CSCE 221 Cover Page

Programming Assignment #4

First Name: Seth Last Name: Barberee UIN: 324009275

Any assignment turned in without a fully completed coverpage will receive ZERO POINTS.

Please list all below all sources (people, books, webpages, etc) consulted regarding this assignment:

CSCE 221 Students	Other People	Printed Material	Web Material (URL)	Other
1.	1. Timothy Ebringer	1.	1.	1.
2.	2. Nathan Leake	2.	2.	2.
3.	3.	3.	3.	3.
4.	4.	4.	4.	4.
5.	5.	5.	5.	5.

Recall that University Regulations, Section 42, define scholastic dishonesty to include acquiring answers from any unauthorized source, working with another person when not specifically permitted, observing the work of other students during any exam, providing answers when not specifically authorized to do so, informing any person of the contents of an exam prior to the exam, and failing to credit sources used. Disciplinary actions range from grade penalties to expulsion. Please consult the Aggie Honor System Office for additional information regarding academic misconduct – it is your responsibility to understand what constitutes academic misconduct and to ensure that you do not commit it.

I certify that I have listed above all the sources that I consulted regarding this assignment, and that I have not received nor given any assistance that is contrary to the letter or the spirit of the collaboration guidelines for this assignment.

Today's Date: 4/19/18

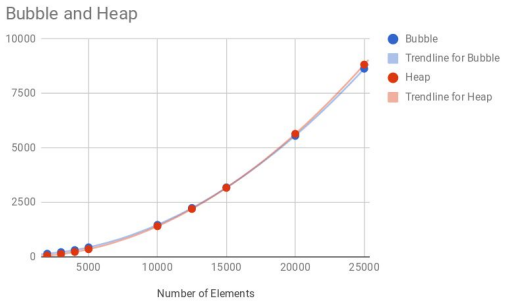
Printed Name (in lieu of a signature): Seth Barberee

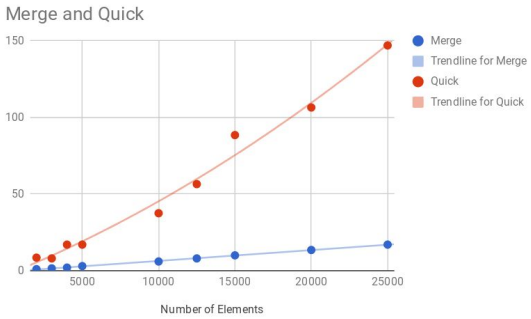
1. Introduction: In this section, you should describe the objective of this assignment.
   For this assignment, we are tasked with programming four different sorting algorithms: Bubble, Heap, Quick, and Merge. Then, we test the efficiency of the algorithms with differing amounts of data ranging from sorted, reverse-sorted, and randomly sorted.
2. Theoretical Analysis: In this section, you should provide an analysis of the complexity of each sorting algorithm on the different input types. What do you expect the complexity should be for each algorithm and input type?
   a. **Bubble -** $O(n^2)$ **for all input types**
   b. **Heap -** $O(nlogn)$ **for all input types**
   c. **Merge -** $O(n)$ **for Sorted/Unsorted and** $O(nlogn)$ **for random**
   d. **Quick -** $O(nlogn)$ **for Sorted/Unsorted and** $O(n^2)$ **for random**
3. Experimental Setup: In this section, you should provide a description of your experimental setup, which includes but is not limited to
   a. Machine specification
      - **Intel Core i7-6280HQ (4 cores, 8 threads) @ 3.6 GHz**
      - **16 GB DDR4 RAM**
      - **Arch Linux x86_64 4.15.15-1**
   b. How did you generate the test inputs? What input sizes did you test? Why? Did you use something other than an array for your data structure? If so, why?
      - **Sorted: For loop that started at 0 up to max elements and counted up. Inserted loop counter into the vector.**
      - **Reverse Sorted: For loop that started at max elements and counted down. Inserted loop counter into the vector.**
      - **Random: Test inputs were generated with the random math function in C++.**
      - **The code for this is in TimeTest (commented in the code where it is). These inputs were inserted into a vector that would be fed into the respective sort functions. Heap sort was fed each input separately. The reason a vector was used was in order to not have to deal with resizes.**

c. How many times did you repeat each experiment?
Each sort was performed two times and an average was taken for each data point in the graphs below.

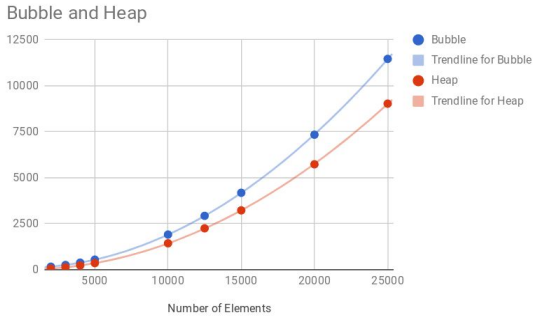
4. Experimental Results: In this section, you should compare the performance (running time) of the sort operation for the four different sorting algorithms to one another and to their theoretical complexity.
    a. Make a plot showing the running time (y - axis) vs. the number of elements (x - axis). You must use some electronic tool (matlab, gnuplot, excel, ...) to create the plot – hand written plots will NOT be accepted.

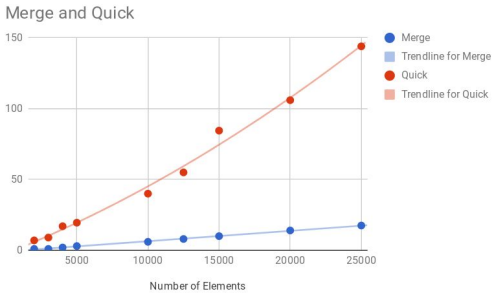
Sorted



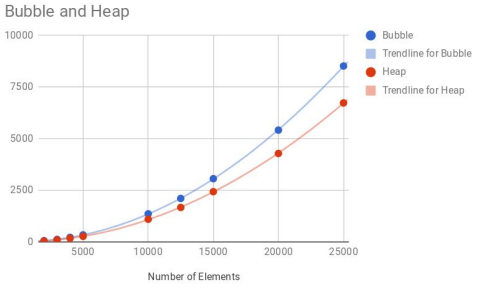


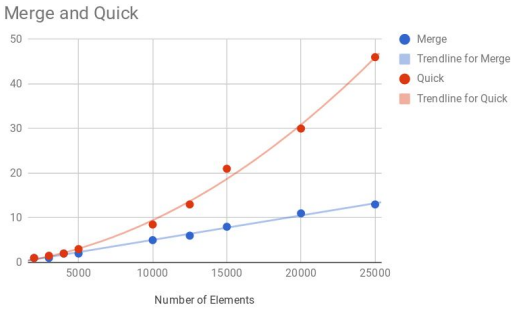
Reverse Sorted





Random





b. Provide a discussion of your results, which includes but is not limited to:
    i. Which of the sorting algorithms performs the best? Does it depend on the input? **No matter the input, the merge sort performed the fastest.**
    ii. To what extent does the theoretical analysis agree with the experimental results? Attempt to understand and explain any discrepancies you note. **The sorted inputs caused Bubble and Heap to have virtually the same times resulting in the same curve. Besides that, the rest of the graphs support the theoretical analysis earlier in the report.**