Assignment Six Report: Sorting Algorithms

Nicole Chu
Schmid College of Science and Technology
Chapman University
Orange, CA 92866
Email: nichu@chapman.edu

***Abstract*—A review of my final assignment for Data Structures and Algorithms in which I implemented four different sorting algorithms (Bubble, Selection, Insertion, and Quick) and output the amount of time that elapsed for each of them. I used a sample text file to test my program containing six doubles to be sorted (3.6, 1.2, 5.7, 2, 7.8, 3.5).**

1. Introduction

To implement the 4 different sorting algorithms in my project (Bubble sort, Insertion sort, Quick sort, and Selection sort), I referenced code provided during class, our textbook, and some online resources. After coding the various sorts, I created a test text file with 6 doubles to be sorted. I struggled to find a way to print time stamps as required in the assignment rubric but instead found a way to use the clock() function to calculate the time elapsed from the start to the end of executing each sort. The results I observed were quite different from what I expected.

1.1. Time Differences of Sorts

While testing my program, Bubble sort and Selection sort took an average of 0.005 milliseconds to execute with each test of the same text file. In some tests, Selection sort was slower than Bubble sort and in other tests it was faster than Bubble sort. Insertion sort almost always took 0.005 milliseconds to execute as well except during one test case in which 0.01 milliseconds had elapsed from the start to the end of execution.

The execution time that shocked me the most was that of Quick sort. I had expected this sort to execute in the shortest amount of time as its runtime is faster than those of the other sorts. However, it continuously exhibited the slowest lapse of time. At its fastest, Quick sort executed in 0.007 milliseconds and at its slowest it operated in 0.023 milliseconds.

1.2. Comparing Algorithms

I was very confused as to why Quick sort did not run as fast as expected. Perhaps my text file required different data to test. However, if it had run as expected, then I could better understand the tradeoffs between these different sorting algorithms. Quick sort was the most complicated sorting algorithm for me to implement. Despite looking at numerous resources, I still found myself confused about how the algorithm worked. On the end of the spectrum, Bubble sort and Selection sort seemed incredibly easy to implement. Had my program run as expected, then I could see how the tradeoff between these two sorts would be complicated/uncomplicated implementation and faster/slower runtime.

1.3. Empirical Analysis

Although empirical analysis allows one obtain real results about their program, it does not encompass every possible test that could be run. For instance, I could not test how the sorting algorithms would perform on larger data sets as I did not have the time nor the resources. Perhaps if I could perform such a test, I would have observed different results.

2. Conclusion

Although it has a runtime of O(n log n) which is faster than the other sorting algorithm runtimes, Quick sort did not perform faster than Bubble sort, Selection sort, nor Insertion sort as I had expected. In theory, Quick sort would run faster than Bubble sort but requires more complicated implementation. Although my tests failed to show that Quick sort is the most efficient sorting algorithm, a different, larger data set may be required to illustrate this conclusion.

References

[1] Michael Shell, School of Electrical and Computer Engineering, Georgia Institute of Technology (provided IEEE template)

[2] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.