International Journal of Education, Information Technology and Others (IJEIT), Agustus 2025, 8(3.B), 148-157
DOI: https://jurnal.peneliti.net/index.php/IJEIT/article/view/12718
p-ISSN: 2654-2528 e-ISSN: 2623-2324
Accredited by Directorate General of Strengthening for Research and Development
Available online at https://jurnal.peneliti.net/index.php/IJEIT

Comparative Analysis of Searching and Sorting Algorithms in Student Data Processing

M. Sulpin Agung Saputra1, M. Raihan Meidiansyah2, Arya Damara Sanputra3, Jaka Wardana4

[1,2,3,4]Faculty of Computer Science and Science, Indo Global Mandiri University

Received: 07 July 2025
Revised: 16 July 2025
Accepted: 23 July 2025

Abstract

Efficient data processing is crucial, especially in searching and sorting large volumes of student data. This study aims to analyze and compare the performance of four algorithms: Linear Search, Binary Search, Bubble Sort, and Quick Sort, in student data processing. The experiments were conducted using datasets of 1,000, 5,000, and 10,000 student records. The results show that Quick Sort delivers the best performance in sorting tasks with the fastest execution time, while Binary Search demonstrates high efficiency in searching within sorted datasets. In contrast, Bubble Sort and Linear Search exhibit poor performance as the dataset size increases. This study recommends the use of Quick Sort for sorting and Binary Search for searching in large-scale data processing.

Keywords: *Linear Search, Binary Search, Bubble Sort, Quick Sort, Student Data Processing, Searching Algorithms, Sorting Algorithms.*

(*) Corresponding Author:

How to Cite: Saputra, M. S., Meidiansyah, M., Sanputra, A., & Wardana, J. (2025). Comparative Analysis of Searching and Sorting Algorithms in Student Data Processing. *International Journal of Education, Information Technology, and Others*, *8*(3.B), 148-157. Retrieved from https://jurnal.peneliti.net/index.php/IJEIT/article/view/12718

INTRODUCTION

Data processing in academia, especially in the field of education, plays a very important role. Student data, which includes information such as identity, grades, and academic achievement, often has to be processed quickly and efficiently for a variety of needs ranging from data search to information sorting. (Muhammad Farhan Abdullah, Isna Hafiza, Rama Wahyuni, Adrian Syahputra, 2023)(Nasution, 2023)

In student data processing, there are various algorithms that can be used to perform these tasks, each with different advantages and disadvantages. Search algorithms such as Linear Search and Binary Search have different methods of searching for specific elements in a dataset. Likewise, sorting algorithms, such as Bubble Sort and Quick Sort, have significant differences in terms of execution time and algorithm complexity. The selection of the right algorithm is very important so that the data processing process can be carried out efficiently, especially when dealing with large datasets that are often found in student data management in educational institutions. (Yanti, F., & Eriana, E. S., 2024) (Audy, 2015) (Muhammad Farhan Abdullah, Isna Hafiza, Rama Wahyuni, Adrian Syahputra, 2023)

Based on this phenomenon, it is important to conduct research that compares these algorithms in terms of performance and efficiency. Thus, this study aims to conduct a comparative analysis between search and sorting algorithms in

student data processing to identify the most efficient algorithms in handling large datasets. (Mahrozi, 2023)(Mahrozi, 2023) (Fitri Yanti, S.Kom., M.Kom. Emi Sita Eriana, S.Kom., M.Kom., 2024)

Problem Formulation

Along with the importance of efficient student data processing, several questions arise that need to be answered in this study:

1. How do the Linear Search and Binary Search algorithms perform in the search for student data?
2. How do the Bubble Sort and Quick Sort algorithms perform in sorting student data?
3. Which algorithm is the most efficient in terms of execution time and resource utilization in student data processing?

Research Objectives

The main objective of this study is to compare the performance of search and sorting algorithms in student data processing, focusing on:

1. Analyze the execution time of the Linear Search, Binary Search, Bubble Sort, and Quick Sort algorithms.
2. Assess the advantages and disadvantages of each algorithm in student data processing.
3. Provide recommendations for the best algorithms to use in the processing of large student data.

Literature Review

Several previous studies have discussed search algorithms and data sequencing. For example, a study by Cormen et al. (2009) that tested the efficiency of various search algorithms in large datasets, as well as a study by Sedgewick & Wayne (2011) that discussed the comparison of sorting algorithms based on execution time and complexity. In addition, basic theories regarding the Linear Search, Binary Search, Bubble Sort, and Quick Sort algorithms can be found in textbooks such as by Cormen et al. (2009), which explain in detail the working principles and uses of each of these algorithms.

