Computer Science 2300: Lab 2

Due: February 16, 2011

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

1 Implementing the Merge Sort Algorithm

Write a program that implements the recursive version of merge sort. 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
```

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
$:~
```

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 program.

2 Analysis against Selection Sort

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 - 9600.txt`. Run your merge-sort implementation 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. Using linear interpolation (ask your TA/UTAs if you don't understand), determine what O(n) behavior will look like, and graph this as well. You may assume that it takes 0.1 seconds to sort 400 numbers in the linear case.

At the end, you should have a graph with three lines: One that represents O(n), one that represents O($n\log n$), and one that represents O(n^2). Show this to your TA to receive full credit.