CS 240

Data Structures and Algorithms

Spring 2014

1 Lab 09

1.1 Goal

The goal of this assignment is to get a more thorough understanding of Big-Oh as well as searching and sorting and how they are related.

2 Assignment

You are to write a class named `DataProcessor`. This class will hold a vector of data that contains some floating point values. You are also required to implement a few functions:

- `readFromFile` – This will take a string (filename) as an argument and read a set of data from a file and load it into the working set. After a call to this function, the entire data contents of the file will be the only things in the working set. The file will consist of floating point numbers, one per line.
- `writeToFile` – This will take a string (filename) as an argument and write the current working set of data to a file, then resetting the working set back to empty. The file will consist of floating point numbers, one per line.
- `linearSearch` – This will take a floating point value as an argumet and perform a linear search of the data for the element, returning that element's location if found. If the element is not found, return

`DataProcessor::not_found`. (This is similar to string::npos, so if you know how that is implemented, this can be done in the same way.)

- `binarySearch` – This will take a floating point value as an argument and perform a binary search of the data for the element, returning the element's location if found. If the element is not found, return `DataProcessor::not_found`. (Hint: This function will not likely work as intended if called on an unsorted vector. This is by design – it is safe to assume that the person using the driver knows this.)

- `slowSort` – This will sort the data in the vector using a "slow" sort that *you have implemented in this assignment*. This sort is required to have an average-case runtime of $O(n^2)$.

- `fastSort` – This will sort the data in the vector using a "fast" sort that *you have implemented in this assignment*. This sort is required to have an average-case runtime of $O(n * lg(n))$.

You are welcome (and encouraged) to write each of these using helper functions. The signatures of the methods are required to be called from a Driver that has no knowledge of the size of the vector (or that you're even using a vector behind the scenes), therefore it should be able to interact with your DataProcessor class in a very streamlined API (one that has minimal arguments).

3 Timing

After you have written and tested each of these things, familiarize yourself with the timing example provided on the course website.

As always, your prelab will need to be completed before lab begins on Tuesday for you to have a chance at finishing the assignment on time.