RESEARCH METHODS

1. Research Design

This study uses an experimental approach with the aim of comparing the performance of search and sorting algorithms in student data processing. This research was carried out by implementing and testing four main algorithms: Linear Search, Binary Search, Bubble Sort, and Quick Sort, on a dataset containing student data consisting of name, student identification number, and major. Each algorithm will be tested on datasets of different sizes to observe their performance in terms of execution time and efficiency. (Retnoningsih, 2018)

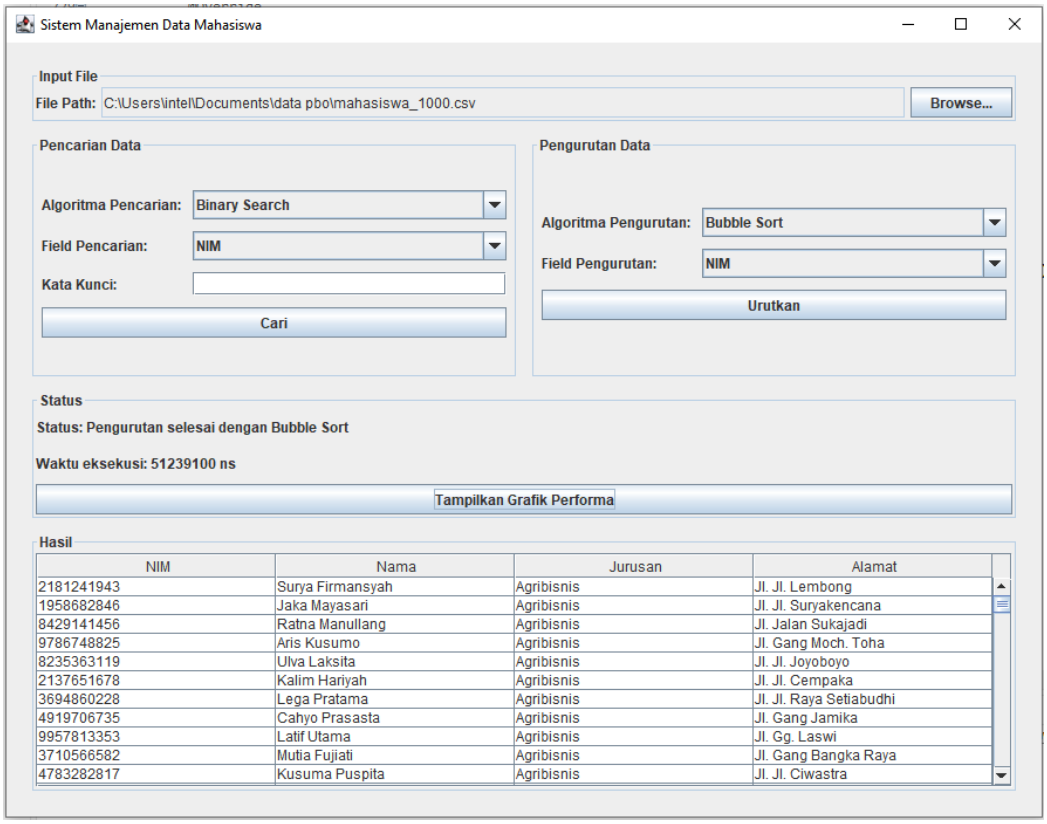
2. Data Collection

The data used in this study is in the form of a student dataset that includes several important information, such as student names, student identification numbers (NIM), majors, and addresses. This dataset is generated from sample data created with the Python programming language. The data taken consisted of various dataset sizes, namely 1,000, 5,000 and 10,000 student data to test the algorithm's efficiency on a large scale.

3. GUI Design

The Graphical User Interface (GUI) design used in this system aims to make it easier for users to search and sort student data. The user interface will have several main components:

Figure 1. GUI Design of Student Data Searching and Sorting Algorithm



1. Form Input:

- Users can enter files in CSV/SQL format that contain student data through the File Input field. The file can be selected by clicking the Browse button and selecting the location of the file on the system.
- After selecting the file, users can fill in the search criteria in the Data Search Section.
- Search Algorithm: Users can choose the desired search algorithm (e.g., Binary Search).
- Search Field: Users can select the data fields to search for (e.g., NIM).
- Keywords: Users can enter the appropriate search value, such as the NIM they want to search.
- Users can then click on the Search button to start the data search process as per the specified criteria.

