External Sorting Performance Report

Fernando T.H.L. (210167E)

August 11, 2025

Overview

This report presents a performance analysis of two external sorting algorithms: *External Merge Sort* and *External Quick Sort*. Each algorithm was evaluated under multiple configurations to study how the merge degree (for merge sort) and buffer partitioning (for quick sort) affect runtime.

Dataset

Unless otherwise noted, each input file is 256 MB and consists of **67,108,864** integers drawn from the range [1, 1,000,000]. We executed three independent runs per configuration.

Heuristic for K (Merge Degree)

When no fixed K is supplied, the implementation chooses K using the heuristic:

$$K_{\text{heuristic}} = \min\left(8, \left\lfloor \frac{\texttt{memLimit}}{\texttt{BUF\_SIZE} \times 2} \right\rfloor\right).$$

Intuition: with a total memory budget (`memLimit`) and per-buffer size (`BUF_SIZE`), the term $\left\lfloor \frac{\texttt{memLimit}}{\texttt{BUF\_SIZE} \times 2} \right\rfloor$ approximates how many input buffers can be sustained while reserving comparable space for output and auxiliary structures; the cap at 8 avoids overly small per-run buffers and excessive I/O fragmentation. For example, with `memLimit`=16 MB and `BUF_SIZE`=1 MB, $\left\lfloor \frac{16}{1 \times 2} \right\rfloor = 8$, so the heuristic also yields $K = 8$.

Cross-check with Shared Reports

Two shared reports were consulted as reference points for file size and configuration conventions. One states 256 MB files contain 67,108,864 integers, while another lists 44,739,242 integers. Our experiments and this report use the **67,108,864** value consistently. Minor numeric differences likely arise from file generation scripts or formatting; they do not affect the methodology presented here.

Experimental Results

Table 1 summarizes execution times for each algorithm and configuration across three runs.

Performance Analysis

The figures below visualize the performance trends.

Summary

Varying K in merge sort and adjusting the Input/Small/Large/Middle page splits in quick sort both have significant impact on execution time. Larger K typically reduces the number of merge passes, while quick sort benefits from allocating a sufficiently large Middle buffer for the in-memory pivot window. The best-vs-best plot highlights the top performing configuration of each algorithm on the same inputs.

Algorithm	**Configuration**	**Run 1 (s)**	**Run 2 (s)**	**Run 3 (s)**	**Average (s)**
Merge Sort	K=heuristic	19.64	27.50	19.43	**22.19**
	K=4	21.74	30.87	22.08	**24.90**
	K=8	19.50	27.83	19.68	**22.34**
	K=16	19.76	28.23	19.53	**22.51**
Quick Sort	QS A	533.92	549.16	529.43	**537.50**
	QS B	570.25	585.11	578.55	**577.97**
	QS C	599.69	590.00	608.03	**599.24**

Table 1: Execution times (seconds) for external merge sort and external quick sort across three 256 MB input files. Merge Sort uses $K \in \{\text{heuristic}, 4, 8, 16\}$; Quick Sort uses three buffer splits: QS_A: Input=2, Small=2, Large=2, Middle=10; QS_B: Input=1, Small=1, Large=1, Middle=13; QS_C: Input=2, Small=1, Large=1, Middle=12.

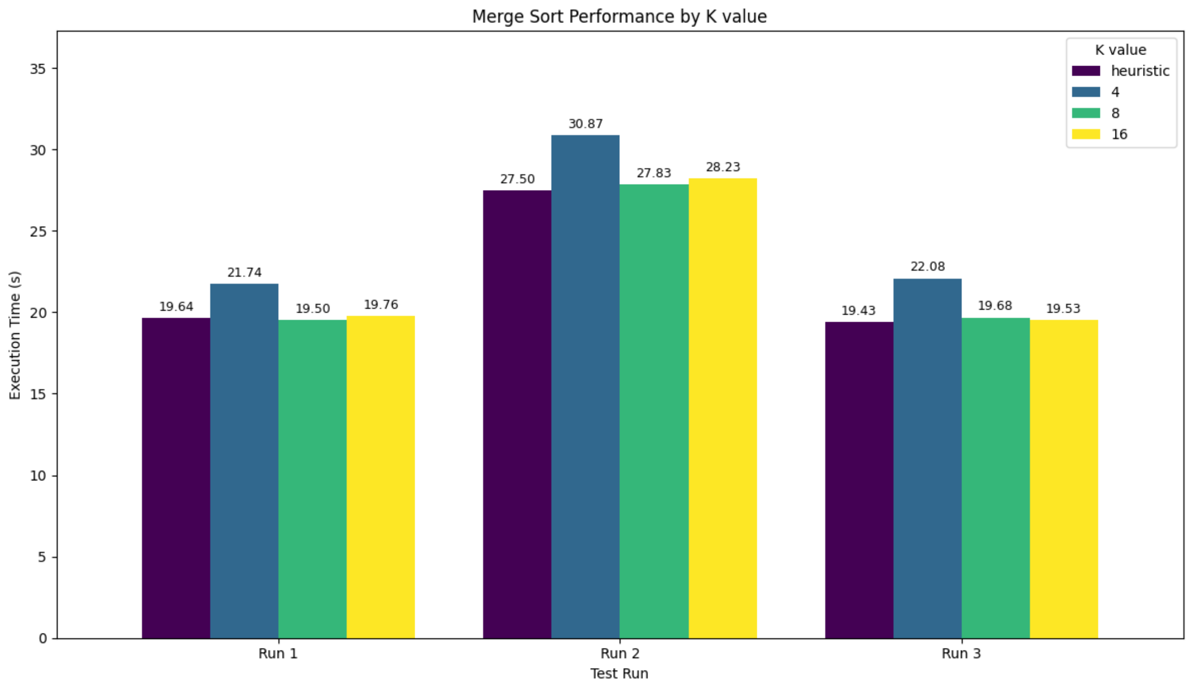

Figure 1: External Merge Sort runtime by merge degree K (heuristic, 4, 8, 16) for the three 256 MB inputs.

