Data Structures and Algorithms

Professor Adrian Rusu

Arjun Aravind

**Project 1: Sorting Algorithms Empirical Time Efficiency Comparison**

Introduction

In this project, the goal is to compare the effectiveness (measured in data input size and speed) of several different sorting algorithms. In order to do this, it was necessary to test different-sized data sets of elements: from 10,000 elements, 50,000 elements, 150,000 elements, 200,000 elements, and 275,000 elements. The sorting algorithms used to run these tests were Bubble sort, Selection sort, Insertion sort, Merge sort, Quick sort, and Shell sort. Each data set was tested with each sorting algorithm five times and the average of the five runtimes was recorded. This was to ensure that one particular failed or erroring run would not skew the results. The results were then compiled into graphs in order to serve as visuals for analysis.

Bubble Sort Implementation

Bubble sort is a sorting algorithm that repeatedly runs through a list of elements, compares adjacent elements, and swaps them if they are in the wrong order. It reruns this process until no more swaps are needed, indicating that the list is sorted.

A graph with a line

Description automatically generated

With this implementation of Bubble sort, it is clear that it took a considerably large amount of time to sort the data set, and drastically increased as the size of the dataset increased. It ranged from 0.19 seconds to 151.83 seconds. Bubble sort can be categorized as a Θ(n2), meaning that time increases quadratically as the number of elements increases.

Selection Sort Implementation

Selection sort is a sorting algorithm that works by repeatedly finding the minimum element from the unsorted part of the list and moving it to the beginning. It does this by iterating through the list, comparing each element with the minimum found so far, and placing it in the front of the list. This process is repeated until the entire list is sorted.

A graph with a line

Description automatically generated

Selection sort was significantly quicker than Bubble sort. Though it also increased at an exponential rate as the size of the datasets increased, the rate was significantly less than that of the Bubble sort. In terms of complexity, Selection sort is also categorized as a Θ(n2).

Insertion Sort Implementation

Insertion sort is a sorting algorithm that works by taking each element from an unsorted list and inserting it into its correct position within a sorted list. It does this by repeatedly comparing an element with the elements before it and shifting larger elements to the right until the correct position is found. This process is repeated for each element until the result is a sorted list.

A graph with a line

Description automatically generated

Insertion sort had similar time results to Selection sort, only slightly faster. It also increased at an exponential rate as the size of the datasets increased. In terms of complexity, Insertion sort is also categorized as a Θ(n2).

Merge Sort Implementation

Merge sort is a sorting algorithm that works by repeatedly dividing an unsorted list into smaller halves until each half contains only one element. Then, it merges these smaller sorted lists back together to create larger sorted lists, ultimately resulting in a completely sorted list.

A graph with a line

Description automatically generated

Unlike the previous sorting algorithms, Merge sort was exceptionally faster at every size of the dataset. It increased at an ascending rate, but it was clearly not as aggressive as exponential growth. The complexity category of Merge sort is Θ(nlog(n)), which is much more efficient than Θ(n2).

Quick Sort Implementation

Quick sort is a sorting algorithm that works by selecting a "pivot" element from the list and partitioning the other elements into two sub-arrays, according to whether they are less than or greater than the pivot. It then recursively sorts these sub-arrays.

A graph with a line

Description automatically generated

Quick sort was the fastest running algorithm with a range from 0.0014 seconds to 0.027 seconds. It was consistently the fastest running algorithm across all trials with all other sorting algorithms. Its rate of change is slightly greater than a linear growth model. In terms of complexity, it is categorized as Θ(nlog(n)), similar to that of Merge sort, but significantly more efficient.

Shell Sort Implementation

Shell sort is a sorting algorithm that's like a more efficient version of Insertion Sort. It works by dividing the list into smaller sublists and then sorting these sublists using a regular Insertion Sort.

A graph with a line

Description automatically generated

Shell sort was a sorting algorithm that had very similar results to that of Merge sort, marginally slower on all input sizes except for 275,000 elements, where it was marginally faster than Merge sort (Shell sort was 0.0576 seconds compared to Merge sort’s 0.0588 seconds). Though its complexity categorization is Θ(n1.25) on average, which means that it becomes less efficient as the size of the dataset increases, an anomaly has occurred by Shell sort performing faster than Merge sort for 275,000 elements.

Conclusion

A graph showing the growth of a company

Description automatically generated

As you can see from the graph above, the various sorting algorithms have drastically different speeds and efficiencies, as the size of the dataset increases. To summarize, there are 3 apparent groupings of sorting algorithms with regard to their efficiency. First is Bubble sort, which is in a group of its own and is the slowest of all the sorting algorithms. The second group contains Selection sort and Insertion sort, which operate very similarly to each other and have respectively similar speeds. The next group consists of Merge sort, Shell sort, and Quick sort, which all run very efficiently and fast, even at extremely large dataset sizes. Quick sort especially stands out as the fastest sorting algorithm.

Citation

GeeksforGeeks. (2023, January 10). *Shellsort*. GeeksforGeeks.

<https://www.geeksforgeeks.org/shellsort/>

OpenAI. (2023). ChatGPT (September 25 Version) [Large language model].

<https://chat.openai.com/chat>

*(I used ChatGPT to create the different Sorting Algorithms)*