Assignment 6

Rielle Dizon

18 May 2018

I. INTRODUCTION

Sorting algorithms come in many different forms. Some are asymptotically better than others. Assignment 6 deals with implementing four different sorting algorithms. The four algorithms sorted for this assignment were: Bubble Sort, Insertion Sort, Quick Sort, and Shell Sort.

II. RESULTS

These algorithms were tested against test files with 15 to up to 10,000 items within a text file. The variation in run-times did grow as the data sets got larger. The results in this document are from a file with 10,000 different items to sort.

A. Bubble Sort

Average run-time:

$$O(n^2)$$

Bubble Sort performed decently, taking 121.84ms to sort 10,000 items. Despite it being one of the slower algorithms to complete with a quadratic run-time, it is one of the easiest and most straight-forward to implement. Bubble Sort is fine to use with smaller data sets as the variation of run-time does get smaller, and the fact that it is the easiest algorithm to implement.

B. Insertion Sort

Average run-time:

$$O(n^2)$$

Insertion sort was similar to Bubble Sort, taking 122.60ms to sort 10,000 items. It is more complex than Bubble Sort, and therefore is ranked the lowest out of the four in choosing sorting algorithms.

C. Quick Sort

Average run-time:

$$O(nlogn)$$

Shockingly, Quick Sort did not have the fastest run-time to sort 10,000 items. It took Quick Sort 0.57ms to complete. However, in terms of implementation, Quick Sort is third on the list, making it a good especially for larger data set.

D. Shell Sort

Average run-time (depends on gap reduction; in this case):

$$O(n^2)$$

Shell Sort had the fastest run-time, sorting 10,000 items in .51ms. The difference between Quick Sort and Shell Sort is small. Shell Sort was more difficult to implement due to having to understand the explanation of its gap and its gap reduction.

III. LANGUAGE CHOICE

C++ was the perfect language to test and implement these languages in as C++ is a compiled language. Interpreted languages like Python may be a little more difficult to implement algorithms and therefore may have some influence on the program performance.

IV. SHORTCOMINGS

A shortcoming of this analysis was the fact that certain data sets could have made some algorithms run faster than they did.

V. CONCLUSION

In conclusion, the best algorithm to use for smaller data sets is Bubble Sort due to its easy implementation, and the best algorithm to use for larger data sets is Quick Sort, as it is still easier to implement than Shell Sort.