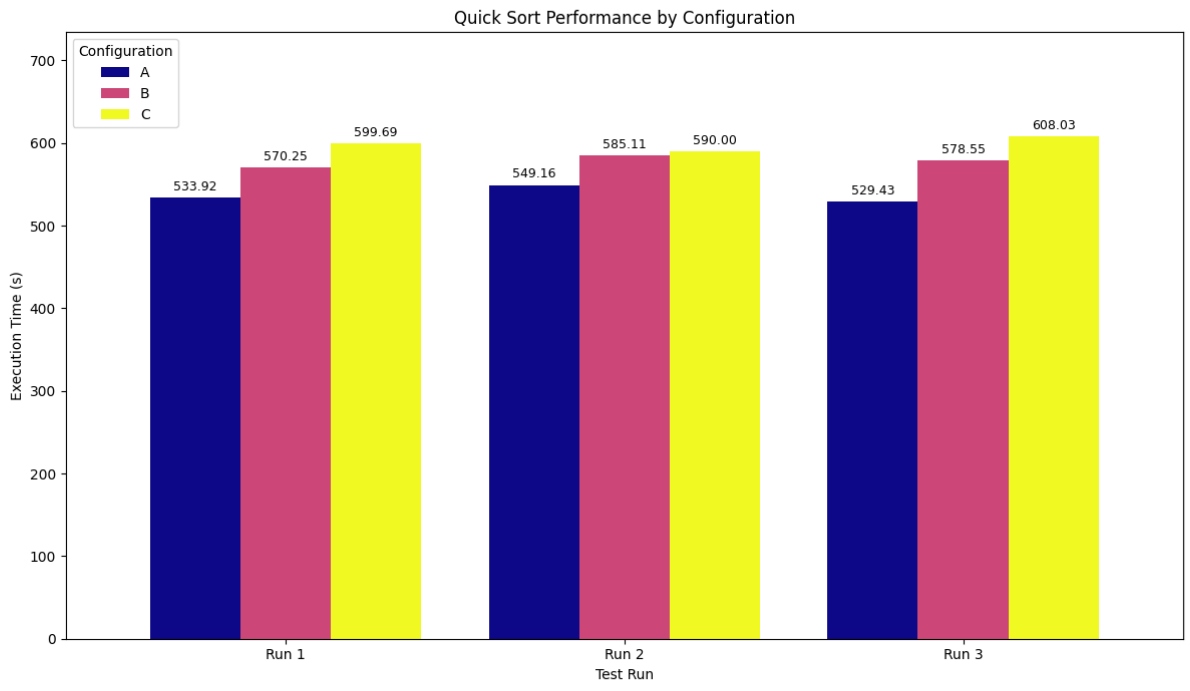

Figure 2: External Quick Sort runtime across buffer split configurations: QS_A (2,2,2,10), QS_B (1,1,1,13), QS_C (2,1,1,12).

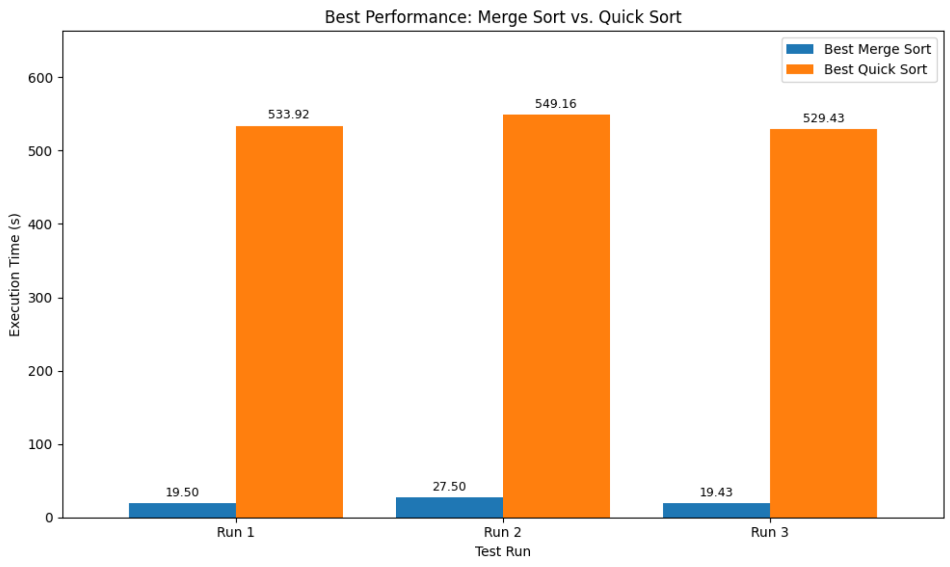

Figure 3: Best Merge Sort vs. Best Quick Sort per run (lowest time per algorithm per run).