Computer Science 2300: Lab 3

Due: February 6, 2008

The purpose of this lab is to give you practical experience with a "divide-and-conquer" *quicksort* algorithm. This is empirically one of the fastest and most widely used algorithms.

1 Implementing the Quicksort Algorithm

Write a program that implements the recursive version of quicksort. The program 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:

```
$:~ ./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 printed:

```
$:~ ./quicksort num.txt 0
$:~
```

The same set of input files that were used for Lab 2 (`400.txt-50000.txt`) will be used for Lab 3.

2 Analysis against Merge Sort

In the last Lab, we implemented mergesort which 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. The purpose of this lab is to figure out how much better quicksort performs empirically. Record the running times for quicksort using the input files(same as Lab 2) and plot along with the data for merge sort that you already have.

At the end, you should have a graph with two lines one for or mergesort and one for quicksort . Show this to your TA to receive full credit.