2. Display of Results:

- After the search or sorting is complete, the results will be displayed in the form of a table containing the student's NIM, Name, Major, and Address.

- The data that appears is the result of a search or sorting according to the algorithm that has been chosen by the user.

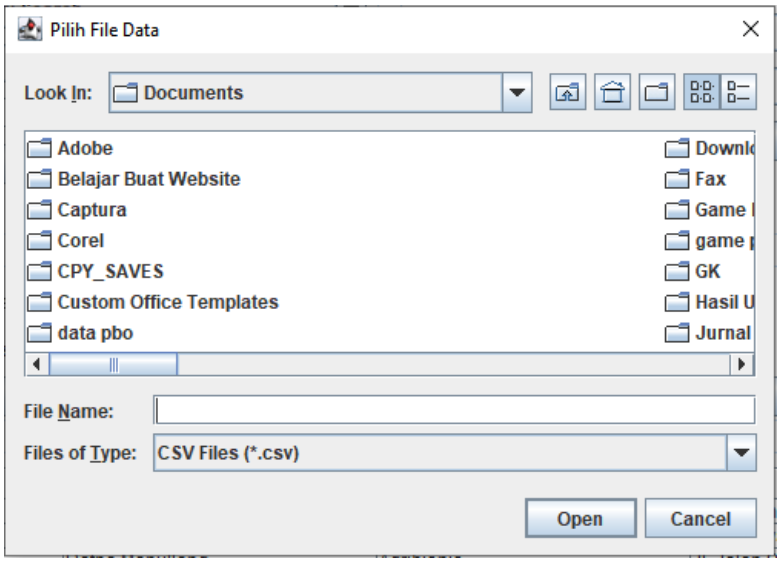
3. Performance Graph View:

- Users can choose to display a performance graph of the algorithm used by clicking on the Show Performance Graph button.
- This graph will show how the search or sorting algorithm performs in terms of execution time.

With this GUI, users can easily search and sort student data as needed using various search and sorting algorithms.

The system is also designed to make it easier for users to interact with applications, including the file selection process. One of the components used is the dialog window to open the file (*Open File Dialog*), as shown in Figure 2.

Figure 2. Design GUI (file data system) elements

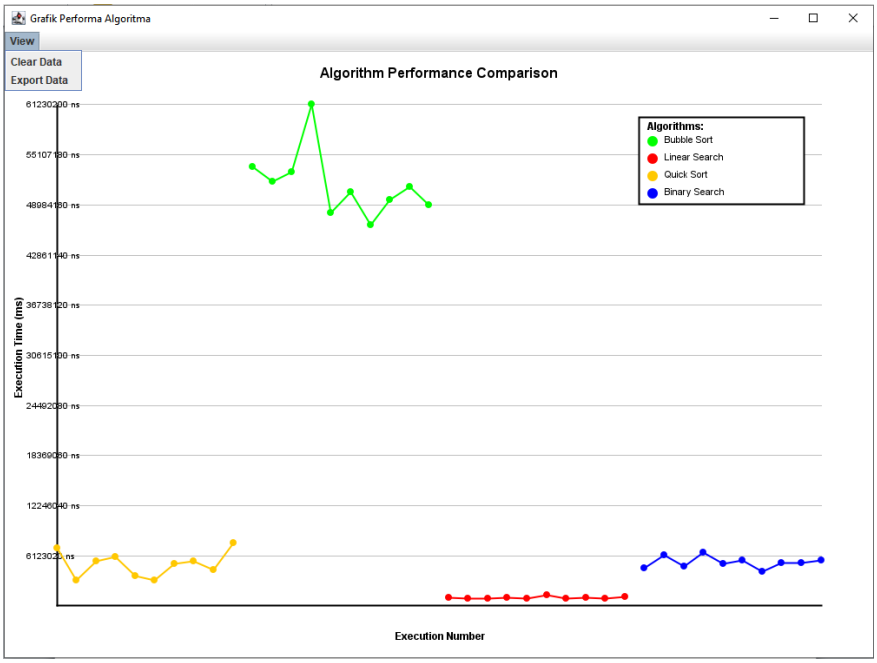


The image above shows the GUI elements used to allow users to navigate the *file system* and select the files they want to open. This component is part of the Java Swing standard library with the JFileChooser class. Some of the important features contained in this window include:

1. "Look In" dropdown: to select the current active directory.
2. Folder and file list view: shows the contents of the selected directory.
3. "File Name" field: where the user manually types the file name (optional).
4. "Files of Type" dropdown: allows users to filter the type of files displayed (in the image set to "All Files").
5. "Open" and "Cancel" buttons: to execute the selection of files or cancel the process.

