CSC505 Jennings
Homework 2, Spring 2020
Please read, implement, and answer all 5 sections below.

This assignment may be done in groups of 2 or 3 people. Turn in one copy of the written part of the assignment on Gradescope, and enter the names of each team member during the submission process.

I. Names of team members:

_____

_____

_____

II. Implementation:

In this assignment, you will use 3 sorting algorithms: bubble sort, quick sort, and merge sort. You may implement these algorithms yourself, or you may use an existing library, or you may obtain source code. (You may make different choices for each algorithm.)

Please indicate which single programming language you will use for this assignment:

Java 12
Python 3.7
C (C11 or ANSI)
C++ 17

Here, provide a citation for each algorithm. Indicate which members of your team implemented it, or which library it came from, or where you obtained source code.

Bubble sort:

Quick sort:

Merge sort:

III. Task:

Using the data files in the HW2 directory of https://github.ncsu.edu/jajenni3/csc505, run each of the sort algorithms, capturing the amount of user process time[1] spent in (1) reading the input data into memory, and (2) performing the sort.

You will sort the lines of each file by the timestamp field, which is the first field of each line. A space separates it from the rest of the line. The sort should be ascending, so the earliest time occurs first in the output.

Put your code and data into a GitHub repository on github.ncsu.edu in a branch called "HW2" (upper case). Give read access to the teaching staff (3 TAs and Dr Jennings – email addresses on Piazza). Put the repo URL here:

Code and data repo: _____

IV. Data collection and presentation:

For each sort algorithm, do the following:

1. Create a data table of user process times with these columns: file name, number of input lines, data load time, sort time, sum of load and sort times. Report all times in micro-seconds.
2. Plot the growth of the data loading time as a function of data size (in lines).
3. Plot the growth of the sorting time as a function of data size (in lines).
4. In another table, the "meta-data" table, indicate (1) how many times the sort was executed before timing data was recorded, (2) how many executions were performed, (3) whether the highest and lowest times were dropped; (4) whether the mean or median time is the one reported in the data table; (5) the operating system and version; (6) the CPU type, speed, and cache sizes; and (7) the type and size of disk holding the data.

[1] User process time is measured by the Unix API `clock()` and is accessible from most programming languages.

V. Analysis questions:

1. Describe the data structure you used to hold the data in memory.
2. Reproduce here the comparison function you used (show your code if you implemented it, or used existing source; otherwise, show the parameters passed to the sort routine and explain what they do).
3. For each algorithm, what order of growth did you expect to see? Compare to what was observed.
4. Comment on the relative amounts of time needed to load data versus sort data.
5. Three data files contained the same number of input lines (1 million = 10^6). Were any differences observed in execution times across these 3 files? Explain why or why not, for each algorithm.
6. How does your quicksort choose a pivot value?
7. Which of your algorithms sorts in place, and which needs additional memory?
8. Which of your algorithms is implemented recursively, and which iteratively?
9. Which of your algorithms implements a stable sort? How do you know?
10. Suppose the data were much larger, and did not fit easily into memory.
    a. Which of the 3 algorithms being studied would you use to implement a solution?
    b. Describe (in just a few sentences) how such a solution would work.
    c. Based on your measurements, estimate how long (in user process time) your proposed solution would require to sort *one trillion* (= 10^{14}) lines of data like the sample data. Assume that at most 10^8 lines will fit in memory at once, in total (input and working memory combined). State any other assumptions. Show how you obtained your answer.