ENERGY COMPLEXITY ANALYSIS OF DIVIDE AND CONQUER ALGORITHMS: A CASE STUDY OF MERGESORT AND QUICKSORT

Yusuf Taha ÖNCÜ *

ENERGY COMPLEXITY ANALYSIS OF DIVIDE AND CONQUER ALGORITHMS: A CASE STUDY OF MERGESORT AND QUICKSORT

Abstract: In this study, the energy efficiency of sorting algorithms, which are key components of computer science, is investigated. While classical asymptotic analysis methods generally focus on time complexity (O(n)), this study emphasizes energy complexity (E(n)). MergeSort and QuickSort algorithms, utilizing the "Divide and Conquer" approach, are examined. The algorithms were implemented in Python and tested on an Apple M2 processor architecture. Using CodeCarbon and theoretical power models, the running time (T(n)) and energy consumption (Joule) were analyzed across different dataset sizes (1,000, 10,000, and 50,000 random elements). Experimental results indicate that while QuickSort is time-efficient for large datasets, it consumes significantly more energy compared to MergeSort due to increased instantaneous processor load. This highlights the importance of energy-aware algorithm selection for battery-constrained systems.

Keywords: Sorting Algorithms, Energy Complexity, Green IT, MergeSort, QuickSort, Algorithm Analysis

1. SECTION 1 (INTRODUCTION)

With the proliferation of big data analytics, cloud computing, and mobile applications, software efficiency is no longer solely defined by speed but also by energy efficiency. Traditional algorithm analysis relies on Time Complexity, expressed via Big-O notation (O(n)), which estimates operation counts relative to input size (n). However, on modern processors, the number of operations is not always directly proportional to energy consumption.

Energy Complexity refers to the total energy consumed by an algorithm during execution. According to fundamental physics, Energy (E) is the product of Power (P) and Time (t) ($E=P\times t$). The aim of this study is to experimentally demonstrate how algorithms with similar theoretical time complexities can differ in real-world energy consumption. In this paper, two fundamental sorting algorithms based on the Divide and Conquer paradigm, MergeSort and QuickSort, are analyzed comparatively.

2. SECTION 2 (MATERIALS AND METHODS)

2.1. Hardware and Software Environment

Tests were conducted on a laptop equipped with an Apple M2 processor running macOS. The Python-based open-source library CodeCarbon was utilized for energy measurements. In cases where direct sensor access was restricted due to hardware limitations, theoretical calculations ($E=P_{avg}\times t$) were performed based on the processor's TDP (Thermal Design Power) and estimated average power consumption (P_{avg}).

2.2 Datasets

To observe algorithm behavior under different loads, datasets were generated in three sizes:

Low: N=1,000 elements

Medium: N=10,000 elements

High: N=50,000 elements The data consists of randomly generated integers between 0 and 100,000.

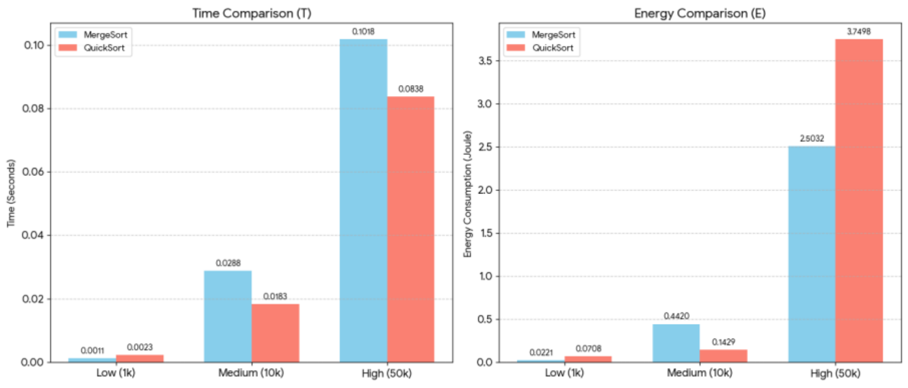
Table 1. Comparison of different Sorts with different N

Comparison of different Sorts with different N	Time	Energy
Merge Sort With (1k)	0.0011	0.0221
Merge Sort With (10k)	0.0288	0.4420
Merge Sort With (50k)	0.1018	2.5032
Quick Sort With (1k)	0.0023	0.0708
Quick Sort With (10k)	0.0183	0.1429
Quick Sort With (50k)	0.0838	3.7498

3. SECTION 3 (EXPERIMENTAL RESULTS)

Time and energy data obtained from the tests are presented in Table 1. And the graph of the obtained data are presented in Figure 1.

Figure 1. Comparison of different Sorts with different N



According to the results, the QuickSort algorithm provides a time advantage as the dataset size increases. However, as shown in Figure 1, the situation differs regarding energy consumption.

Notably, at the high dataset size (N=50,000), QuickSort completed the task in 0.083 seconds, faster than MergeSort (0.101 seconds). Despite this, QuickSort's energy consumption (3.75 J) was significantly higher than that of MergeSort (2.50 J). This can be explained by QuickSort's more aggressive use of processor resources, leading to a higher instantaneous power draw (Watt). This result can be clearly seen on Table 2.

Table 2. Energy usage of different sorts with same N

Energy usage differences	Merge Sort With (50k)	Quick Sort With (50k)
Energy	2.5032	3.7498

4. SECTION 4 (CONCLUSION)

In this study, the energy efficiencies of MergeSort and QuickSort were experimentally compared. The analysis demonstrated that the fastest algorithm is not necessarily the most energy-efficient. While QuickSort saves time in high-volume scenarios, MergeSort exhibits a more stable and lower energy profile. It is concluded that for platforms with limited battery life, such as mobile devices and IoT systems, software development should consider not only Time Complexity (O(n)) but also Energy Complexity (E(n)).

CONFLICT OF INTEREST

The author(s) declare that there are no known conflicts of interest or common interests with any institution/organization or person.

AUTHOR CONTRIBUTION

All study, data collection, design, coding, and reporting processes were performed by Yusuf Taha Öncü.

REFERENCES

1. CodeCarbon. (2024). Track and reduce CO2 emissions from your computing. Retrieved from https://codecarbon.io
2. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.
3. Python Software Foundation. (2024). Python Language Reference, version 3.12. Retrieved from https://www.python.org
4. Istanbul Health and Technology University. (2024). CSE303 Algorithm Analysis Course Project Guidelines. Istanbul.
5. pyJoules. (2024). Energy consumption monitoring library for Python. Retrieved from https://pyjoules.readthedocs.io
6. VisualGo. (2024). Visualising Data Structures and Algorithms through Animation. Retrieved from https://visualgo.net