**Lab 3 Report**

Matthew Sabo

CISC Program, Harrisburg University

CISC 233: Essential Algorithms

Professor Rohrbaugh

March 16, 2023

**Overview**

The objective of the lab was to implement various types of sorting algorithms into an existing program and then implement two other types of data that would be inputted into the algorithms themselves. The algorithms we needed to implement was insertion sort, selection sort, merge sort, and quick sort. They types of data that would be inputted into the algorithms are a random list of integers, a near sorted list of integers, and a reverse sorted list of integers. With these types of data, we can test the big-O efficiency of these algorithms in different scenarios.

**Limitations**

Some of my code that isn’t working correctly is the time that it took to run the trials. For some reason it is inconsistent and I can tell that its running since the comparisons and swaps make sense. I have no idea why that happened, and I think its because of CPU clock speed but I’m not too sure since I never edited any of the code that dealt with calculating the time of the algorithm. Other than a 600-line main function the code ran relatively smoothly.

**Results**

**Table

Description automatically generated**Table

Description automatically generated

**Table

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generated**

**Discussion**

Some algorithms are better for certain types of data than other types of algorithms. To start the bubble sort algorithm has a big-O of O(n^2) and this can be shown in all 3 data configurations where the comparisons and swaps increase at a linear rate for all 3 configurations however the time for them show that the algorithm is adaptive since the times for the almost sorted configuration is significantly lower than the other 2 configurations. For Selection sort the efficiency of this algorithm is like bubble sort where it is O(n^2). However selection sort is not adaptive making the time difference between the near sorted configuration and the other two configurations much lower than bubble sort since the algorithm can not adapt to the configurations that are inputted. The efficiency of insertion sort is also O(n^2) and since it is adaptive the efficiency of the almost sorted configuration is a lot lower than the other 2 configurations. In the data given in the results section this is the start of the time calculation breaking in the data so unfortunately I cant tell exactly how the efficiency changes but I can tell that the algorithm is more efficient on the sorted algorithm since the average swaps and comparisons are a lot lower than the other 2 configurations. For merge sorting the efficiency is O(n log n). This is because merge sort is a recursive algorithm making the efficiency of the algorithm better than the previous algorithms however merge sort is not adaptive, so the configurations of data does not affect the results of the algorithm heavily. Quick sort is like merge sort where it is a divide and conquer algorithm. However the efficiency of quick sort is O(log n) which is significantly faster than any of the previous algorithms. This can be shown in the random sort where the number of swaps and comparisons are the lowest of any algorithm or configuration for the highest number of elements in the list. However since the algorithm is not adaptive the difference between the number of swaps and comparisons in the almost sorted and reverse sorted configurations are significantly higher than the random configuration.