CSCE 221-200: Honors Data Structures and Algorithms Assignment Cover Page Spring 2021

Name:	Alexander Akomer
Email:	alexakomer@tamu.edu
Assignment:	PA 4
Grade (filled in by grader):	

Please list below all sources (people, books, webpages, etc) consulted regarding this assignment (use the back if necessary):

CSCE 221 Students	**Other People**	**Printed Material**	**Web Material (give URL)**	**Other Sources**
1.	1.	1.	1. https://www.geeksforgeeks.org/quick-sort/	1.
2.	2.	2.	2.	2.
3.	3.	3.	3.	3.

Recall that TAMU Student Rules define academic misconduct 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 penalty to expulsion.*

"On my honor, as an Aggie, I have neither given nor received unauthorized aid on this academic work. In particular, I certify that I have listed above all the sources that I consulted regarding this assignment, and that I have not received or given any assistance that is contrary to the letter or the spirit of the collaboration guidelines for this assignment."

Signature:	Alexander Akomer
Date:	04/14/2021

Alexander Akomer

PA 3 Report

1. *Introduction:*

This assignment is an early exercise in calculating efficiency of different sorting algorithms. We were tasked with creating three different sorting algorithms – one following the method of heap sorting, one following the method of quick sorting (pivoting at the first element), and the last following the method of quick sorting (pivoting at a random element). To compare the time efficiency of the three, sets of random, increasing, and decreasing values were generated and sorting using each different sorting algorithm. This was repeated with 100 values, 1000 values, 10,000 values … until failure memory or time failure (segmentation fault). This was done to measure the time complexity of each structure so that the advantages and disadvantages of each could be easily displayed. The results are as follows.

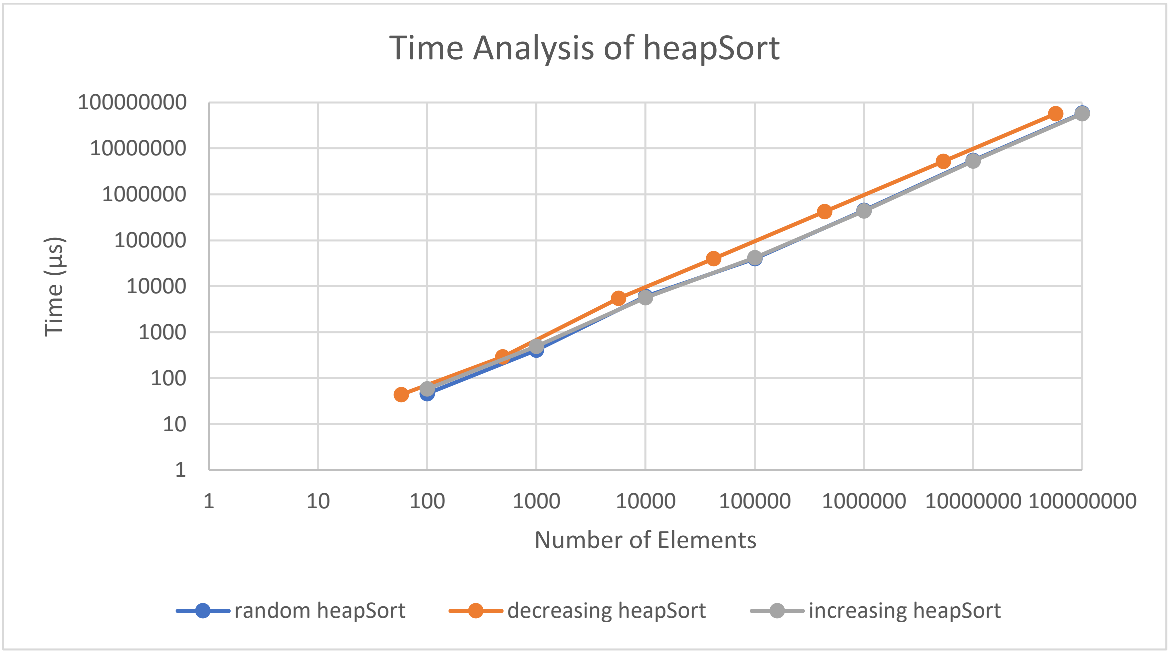
2. *Theoretical analysis:*

Three different sorting algorithms were required to be implemented for this assignment. These sorting algorithms are heapsort, quicksort (pivot at first element), and quicksort (pivot at random element). The heapsort is known to have an average, best-case, and worst-case of O(n log n). The typical quicksort algorithm is known to have an average and best-case time complexity of O(n log n) and a worst-case time complexity of O(n^2). These complexity values are true of the quicksort that chooses pivots at random as well as the quicksort that chooses pivots from the first element; however, the first element implementation is stunted in this experimental analysis as its worst case scenario occurs whenever all values are sorted in increasing or decreasing order (and therefore the first quicksort should appear to be O(n^2) for all but random valued trials).

