Tony Bautista

Dr. Rao Ali

CPSC 350 Data Structures

May 12, 2022

Assignment 8

This paper is based on my findings to evaluate three array sorting algorithms: Merge Sort, Bubble sort, and Insertion Sort.

The three array sorting algorithms were implemented using a dynamic array in a fully integrated C/C++ development program, tested in Clion and Atom IDE. For this assignment, two sorting algorithms were required; Merge Sort and Selection Sort, and I picked Bubble Sort for a third and final sorter.

Merge Sort is an algorithm that "divides a list into two halves, recursively sorts each half, and then merges the sorted halves to produce a sorted list." Selection sort is an algorithm that "selects the smallest element from an unsorted list in each iteration and places that element at the beginning of the sorted list." Bubble sort is an algorithm that "repeatedly swaps the adjacent elements if they are in the wrong order."

The user inputs the text file name at the command prompt or terminal window to run the program. The program then reads the text file's first line, which tells the program how many elements to parse. Once the lines are read, the program then sorts the information and uses a timer to output the start, end, and elapsed time of each sorting algorithm. The three input files used had one thousand elements, ten thousand elements, and the last one had one hundred thousand elements of data. Each sorting algorithm was run multiple times with similar results, and for this paper, I used one test and not an average of multiple runs.

The test for the one thousand elements had elapsed runtimes that were close to each other, but once I got into the ten thousand and one hundred elements, the times grew increased dramatically.

The first test of one-thousand elements showed that merge sort came in first at .223 ms, followed by selection sort at 1.046 ms, and bubble sort came in third at 2.769 ms.

The second test of ten-thousand elements had merge sort first at 2.594 ms, followed by selection sort at 100.106 ms and bubble sort was last at 343.275 ms.

The third test of one-hundred thousand elements had merge sort come in first at 29.552 ms, followed by selection sort at 9635.27 ms, and bubble sort came in third at 37963.8 ms

I was surprised to see the time difference between the smaller and larger elements. However, it does make sense since Merge Sort's worst-case time complexity is 0(n log N), Selection Sort’s worst-case time complexity is O(n^2), and Bubble Sort also has a time complexity of O(n^2). I knew that merge sort would be the better of the three, but the difference between selection sort and bubble sort was rather alarming

The trades off differ based on the amount of sorted data used and the sorting algorithm. As the amount of data being sorted increased, the times for selection sort and bubble sort grew in line with the data being sorted, while merge was the overall winner.

Regarding programming languages, I cannot say that one will be better than the other because this is the first time I have used data structures in any language.

The shortcomings of this empirical data would be the type of IDE, computer, the efficiency of the code, and the data being sorted. In addition, I would need to run more tests to get a more precise measurement of what each algorithm can do.

Reference:

https://learn.zybooks.com/zybook/CHAPMANCPSC350AliSpring2022/chapter/3/section/2

https://www.bigocheatsheet.com