Project 1

Sorting Algorithms

Pratham Aggarwal

CSCI 41

TABLE:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Size | Bubble Sort | Selection Sort | Insertion Sort | Merge Sort | Quick Sort |
| 200,000 | 364712 | 416756 | 159923 | 1311 | 440 |
| 400,000 | 1175426 | 1612203 | 616783 | 1150 | 704 |
| 600,000 | 2434165 | 3435998 | 1368164 | 1785 | 685 |
| 800,000 | 4108522 | 5056533 | 2437323 | 2154 | 711 |
| 1,000,000 | 6218498 | 5951689 | 3833844 | 2807 | 1018 |

**Note**: Execution time is in millisecond

The table shows the execution times for various input sizes for various sorting algorithms. All algorithms experience a rise in execution time as the amount of the input does, but at varying rates. The slowest algorithms are Bubble sort and Selection sort, with execution durations that are much longer than those of Insertion sort, Merge sort, and Quick sort. While slower than Merge sort and Quick sort, Insertion sort is more effective than Bubble and Selection sorts. The quickest method overall is Merge sort, closely followed by Quick sort. As the input size increases, both of these algorithms exhibit a comparatively steady and predictable increase in execution time. In conclusion, Merge sort and Quick sort are more effective algorithms for bigger datasets, but Insertion sort, with its simplicity and effectiveness, might be a good option for smaller datasets.

Below are the graph and explanation for each algorithm.

**Bubble Sort:**

The graph above shows the Bubble sort algorithm's execution time in milliseconds for various input sizes. The algorithm's execution time grows as the amount of the input does as well. The size of the input and the algorithm's execution time are seen to have a linear connection. As an illustration, the execution time is 364712 milliseconds for input sizes of 200000, whereas it is 6218498 milliseconds for input sizes of 1000000. This shows that the Bubble sort algorithm's execution time increases quickly as input size increases. Therefore, bubble sort may not be the most effective technique for huge datasets as a result.

**Selection Sort:**

The graph keeps track of how long the Selection sort algorithm takes to run for various input sizes. The execution time considerably increases along with the magnitude of the input. An input of size 200,000 is sorted by the method in 416,756 milliseconds, but an input of size 1,000,000 is sorted in 5,951,689 milliseconds. The rise in execution time is exponential in nature rather than linear. For instance, if the input size rises from 200,000 to 400,000, the execution time roughly triples. Given that the execution time gets longer and longer as the input size increases, it can be concluded that Selection sort is not an effective method for sorting huge datasets.

**Insertion Sort:**

The Insertion Sort algorithm's execution time for various input sizes is displayed in the graph. When compared to Selection sort, the execution time grows along with the size of the input but much more slowly. For instance, the method sorts a size of 200,000 in 159,923 milliseconds whereas a size of 1,000,000 requires 3,833,844 milliseconds to sort. It is possible that the Insertion sort algorithm is more effective than the Selection sort method for sorting huge datasets since the increase in execution time is essentially linear. Insertion sort still has its drawbacks, and it might not be appropriate for very big datasets. However, due to its simplicity and effectiveness, Insertion sort can be a good option for datasets that are not very large.

**Merge Sort:**

The graph shows how long the Merge sort algorithm takes to complete for various input sizes. The execution time grows from 200,000 to 1,000,000 as the input size does. This pattern suggests that the time complexity of the merge sort method increases with the volume of the input data. With execution times of 1311, 1150, 1785, 2154, and 2807 milliseconds for input sizes of 200,000, 400,000, 600,000, 800,000, and 1,000,000, respectively, the execution times for different input sizes vary significantly. The execution timings exhibit a varying pattern, with certain input sizes requiring less time than the preceding input size. However, the overall pattern indicates that, for the specified input sizes, the Merge sort method is slower than the Quick sort algorithm.

**Quick Sort:**

The graph displays the Quick Sort algorithm's execution time for various input data size ranges. The execution time grows from 200,000 to 1,000,000 as the amount of the input does. This pattern suggests that the time complexity of the Quick sort algorithm increases with the volume of the input data. With a maximum execution time of 1,018 milliseconds for an input size of 1,000,000, the execution time varies a little bit with bigger inputs. The Quick sort method is effective for small to medium-sized data sets, but it may become less effective for bigger data sets, according to the graph.