The design of this GUI takes into account the usability aspect, with a familiar interface approach as in the Windows operating system. The goal is to minimize the user learning curve and speed up the interaction process, according to *the principle of user-centered design*.

Figure 3. Design GUI elements (Algorithm Performance Graph)



1. Algorithm Performance Graph:

- This graph shows a comparison of the performance of multiple algorithms based on the execution time for each execution performed.
- Y-axis (Execution Time): Displays the algorithm's execution time in nanoseconds (ns).
- X-axis (Execution Number): Displays the number of executions performed on each algorithm experiment.

2. Algorithm Display:

In this graph, there are three algorithms that are compared, namely:

- Bubble Sort (green line).
- Linear Search (yellow line).
- Quick Sort (garis merah).
- Binary Search (blue line).

Each algorithm has a different line color to make it easier to visualize the performance of each algorithm in various experiments.

3. GUI Features:

- View: Enables the display of algorithm performance graphs.
- Clear Data: Deletes the data present in the chart, so that the chart will be empty for the next attempt.
- Export Data: Saves the data that has been obtained from the execution of the algorithm into an external file.

4. Chart Interpretation:

- Each point on the graph shows the execution results of an algorithm in an experiment. For example, the dot on the Bubble Sort (green line) indicates the execution time on a given experiment.
- With this comparison, users can immediately see which algorithm has the fastest or slowest execution time, as well as analyze the performance of each algorithm more clearly.

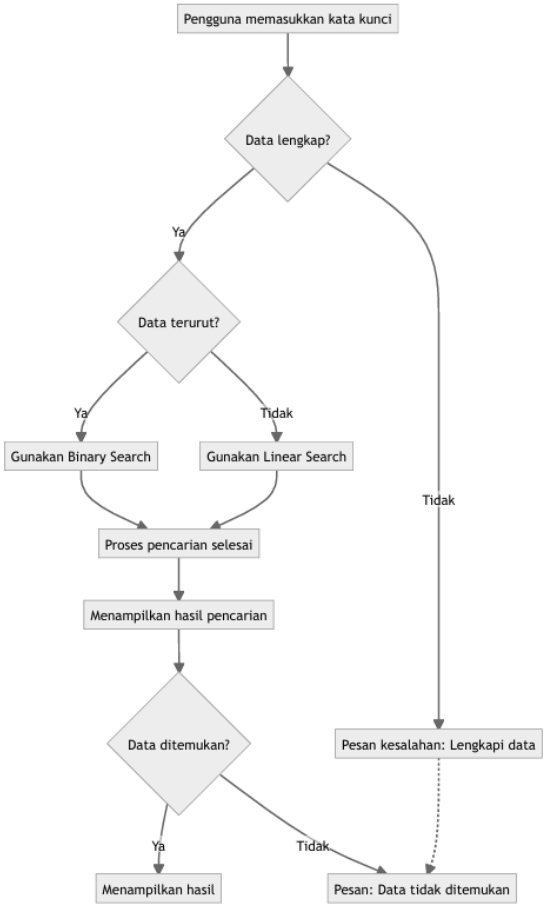
With these GUI elements, users can analyze the performance of various search and sorting algorithms, compare execution times directly, and export calculated data for further analysis.

4. Flowchart Sistem

The following flowchart illustrates the flow of the process that is run in the search algorithm and the student data sorting algorithm:

