UDC 004.421

COMPARISON OF SORTING ALGORITHMS PERFORMANCE: FUNCTIONAL VS. CLASS-BASED APPROACHES

Osipov Y.D., student, eosipopo@gmail.com, GoIT Neoversity

Introduction. In modern programming, sorting algorithms play a key role in data processing and analysis. The efficiency of sorting algorithms directly affects the performance of software systems. This work examines the comparison of the performance of classic sorting algorithms implemented through functional and class-based approaches in the Python programming language.

Research Methodology. The study examined three sorting algorithms: merge sort, insertion sort, and TimSort. Two versions of implementation were developed for each algorithm: functional and class-based. Input data for sorting were randomly generated with varying amounts of elements to assess the performance of the algorithms on different data sets.

Merge Sort. The results show that the class-based implementation of the merge sort algorithm is more efficient in terms of execution time compared to the functional implementation, especially on large data volumes. For example, at a data size of 6400 elements, the difference in performance reaches 39.57%, and at 12800 elements — 9.58%.

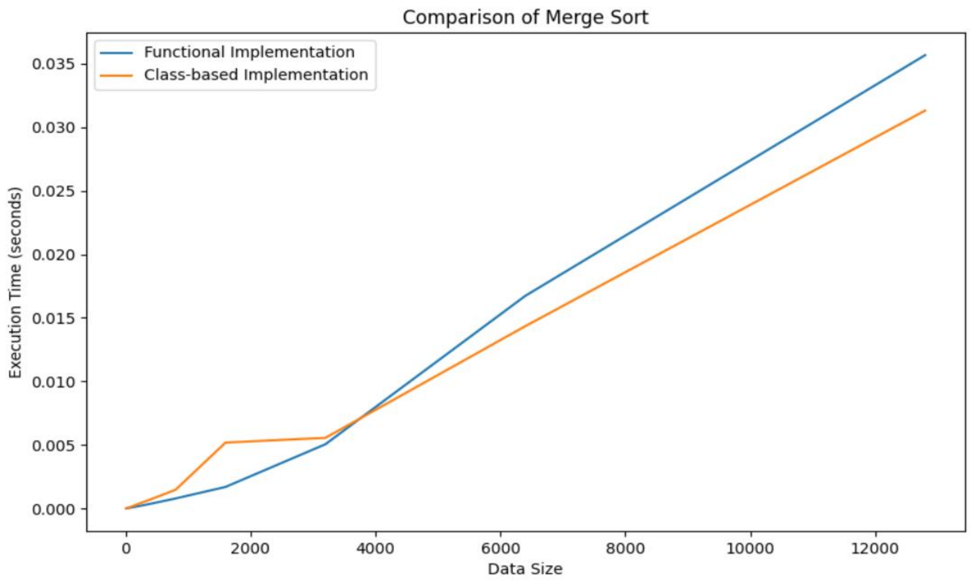

Figure 1 – Merge sort performance[2].

Insertion Sort. A similar trend is observed for insertion sort: the class-based implementation shows better performance compared to the functional one. The difference in performance becomes particularly noticeable on data volumes of 1600 elements and more, where the speed of the class-based implementation exceeds the functional one by more than 89.76%.

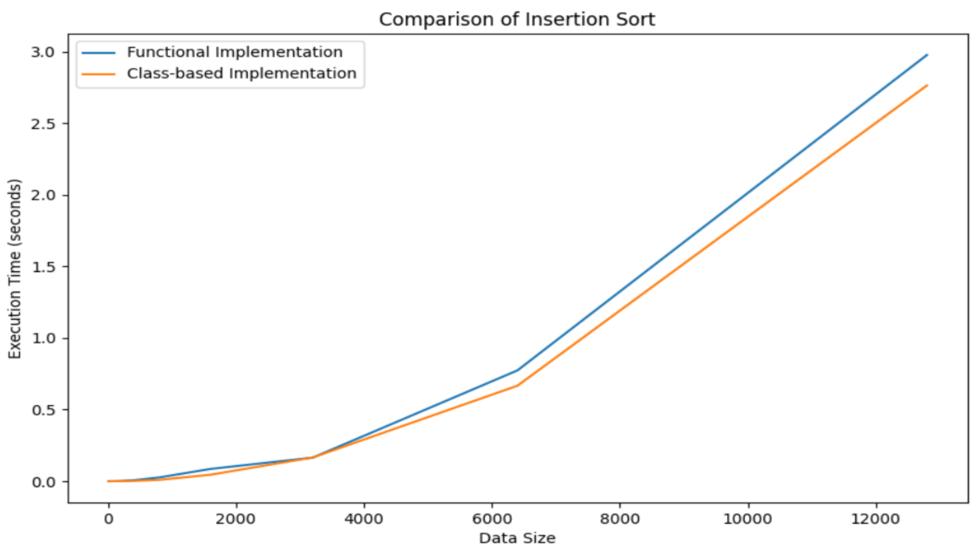

Figure 2 – Insertion sort performance[2].

