JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Comparative Study and Analysis of Sorting Algorithms and Their Application in Operating Systems

Atul Sharma[1], Akshit Bhandari[2], Dr. Shilpa Bhalerao[3], Prof. Shruti Lashkari[4]

Department of Computer Science and Information Technology Acropolis Institute of Technology and ResearchvBypass Road Manglia Square, Indore, Madhya Pradesh 453771

Email: atulsharma220538@acropolis.in

akshitbhandari220505@acropolis.in

shilpabhalerao@acropolis.in

shrutilashkari@acropolis.in

Abstract

This research paper presents a brief analysis of major sorting algorithms in terms of their time and space complexities in varying circumstances. The paper compares Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, and Heap Sort on the basis of performance in best case, worst case, and average case. Space complexity is also analyzed to determine the amount of space each algorithm would need. Graphs are shown to compare algorithm efficiency for different input sizes. Moreover, the paper also studies the implementation of these algorithms within Windows and Linux operating systems. It emphasizes how they are significant in process scheduling, file system organization, and memory management. Certain OS-level implementations and optimizations are discussed here, showing how sorting algorithms benefit system performance. Real-world test cases are used to analyze the effectiveness of the algorithms in operating system operations.

The article concludes by presenting recommendations for the choice of appropriate algorithms for given OS functions and provides potential future areas for optimization and research into operating system design and performance.

Keywords: Sorting algorithms, time complexity, space complexity, big data analytics, memory utilization, operating systems,Linux.

1. Introduction

The importance of effective sorting algorithms in big data analysis cannot be overemphasized. In the age of exponential data expansion, the selection of a suitable sorting algorithm is crucial to maximizing the performance of data processing operations. This paper provides an in-depth study and comparative evaluation of popular sorting algorithms with emphasis on their time and space complexities in best-case, worst-case, and average-case scenarios.

Aside from their applicability in data analysis, the study further investigates the practical use of sorting algorithms in contemporary operating systems like Windows and Linux. It considers their effect on critical system-level operations like file system structure, memory allocation, and process scheduling. Through a comparison of the performance of various algorithms in these applications, the research reveals their practical significance and efficiency in real operating system systems. The results are intended to help system architects and developers choose the best possible algorithms that optimize efficiency, responsiveness, and system performance in general.

2. Literature Review

Previous studies have emphasized the need for algorithmic optimization in managing massive datasets.[1] Notable research includes foundational work by Knuth, Sedgewick, and Cormen et al., highlighting the importance of selecting appropriate sorting strategies. Comparative studies suggest that while simpler algorithms like Bubble Sort and Selection Sort are easier to implement, more complex algorithms like Merge Sort and Quick Sort perform better with large datasets.Researchers such as Knuth and Sedgewick have provided foundational insights into sorting algorithms' theoretical aspects, while studies by Cormen et al. and Sedgewick have emphasized the importance of considering worst-case, best-case, and average-case complexities.[3]Despite existing efforts, a comprehensive comparative study that focuses on both algorithm performance and their practical application in operating systems is still limited, forming the gap this study addresses.

3. Methodology

The methodology involves a systematic analysis of sorting algorithms, focusing on Bubble, Selection, Merge, Insertion, and Quick sort, due to their relevance in big data analytics scenarios. A thorough literature review is conducted to understand previous findings and methodologies related to sorting algorithm analysis.A comprehensive dataset representing varying sizes and complexities is curated to test algorithm performance, including synthetic and real-world datasets.[2] Key metrics such as overall running time, number of comparisons, and memory usage are measured and compared across different scenarios to evaluate algorithm efficiency.Rigorous experimentation using programming languages like Python or Java is conducted to implement sorting algorithms and collect performance metrics. [4]Statistical analysis techniques are employed to derive meaningful insights from the gathered data, facilitating a robust comparison and evaluation of algorithms.Validation steps are included to ensure the reliability and accuracy of findings through cross-verification and sensitivity analyses.

Author Name	Methodology	Gaps	Outcome
Miss. Pooja K. Chhatwani, Miss. Jayashree S. Somani	Comparative study based on pros and cons	Limited sorting methods considered	Quick Sort is fastest; Bubble Sort is slowest
Michael Polyntsov	External sort and DBMS implementation	Few prior works in the domain	Experimental validation using PosDB
Bülent Arslan	Evaluation of sorting on different OS	No major gap identified	Successful initialization across platforms

Table 1

4. Results

The research findings provide valuable insights into sorting algorithms' performance, particularly in the context of big data analytics. Bubble Sort exhibited longer processing times with increasing data sizes, aligning with theoretical expectations. Quick Sort and Merge Sort demonstrated consistent and efficient performance with larger datasets, showcasing their superiority.