1. Searching Algorithm Process:

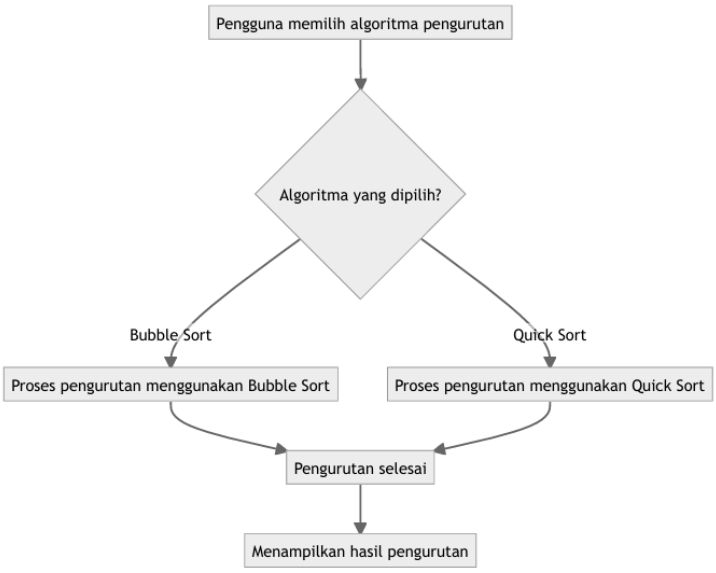
Figure 3. Flowchart Searching Algorithm



a. The user enters the keyword on the input form.
b. The system checks whether the data entered is complete or not.
c. If the data is complete, the system uses Binary Search (if the data is already sorted) or Linear Search (if the data is not sorted).
d. Search results are displayed in the results area.

2. Proses Algorithm Sorting:

Gambar 4. Flowchart Algoritma Sorting



a. The user chooses a sorting algorithm (Bubble Sort or Quick Sort).
b. The system sorts the data based on the selected algorithm.
c. The sorting results are displayed in the results area.

5. Tools and Materials

1. Programming Language: Java.
2. Hardware: A computer or laptop with amd Ryzen 3 specifications and 8GB of RAM.
3. Software: Java IDE (Eclipse).

6. Testing Method

The test was carried out to evaluate the performance of the *search* and *sorting* algorithms using three different dataset sizes, namely 1000 data, 5000 data, and 10000 data. The use of these three measures aims to observe the scalability and time efficiency of each algorithm.

1. Searching Algorithm Testing:
a. Linear Search: This algorithm is tested with various dataset sizes to measure execution time on unordered datasets.
b. Binary Search: This algorithm is tested on a sequenced dataset, to measure the efficiency of data search with a division approach.
2. Testing Sorting Algorithm:
a. Bubble Sort: Testing is performed on an unsorted dataset to measure the performance of this algorithm.
b. Quick Sort: Tested on a larger dataset to demonstrate its superiority in terms of execution time compared to Bubble Sort.

RESULTS AND DISCUSSION

1. Test Results

The test was carried out on four algorithms, namely Bubble Sort, Quick Sort, Linear Search, and Binary Search using three different datasets

measuring 1,000, 5,000, and 10,000 student data. Execution time is measured in nanoseconds (ns). The following table summarizes the test results:

Table 1. Table of test results 1000 Data

1000	Linear Search	Binary Search	Bubble Sort	Quick Sort
NIM	636520 ns	778540 ns	35116800 ns	770560 ns
Nama	874170 ns	918080 ns	43554380 ns	1175350 ns
Jurusan	1002130 ns	5185700 ns	51457460 ns	5119580 ns
Alamat	954060 ns	2057800 ns	68424200 ns	487560 ns

Table 2. 5000 Data test results table

5000	Linear Search	Binary Search	Bubble Sort	Quick Sort
BEFORE	3575430	7151400	911978110	4599760
Name	2407940	9083110	1074896780	4211120
Department	1279590	108637620	1178283500	109969140
Address	2562770	11977020	9056180	1504746130

Table 3. Table of test results 10000 Data

10000	Linear Search	Bubble Sort	Binary Search	Quick Sort
BEFORE	2248160 ns	4828175730 so-called	13933680 ns	9488960 ns
Name	3958040 ns	3690213320 so-called	11961600 ns	8998420 ns
Department	3194240 ns	4663592720 so-called	289422080 ns	287286950 ns
Address	3098360 ns	5460535130 ns	27544120 ns	21109470 ns

2. Discussion

a) Algoritma Sorting

Bubble Sort showed the worst performance of all the algorithms tested. This is in accordance with the characteristics of this algorithm which has $O(n^2)$ complexity.

- The execution time is enormous even on 1,000 data (about 35 million to 51 million ns).
- At 10,000 data, the execution time reached more than 500 million ns, proving the inefficiency of this algorithm at scale.

Quick Sort On the other hand, Quick Sort delivers the best results in sequencing, with much lower execution times than Bubble Sort.

- At 1,000 data, Quick Sort only requires about 500 thousand to 1.1 million ns.

- At 10,000 data, despite the increased time (up to 28 million ns), Quick Sort remains the fastest sorting algorithm, according to the average complexity of O(n log n).

b) Algoritma Searching

Linear Search shows the most execution time among search algorithms for large datasets. This is because this method performs searches one by one from start to finish.

