Sorting Algorithms Analysis

Tristan M. Chilvers
2288893
chilvers@chapman.edu

***Abstract*—This paper discusses a brief analysis of four sorting algorithms: BubbleSort, InsertionSort, SelectionSort, and QuickSort. Taking an empirical analysis approach, a simple program was made to compare the runtimes between each algorithm and prove previous thereotical analysis of these runtimes to be true.**

I. Introduction

Empirical analysis is crucial for quantifying runtimes of computer algorithms. It helps put into perspective just how much time it takes for a computer to process so much information, and how vital it is to carefully consider which algorithm to use within given constraints.

II. Runtimes from Theoretical and Empirical Analysis

A. Expectations from Theoretical Analysis

Three of the sorting algorithms (BubbleSort, InsertionSort, and SelectionSort) have a similar runtime of $O(n^2)$, leading me to believe that they will have a similar runtime when tested. However, the final results of the program proved to be just the opposite. While they all share the same worse case scenario, that does not necessarily mean an average performance will have the same runtime. Furthermore, QuickSort is a recursive function of runtime $O(n(log(n)))$ which proved theoretically and in the experiment to have a runtime that is substantially faster than the previously mentioned algorithms.

B. Results from Empirical Analysis

Each algorithm ran through a double data set of 180,435 numbers, a large enough data set to show the difference in runtime between each algorithm. The results are shown below:

- BubbleSort: 178.005 seconds
- SelectionSort: 88.8178 seconds
- InsertionSort: 49.1516 seconds
- QuickSort: 1.42192 seconds

As shown above, BubbleSort has a drastically slower runtime compared to all other algorithms. What is surprising to me however, is how SelectionSort proved to have a runtime that is half the time of BubbleSort. Even though both algorithms share a very similar theoretical runtime and code, the minor difference between them is enough to cut the runtime in half. To me, the time difference between BubbleSort and QuickSort is extremely drastic. QuickSort is 125 times faster than BubbleSort, this is because of partitioning the data set and recursively sorting through the data set.

III. Tradeoffs

Of course with each algorithm there is a tradeoff. Generally, the algorithms with faster runtimes have a more difficult time of implementation. While BubbleSort has the worst runtime of the four, it is the quickest to implement thus proving use for small data sets that need to be sorted without very much time to spend implementing the algorithm. The same also goes for SelectionSort, but compared to BubbleSort, it is not as easy to implement. While QuickSort has an amazing runtime, if there is limitations on hardware it is then not the best idea to use QuickSort for data processing.

IV. Choice of Language

This program used the C++ language to test the runtimes. Theoretically, if the runtimes were compared to the same algorithms in a language such as Python, it would prove to be faster. This is because C++ is a compiler language, which generally prove to be faster than an interpreter language like Python.

V. Shortcomings of Empirical Analysis

While Empirical Analysis is great for showing the runtimes with hard data, it is often times hard to implement and test when there is too much data. It is more efficient to analyze an algorithm through Theoretical analysis, thus saving time and cost. Furthermore, if the hardware is limited in power it will be impossible to test through Empirical Analysis, leaving us to use a Theoretical Approach instead.

VI. Conclusion

This assignment overall was very interesting and helped put into perspective just how much faster certain algorithms compared to others. Observing the difference in runtime between $O(n^2)$ and $O(n(log(n)))$ allows me to better appreciate the runtime of the algorithms I develop and better improve my programming skills as a developing computer scientist.