Comparison-based analysis highlighted Bubble, Selection, and Insertion sort algorithms' suitability for smaller lists but inefficiency for larger datasets. Quick Sort emerged as the fastest algorithm, while Bubble Sort was the slowest. Specific sorting algorithms were recommended based on dataset sizes and complexity.

The study also explored modified Bubble Sort, which proved more efficient than standard Bubble Sort, and Insertion Sort's optimal performance for smaller datasets.

Parameter	Bubble Sort	Selection Sort	Insertion Sort	Merge Sort	Quick Sort
Time Complexity 1.Best Case 2.Average Case 3.Worst Case	 O(n) $O(n^2)$ $O(n^2)$	 $O(n^2)$ $O(n^2)$ $O(n^2)$	 O(n) $O(n^2)$ $O(n^2)$	 O(nlogn) O(nlogn) O(nlogn)	 O(nlogn) O(nlogn) $O(n^2)$
Space Complexity	O(1)	O(1)	O(1)	O(n)	O(1)
Method	Exchange	Selection	Insertion	Merging	Partitioning
Stable	Yes	No	Yes	Yes	Depends on Elements
In-Place	Yes	Yes	Yes	No	Yes
Type	Internal	Internal	Internal	Can be both Internal and External	Internal
Strategy	Scan all the elements & bubble up largest element	Scan all the elements & sort the list	Scan all the elements & insert the smallest element before largest	Divides An Array Into Two Separate Lists(Sub Arrays)	Concept of pivot element

Figure 1

The figure 1 illustrates a comprehensive comparison of five basic sorting algorithms: Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, and Quick Sort. Time and space complexities, method, stability, in-place property, type, and sorting strategy are used to evaluate each algorithm. In time complexity, Merge Sort and Quick Sort are usually more efficient with O(n log n) in average cases, whereas Bubble, Selection, and Insertion Sorts have $O(n^2)$ complexity. Insertion and Bubble Sort are better in best-case situations (O(n)), particularly with almost sorted data.

Space complexity is O(1) for everyone except Merge Sort, which takes O(n) extra space. The methods of sorting are exchange (Bubble), selection (Selection), insertion (Insertion), merging (Merge), and partitioning (Quick). Bubble, Insertion, and Merge Sorts are stable; Quick Sort is conditionally stable, and Selection Sort is not stable. All algorithms except Merge Sort are in-place. Approaches vary from straightforward element scans to divide-and-conquer and pivot-based partitioning schemes.

5. Complexity Analysis Table

The table 2 contains the problems faced in application of the sorting algorithm and their recommended algorithm.

Problem Definition	**Recommended Algorithm**
No extra memory needed	Insertion, Merge, Selection, Bubble Sort
Business/DB apps needing extra memory	Merge Sort
Uniform range distribution (0,1)	Bucket Sort

Problem Definition	Recommended Algorithm
Sorting multiple fields/alphabet	Radix Sort
Smaller data input	Insertion Sort
Larger data input	Merge, Quick, Heap Sort
Repeated data items	Counting Sort
Sort by address	Bucket Sort
Fast solution required	Bubble Sort, Smart Bubble Sort

Table 2

6. Conclusion and Future Work

The research study comprehensively analyzed sorting algorithms' performance and their application in operating systems. Future work could explore additional sorting algorithms, advanced variations, and optimizations to provide a more comprehensive comparison. Investigating the impact of different hardware configurations and parallel processing techniques on sorting algorithm performance would yield further insights.

Overall, the research contributes valuable insights to algorithm selection processes, aiding decision-makers in optimizing data processing methods for enhanced efficiency and performance in big data analytics applications within operating systems.

7.References

[1] Seymour Lipschutz and G A Vijayalakshmi Pai , Data Structures, (Tata McGraw Hill companies), Indian adapted edition 2006-07 West patel nagar,New Delhi-110063.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest , Introduction to Algorithms, Fifth Indian printing (Prentice Hall of India private limited), New Delhi-110001

[3] Eshan Kapur, Parveen Kumar and Sahil Gupta,"Proposal Of A Two Way Sorting Algorithm And Performance Comparison With Existing Algorithms" International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.2, No.3, June 2012.

[4]Aayush Agarwal ,Vikas Pardesi , Namita Agarwal, "A New Approach To Sorting: Min-Max Sorting Algorithm" International Journal of Engineering Research & Technology (IJERT) Vol. 2 Issue 5, May – 2013.

[5]Tarundeep Singh Sodhi, Surmeet Kaur, Snehdeep Kaur, "Enhanced Insertion Sort Algorithm ",International Journal of Computer Applications (0975 – 8887) Volume 64– No.21, February 2013.

[6] Search and Sort Algorithms for Big Data Structures Bülent Arslan

Haaga-Helia University of Applied SciencescBachelor's Degree Programme in Business Information Technology Bachelor's Thesis