­Shyam Ramesh

CSS 342

Professor Dimpsey

UW Bothell

**Program 4 Report: Algorithm Efficiencies**

**Introduction**

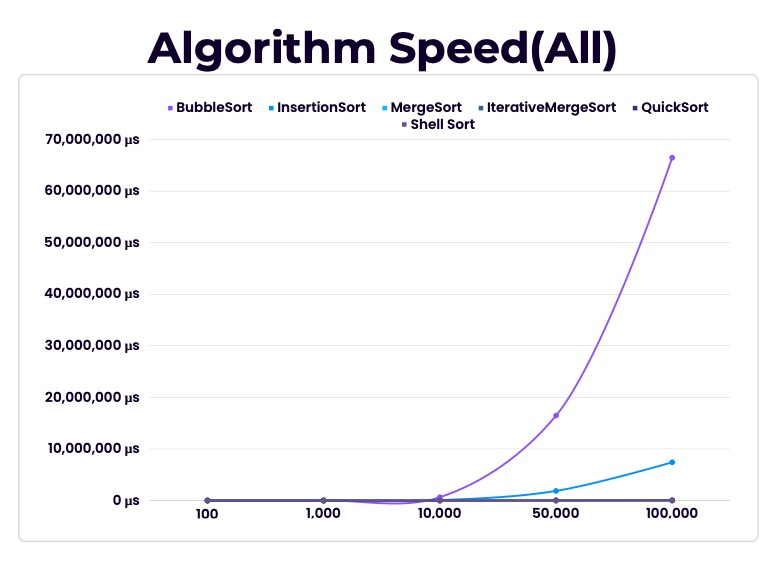
There are many different sorting algorithms. While they may have a seemingly small difference, an extra ‘n’ can make a drastic difference in efficiency when dealing with large data sets. This project aims to create and test a variety of different algorithms and compare their results in a report format.

**The Data**

For each algorithm at each input size, I ran that specific point anywhere from three to ten times, then I took the median of those results and placed the results in this table.

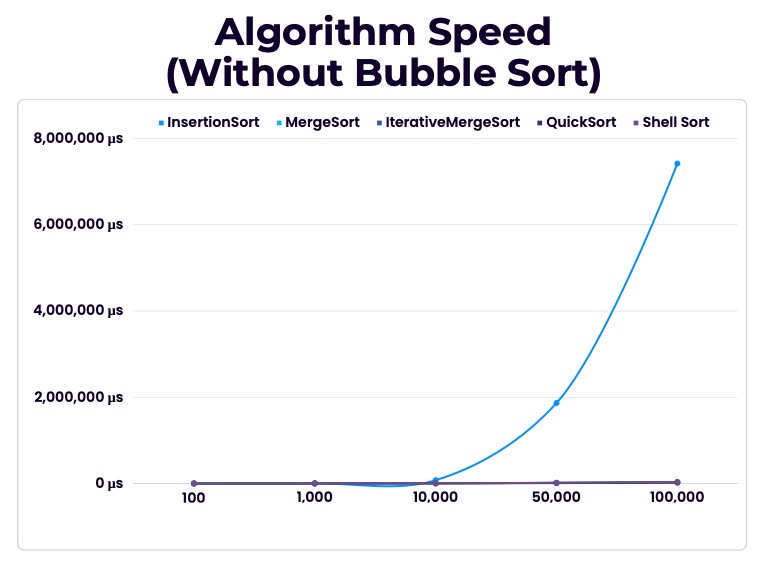
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Algorithm**  **Input size** | **BubbleSort** | **InsertionSort** | **MergeSort** | **IterativeMergeSort** | **QuickSort** | **Shell Sort** |
| **100** | **212 μs** | **25 μs** | **61 μs** | **24 μs** | **29 μs** | **28 μs** |
| **1,000** | **11,316μs** | **2,234 μs** | **696 μs** | **290 μs** | **372 μs** | **584 μs** |
| **10,000** | **644,477 μs** | **76,161 μs** | **3,315 μs** | **4,226 μs** | **2,039 μs** | **3,766 μs** |
| **50,000** | **16,472,824 μs** | **1,865,853 μs** | **14,542 μs** | **15,905 μs** | **8,544 μs** | **16,119 μs** |
| **100,000** | **66,475,843 μs** | **7,421,271 μs** | **30,006 μs** | **33,399 μs** | **17,574 μs** | **34,889 μs** |

From those results I inputted them into a graph maker and created these graphs. For each of the three graphs the x-axis is the input size and the y-axis is the time in microseconds.

****

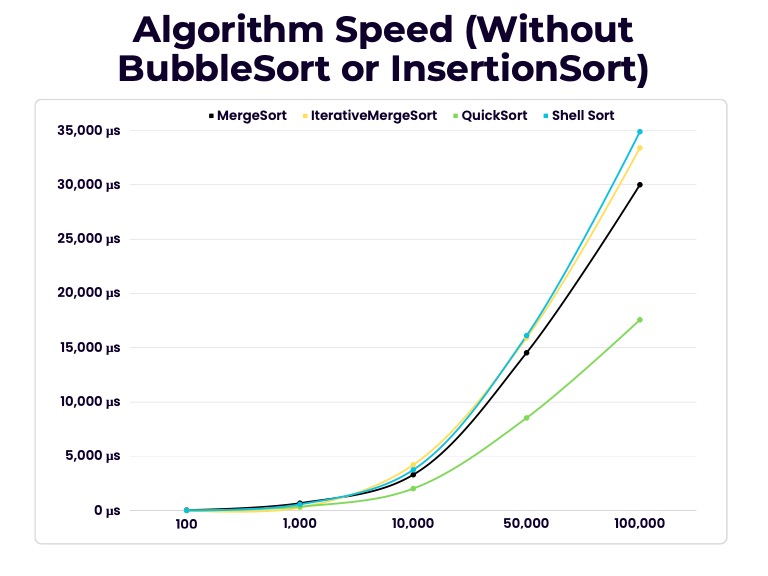
Input Size(Int)

Time in Microseconds



Input Size(Int)

Time in Microseconds



Input Size(Int)

Time in Microseconds

**Analysis**

From the data I collected I have come to these conclusions:

* BubbleSort is the slowest sorting algorithm tested.
* InsertionSort is the second slowest sorting algorithm tested.
* Both BubbleSort and InsertionSort are much slower than the other sorts.
* QuickSort is the fastest sorting algorithm tested.

Here are the time complexities of these sorting algorithms:

|  |  |  |  |
| --- | --- | --- | --- |
| Sort Type / Time Complexity | Average | Best | Worst |
| BubbleSort | O(n²) | O(n) | O(n²) |
| InsertionSort | O(n²) | O(n) | O(n²) |
| MergeSort | O(n log n) | O(n log n) | O(n log n) |
| IterativeMergeSort | O(n log n) | O(n) | O(n log n) |
| QuickSort | O(n log n) | O(n log n) | O(n²) |
| ShellSort | O(n log n) | O(n) | O(n log n) |

**Conclusion**

In conclusion, the results of this experiment show that the sorting algorithms with the best time complexity are MergeSort, IterativeMergeSort, QuickSort, and ShellSort. These algorithms are much faster than BubbleSort and InsertionSort, which have an average time complexity of O(n²). As a result, BubbleSort and InsertionSort are not practical for sorting large datasets.

Overall, the best sorting algorithm for a given application will depend on the specific requirements of that application. However, if you need to sort a large dataset, then MergeSort, IterativeMergeSort, QuickSort, or ShellSort are all good choices.

**Citations**

Time Complexities – (Google Bard) Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to algorithms. MIT press, 2009.