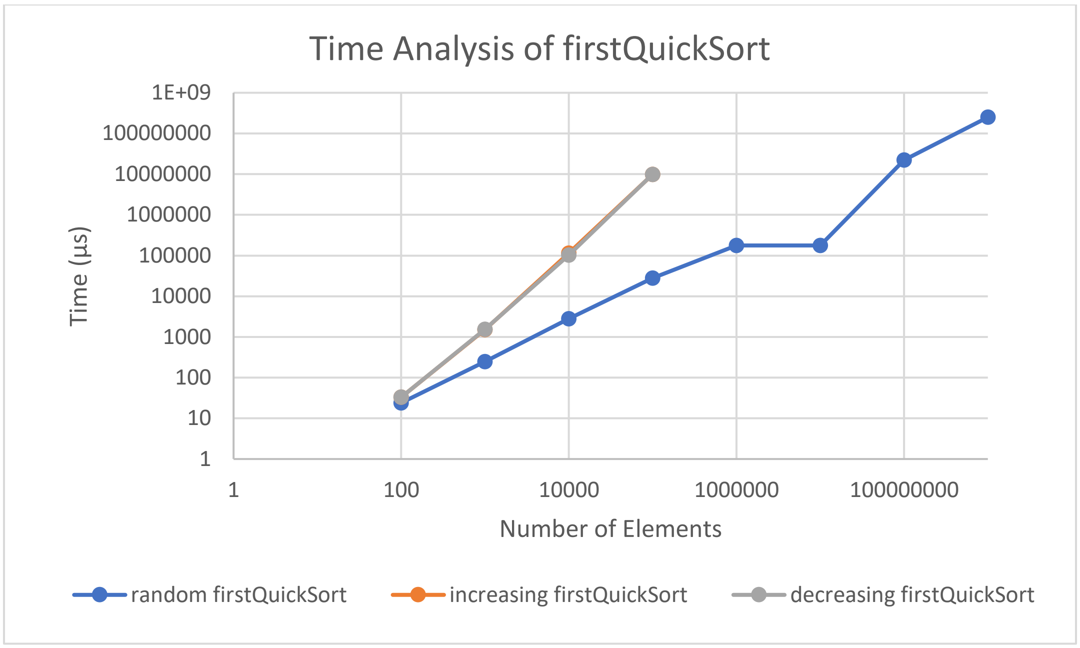
3. *Experimental Setup:*

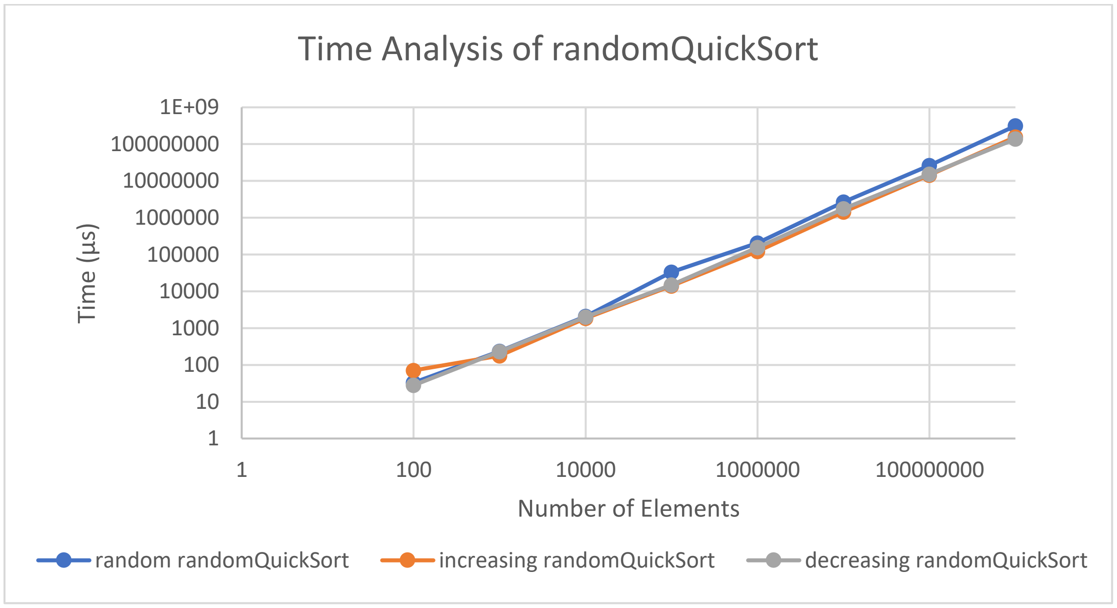
The machine the program was run on for my trials was a Surface Book 2 with 16 GB of RAM, an i7-8650U @ 1.90 GHz, and 106 GB / 475 GB free on the C: drive. My timing mechanism was the code attached on Canvas, which included the chrono package. This package allowed me to run a high-resolution clock and output the runtime in microseconds for each trial. Because the instructions only called for one trial of each function call for each sorting algorithm, that is the all of the data that was recorded, which may lead to faults in accuracy.

4. *Experimental Results:*



Alexander Akomer





After testing the three different sorting algorithms, the following experimental conclusions were made:

Heapsort: Worst/Average/Best Case: O(n log n)

First Quicksort: Worst Case: O(n^2)

Average/Best Case: O(n log n)

Random Quicksort: Worst/Average/Best Case: O(n log n)

The runtime data closely matches the theoretical analysis of each sorting algorithm. The graphs show that heapsort shows a very similar asymptotic analysis for the average, best, and worst case as mentioned in the theoretical analysis. This trend follows an O(n log n) time complexity, also similar to that described in the theoretical analysis. The first quicksort algorithm shows an average time complexity of O(n log n) whereas the worst case shows a complexity of O(n^2) which is clearly shown to be in the cases in which the values are given in increasing or decreasing order. The random quicksort shows an average time complexity of O(n log n), similarly to the first quicksort but slightly slower (by a multiplicative constant). However, the worst cases of the random quicksort appear to be much better because the increasing and decreasing order are the worst possible cases when choosing the first element as the pivot. Because of the randomness of the random quicksort method, a more suitable pivot is chosen in these situations which reduces the frequency of worst-case scenarios. Therefore, the worst-case is not more efficient, but occurs less often (it doesn’t occur when the values are already given in increasing or decreasing order). The only discrepancies between my experimental results and theoretical analyses is shown in the randomly ordered random quicksort, but this is due to the nature of randomness as well as the limited trials attempted. Despite the discrepancies, every algorithm is bounded exactly as depicted in the theoretical analyses.