Assignment 3
Sorting

Release Time	Due Date
10/19/2017	10/26/2017

Objectives

- To practice bubble sort, insert sort, selection sort and merge sort.
- Practice developing high-performance solutions.
- Compare the effect of data structures for these 4 sorting methods.

Problem Specification

Sorting is a very important part in data structure and algorithms. When you have a job interview, the HR always asks you to write some basic sorting algorithms without any extra information. So it is necessary to practice the sorting algorithms.

1) Generate N random lower case letters and store them in a linked-list. N is an integer value given by user (i.e., console input).

2) Sort the linked-list with 4 types of sorting algorithms (bubble sort, selection sort, merge sort, insert sort, and for insert sorting, please use *binary-insert* sorting).

3) When you are sorting the letters, instead of alphabetical order, you need to sort it according to your last name before considering alphabetical order. For example, if your last name is Gupta, the order of your sorting should be "guptabcdefhijklmnoqrsvwxyz". If N is 11 and the random letters are "uriojrwhthu", the output should be "uuthhijorrw".

4) Your output must include the value of N, your last name, random string and the sorted string.

5) Make sure your tool is readable, and has a friendly UI.

6) Analyze these 4 sorting methods (by time and space complexity, and some visible graph), and specify which is the best one

7) Redo the steps (1)-(7) with array instead of linked list, and compare the data structure's effect on different sorting algorithms. (Just compare the difference of time complexity and space

complexity between link-list and array for the same sorting algorithms. Compare them empirically as well as theoretically).

8) Testing: test your program with several strings, design your test cases so that you cover most (if not all) cases of user-input (this implies your application should be robust and fail-safe as much reasonably as possible).

Design Requirements

Code Documentation

For this assignment, you must include documentation for your code as generated by JavaDoc. You should have JavaDoc comments for every class, constructor, and method. By default, JavaDoc should output html documentation to a subfolder within your project (/dist/javadoc). Make sure this folder is included when you zip your files for submission. You do not need to submit a hard copy of this documentation.

Hint: http://stackoverflow.com/questions/4468669/how-to-generate-javadoc-html-in-eclipse

Coding Standards

You must adhere to all conventions in the CS 3310 Java coding standard. This includes the use of white spaces for readability and the use of comments to explain the meaning of various methods and attributes. Be sure to follow the conventions for naming classes, variables, method parameters and methods.

Testing

Make sure you test your application with several different values, to make sure it works.

Assignment Submission

- Generate a .zip file that contains all your files, including:
    - Source code files
    - Including any input or output files
    - Documentation of your code – e.g. using Javadoc if using Java
    - A brief report (in a pdf file) on your observations of comparing theoretical vs empirically observed time complexities. Note this report will include (a) a brief description of problem statement(s), (b) algorithms descriptions (if these are standard, commonly known algorithms, then just mention their names along with customization to your specific solution(s), otherwise give the pseudo-code of your algorithms, (c) theoretically derived complexities of the algorithms used in your code, (d) table(s) of the observed time complexities, and (e) plots comparing theoretical vs. empirical along with your observations (e.g. do theoretical agree with your implementation, why? Why not?).
- Don't forget to follow the naming convention specified for submitting assignments