- At 1,000 data, the time is still relatively small (about 600 thousand to 900 thousand ns).
- However, at 10,000 data, the time increases dramatically (about 2.2 million to 3.9 million ns).
- This shows that the O(n) complexity of this algorithm has a great effect as the data increases.

Binary Search is much more efficient than Linear Search, as it only requires $\log_2(n)$ steps on the sequenced data.

- Although the difference in small data is not very noticeable, at 10,000 data Binary Search performance is still better than Linear Search, although it increases (from 0.7 million in 5,000 data to about 13 million to 27 million ns).
- This increase in time is also affected by the process of sorting the data before the search, because Binary Search can only be done on the sorted data.

CONCLUSION

Based on the results of testing four algorithms, namely Linear Search, Binary Search, Bubble Sort, and Quick Sort, on student datasets of 1,000, 5,000, and 10,000 data, it can be concluded that the performance of each algorithm is greatly influenced by the amount of data processed. Linear Search, while simple and easy to implement, shows a linearly increased execution time as the amount of data increases. This is due to the characteristics of the algorithm that searches sequentially from the beginning to the end of the dataset, making it less efficient to use in large-scale data processing.

Binary Search, on the other hand, provides better results than Linear Search in terms of search time efficiency. However, this algorithm can only be applied to data that has already been sequenced. However, Binary Search is able to maintain a relatively low execution time even when the amount of data increases, demonstrating its high efficiency under appropriate data conditions.

On the data sorting side, Bubble Sort shows very poor performance as the dataset size increases. Its execution time increases exponentially and becomes very inefficient for large datasets. This is in line with the time complexity of the Bubble Sort algorithm which is quadratic. On the contrary, Quick Sort came out as the best sorting algorithm in this test. Quick Sort's execution time remains low despite the significant increase in the amount of data, demonstrating its consistency of performance and efficiency in handling data sequencing at scale.

Overall, the results of these tests show that the selection of algorithms greatly determines the efficiency of the search and data sequencing process. Quick Sort proved to be the most efficient sorting algorithm among the algorithms tested, whereas Binary Search provided the best search performance for the sorted data. Linear Search and Bubble Sort, although they can be used on a small scale, are not

recommended for large amounts of data processing due to their inefficient execution times. In addition, the types of attributes tested (such as NIM, Name, Department, and Address) did not provide significant time differences, so performance was more influenced by the type of algorithm and data size used.

REFERENCES

Audy. (2015). Comparison of Quicksort and Bucket Sort Algorithms in Integer Data Sequencing. *ULTIMATICS*, Vol. VII, No. 1.

Ferrosi Pratama, Dwiky Juniardi, Muhammad Fauzan Arif, Suharsono. (2024). IMPLEMENTATION OF SELECTION SORT ALGORITHM FOR STUDENT DATA SEQUENCING IN THE INFORMATICS ENGINEERING STUDY PROGRAM OF PONTIANAK STATE POLYTECHNIC. *Proceedings of the National Seminar on Science and Technology Series 02* .

Fitri Yanti, S.Kom., M.Kom. Emi Sita Eriana, S.Kom., M.Kom. (2024). *SORTING AND SEARCHING ALGORITHMS.* Central Java: EUREKA MEDIA AKSARA.

Mahrozi, N. & (2023). Comparative analysis of the speed of selection sort and bubble sort algorithms. *Scientica: Scientific Journal of Science and Technology, 1(2)*, 89-98.

Muhammad Farhan Abdullah, Isna Hafiza, Rama Wahyuni, Adrian Syahputra. (2023). The Use of the Bubble Sort Algorithm in Sequencing Student Identification Numbers. *Journal of Computer Science*, 19.

Nasution, R. S. (2023). Informatics Students' Perception of the Effectiveness of the Bubble Sort Algorithm in Sorting Data. *Engineering Journal of Science, 1(3),* 220-225.

Nilal Haming, Sri Lestanti, Saiful Nur Budiman. (2022). OUTGOING MAIL MANAGEMENT APPLICATION USING SEQUENTIAL SEARCH AND SELECTION SORT AT THE BLITAR CITY KPU. *JATI (Journal of Informatics Engineering Students),* 25.

Retnoningsih, E. (2018). Data Sorting Algorithm with Insertion Sort and Selection Sort Methods. *INFORMATION MANAGEMENT FOR EDUCATORS AND PROFESSIONALS* .

Yanti, F., & E. S. (2024). *Sorting and Searching Algorithms.* South Tangerang: Eureka Media Aksara.