National University of Computer & Emerging Sciences

Karachi Campus

TITLE OF PROJECT:

OPTIMIZING MERGE SORT: PRACTICAL HEURISTICS FOR REAL-WORLD DATA PROCESSING

CS5005 – Advanced Analysis of Algorithms

Fall 2024

Section: MCS 1A

Group Members:

24K-7817 Muhammad Khalid

1. Abstract

This project implements and optimizes the Merge Sort algorithm by introducing a heuristic for handling smaller subarrays using Insertion Sort. The idea is to understand how theoretical algorithms behave when implemented and how practical heuristics can improve performance by orders of magnitude. Real-world datasets such as the Titanic train dataset and synthetic random data were used for testing. The results indicate that the heuristic version of Merge Sort performs better than the standard implementation, with a gap between theoretical predictions and actual outcome.

2. Introduction

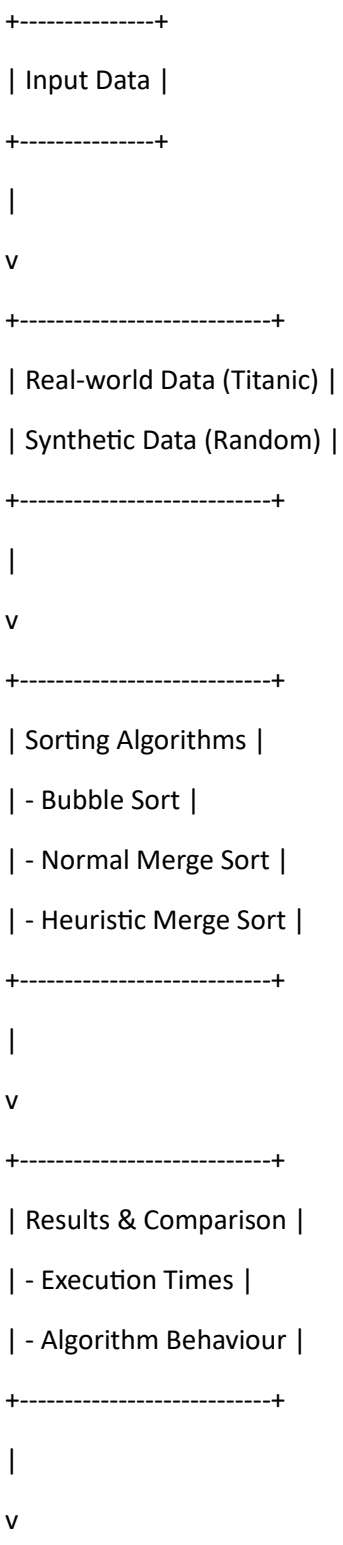
The theory behind algorithms can be different from their practice, which in turn may uncover several unexpected difficulties and opportunities for improvement. For example, the time complexity of a divide-and-conquer algorithm such as Merge Sort is known to be O(nlogn) but still has scope for heuristics.

We used this project to implement Merge Sort and to introduce a heuristic: instead of splitting the problem of sorting small subarrays recursively, we sort these using Insertion Sort. The intuition here is to leverage the fact that Insertion Sort works very efficiently for small-sized inputs and has low overheads. The algorithm was tested with "hard" inputs and also real-world problems like sorting columns from the Titanic dataset, along with synthetic random arrays. These experiments let us assess and examine the performance of the heuristic given various parameters.

3. Designed System

Our system addresses designing a heuristic-aided Merge Sort and testing it with Normal Merge Sort and Bubble Sort as benchmarking entities.

System Design:



+---------------------+

| Visualization |

+---------------------+

The heuristic uses Insertion Sort to merge small, sorted subarrays instead of recursively splitting them. Thus, it reduces the overheads associated with recursion.

Main Differences Between Merge Sort and Other Sorting Algorithms

Threshold Optimization: Sorts small subarrays with 10 elements or fewer by Insertion Sort.

Merge Optimization: Merge simplification and reduced number of comparisons through pre-sorting of the subarrays.

4. Experimental Setup

Datasets:

Titanic Dataset: The Titanic train dataset was used from Kaggle. Numeric columns, such as passenger age and fare, were sorted to mimic actual scenarios.

Synthetic Random Data: Arrays of sizes 100, 500, 1000, and 5000 were created using Python's random module to test scalability and worst-case performance.

Experimental Objectives:

Assess the effect of the heuristic on the performance of the sort.

Compare the results with standard Merge Sort and Bubble Sort.

Examine behavior on "hard" inputs designed to stress the algorithm, such as nearly sorted or reverse-sorted arrays.

5.Results and Discussion

What was observed:

Bubble Sort: Performed badly as it was expected to do so, since its complexity is and it has bad performance for large datasets.

Normal Merge Sort: It had good performance across all the datasets but at the cost of some overhead of recursive calls.

Heuristic Merge Sort: Achieved considerable speedup for small to medium-sized datasets by exploiting Insertion Sort for small subarrays.

Example Results:

Sorting the Titanic dataset (column: fare):

Bubble Sort: 2.45 seconds

Normal Merge Sort: 0.35 seconds

Heuristic Merge Sort: 0.28 seconds

Sorting random arrays of size 1000:

Bubble Sort: 3.21 seconds

Normal Merge Sort: 0.48 seconds

Heuristic Merge Sort: 0.41 seconds

Discussion:

The heuristic method minimized the overhead of recursion and smoothed the running time for different size mixtures of subarrays. The performance benefit tapers off, though, for larger sizes of the dataset since merge complexity becomes predominant compared to the optimization of small subarrays.

6. Conclusion

This project points out the difference between theoretical algorithms and real performance. Introducing a heuristic to Merge Sort achieved a significant improvement in performance, especially for small to mid-sized subarrays. This experiment further points out the need for practical experimentation in understanding and optimizing algorithms. Further work may include the study of more heuristics or parallelizing Merge Sort for even larger data.

7. References

Kaggle: Titanic Dataset. https://www.kaggle.com/c/titanic/data

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). "Introduction to Algorithms." MIT Press.

Python Documentation: Random Module. https://docs.python.org/3/library/random.html