TimSort. TimSort, implemented in a class-based approach, demonstrates a significant advantage over the functional implementation across the entire spectrum of data sizes. The largest difference in performance is observed on small data volumes (up to 800 elements), where the class-based implementation can be more than 20 times faster than the functional one. This highlights the efficiency of the class-based approach for optimizing the execution of sorting algorithms in Python.

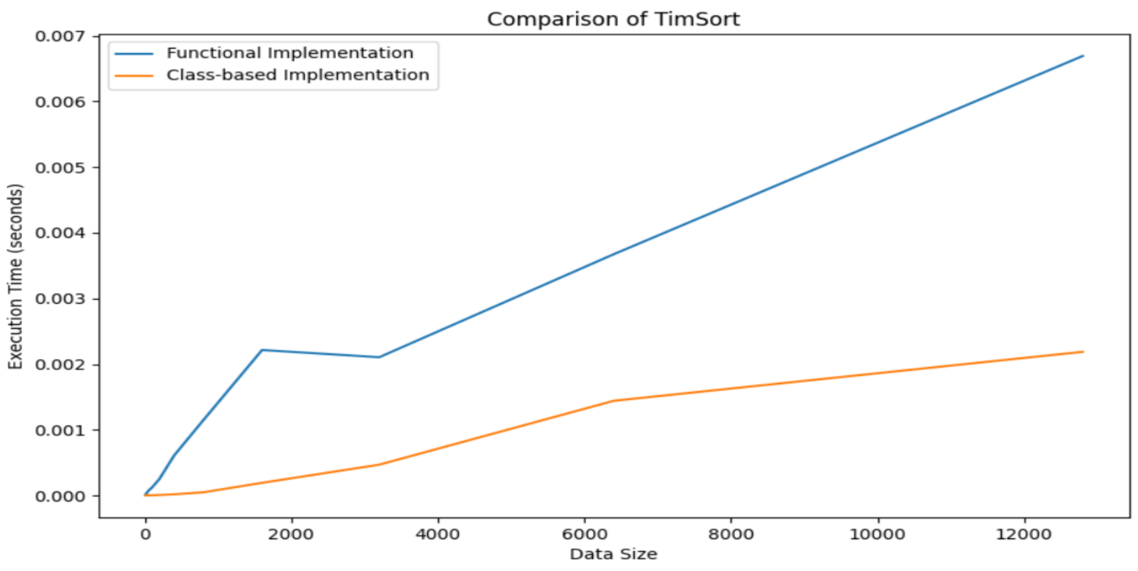

Figure 3 – TimSort performance[2].

Conclusions. The analysis of the results shows that the class-based implementation of sorting algorithms provides better performance compared to the functional implementation, especially on large data volumes. This may be due to more

efficient memory management and optimizations that Python applies to classes and methods during execution. As noted by Gamma et al. [3], the use of object-oriented approaches and design patterns can significantly enhance software development efficiency, simplify the management of complex systems, and improve program execution performance. This supports our observations about the advantages of the class-based implementation for sorting algorithms.
The results confirm the hypothesis that the choice of implementation approach for sorting algorithms affects their performance. Thus, when developing high-performance applications that require efficient data sorting, it is recommended to prefer the class-based implementation of algorithms.

References

1. Python Software Foundation. Python Language Reference, version 3.8 [Internet]; 2024; Available from: https://www.python.org.
2. Osipov Y. Comparison of sorting algorithms performance: functional vs. class-based approaches [Internet]; GitHub; 2024; Available from: https://github.com/DeadNord/class-func-algo-performance.
3. Gamma E, Helm R, Johnson R, Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional; 1994: 395.

UDC 004.4

DEVELOPMENT OF A MOBILE APPLICATION FOR MATHEMATICS LEARNING

Diana Paievska, student, dianapaievska111@gmail.com, State University of Trade and Economics
Vitaliy Bazurin, Candidate of Pedagogical Sciences, Assoc. Prof, v.bazurin@knute.edu.ua, State University of Trade and Economics
Yurii Yurchenko, Assistant, y.yurchenko@knute.edu.ua, State University of Trade and Economics

Over the past four years, the educational process has undergone significant digitization and has become a reality of our present day. This rapid spread is driven by a variety of factors: the rapid advancement of digital technologies; the shift to distance learning, initially triggered by the Covid-19 pandemic, and later by the events of full-scale Russian military invasion of Ukraine; the emergence of a new generation of learners in general secondary and higher education, who better perceive bright visualized information, as well as the practical functions of the educational process aimed at developing key competencies. However, the modern world is characterized by rapid development of innovative technologies, which are becoming increasingly