Analysis Report for sorting algorithms

Data for all sorting algorithms:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Data Set | | | | | | |
| Sorting Algorithm | 100 | 1000 | 10000 | 50000 | 100000 | 500000 | 1000000 |
| Bubble Sort | 99 | 12417 | 916950 | 23634099 | 88658445 | 1709441161 | 2508329319 |
| Insertion Sort | 52 | 1036 | 97904 | 2466071 | 7661328 | 71133089 | 128017801 |
| Merge sort | 79 | 1880 | 4482 | 23693 | 45454 | 764114 | 1386559 |
| Iterative Merge Sort | 34 | 385 | 11589 | 60318 | 118535 | 632754 | 1246894 |
| Quick Sort | 9 | 168 | 1401 | 11231 | 16309 | 222032 | 714686 |
| Shell Sort | 12 | 344 | 4210 | 16509 | 36116 | 232085 | 745610 |

Graphs:

Bubble Sort:

Theoretical Complexity: O(n^2)

Empirical Data: The data shows a significant increase in time as the size of the data set increases, which aligns with the theoretical complexity.

Insertion Sort:

Theoretical Complexity: O(n^2)

Empirical Data: Similar to Bubble Sort, the data shows a significant increase in time as the size of the data set increases, which aligns with the theoretical complexity.

Merge Sort:

Theoretical Complexity: O(n log n)

Empirical Data: The data shows a more gradual increase in time as the size of the data set increases, which aligns with the theoretical complexity.

Iterative Merge Sort:

Theoretical Complexity: O(n log n)

Empirical Data: The data shows a similar trend to Merge Sort, which aligns with the theoretical complexity.

Quick Sort:

Theoretical Complexity: O(n log n) in the best case and O(n^2) in the worst case

Empirical Data: The data shows a more gradual increase in time as the size of the data set increases, suggesting that the pivot selection was good and the best-case complexity was achieved.

Shell Sort:

Theoretical Complexity: Ranges from O(n log n) to O(n^(3/2)) depending on the gap sequence

Empirical Data: The data shows a more gradual increase in time as the size of the data set increases, suggesting that a good gap sequence was chosen.

From the empirical data, it appears that Merge Sort, Iterative Merge Sort, and Quick Sort are the most efficient algorithms for large data sets, aligning with their theoretical time complexities of O(n log n). Bubble Sort and Insertion Sort show significant increases in time as the size of the data set increases, which aligns with their theoretical time complexities of O(n^2). Shell Sort’s efficiency seems to depend heavily on the chosen gap sequence.

It's important to note that the efficiency of sorting algorithms can vary based on various factors, including programming language, hardware, compiler optimizations, and characteristics of the dataset (such as size, initial order, and distribution of elements).

Adjusting these implementation decisions, such as experimenting with different gap sequences in Shell sort or pivot selection strategies in Quick sort, could lead to variations in the observed efficiency of the algorithms. Therefore, understanding the impact of these implementation choices is crucial for optimizing sorting algorithms for different scenarios and datasets.

As for the best algorithm in terms of efficiency, it would depend on the specific use case. If the data is nearly sorted, Insertion Sort can be very efficient. For larger, random data sets, Merge Sort, Iterative Merge Sort, or Quick Sort (with a good pivot selection) would likely be the best choice.