TEPIAN Vol. 5 No. 2 (June 2024) 79-86 p-ISSN 2721-5350 e-ISSN 2721-5369

https://doi.org/10.51967/tepian.v5i2.3055 © 2024 Tepian Politeknik Pertanian Negeri Samarinda

Application of Bubble Sort Optimization in New Student Admission Selection Using Brute Force Algorithm

Arbansyah
Informatics Engineering, Universitas Muhammadiyah Kalimantan Timur, Samarinda, 75124
arb381@umkt.ac.id

Muhammad Fauzan Nur Ilham *
Informatics Engineering, Universitas Muhammadiyah Kalimantan Timur, Samarinda, 75124
2111102441149@umkt.ac.id
**Corresponding author*

Sayekti Harits Suryawan
Informatics Engineering, Universitas Muhammadiyah Kalimantan Timur, Samarinda, 75124
shs500@umkt.ac.id

Pandu Wirayuda
Informatics Engineering, Universitas Muhammadiyah Kalimantan Timur, Samarinda, 75124
2111102441099@umkt.ac.id

Submitted: 2024-05-29; Accepted: 2024-06-09; Published: 2024-06-10

***Abstract*—**This study investigates the application of a Brute Force algorithm optimized with Bubble Sort for new student admission selection. The Brute Force algorithm, while guaranteeing accurate results, suffers from exponential time complexity with increasing data size, posing a challenge for large applicant pools. To address this limitation, this research integrates Bubble Sort optimization to reduce the execution time complexity of the Brute Force algorithm. This study goes beyond solving the student admission selection problem; it explores optimizing the Brute Force algorithm by leveraging the simplicity and efficiency of Bubble Sort. This approach aims to determine the extent to which the Brute Force algorithm can be optimized for student selection, particularly regarding execution time complexity. The integration of Bubble Sort is hypothesized to significantly improve the performance of the Brute Force algorithm by reordering data before processing, thereby minimizing unnecessary comparisons. This paper presents a comparative analysis of execution times between the traditional Brute Force approach and the optimized version. Preliminary results indicate a substantial improvement in efficiency, suggesting that this hybrid approach could be a valuable solution for similar combinatorial problems with time complexity constraints. Further research could explore the applicability of this optimized algorithm in other domains where time complexity is a critical factor.

***Keywords*—**Brute Force Algorithm, Bubble Sort, New Student Admission, Algorithm Optimization, Efficiency

I. Introduction

The selection process for new student admissions is a crucial aspect of maintaining the quality and sustainability of educational institutions. This process involves meticulously evaluating a diverse range of criteria to ensure that the most suitable candidates are admitted. In this context, the Brute Force algorithm emerges as a comprehensive approach that considers all possible solutions. However, despite its accuracy, the increasing time complexity of the Brute Force algorithm as the problem size grows poses a significant challenge (Febria Widodo et al., 2022).

Previous research has explored the Greedy approach utilizing the Knapsack algorithm for new student admissions selection (Fauzan et al., 2023). While efficient in certain scenarios, the Greedy approach often falls short in providing the optimal solution when faced with complex, multifaceted criteria. This study focuses on enhancing the Brute Force algorithm by integrating Bubble Sort optimization as an alternative solution. Bubble Sort, renowned for its simplicity, can significantly improve the initial arrangement of data, thereby facilitating a more efficient Brute Force computation (Chen, 2022; Thomas, 2023).

The Brute Force algorithm with Bubble Sort optimization will be thoroughly examined in handling the problem of new student admissions selection. In educational contexts, accurate and efficient selection processes are essential for maintaining academic standards and institutional reputation. By integrating Bubble Sort optimization, we aim to reduce the execution time complexity of the Brute Force algorithm. This integration is expected to streamline the selection process, allowing for the handling of larger datasets more effectively (Vierdansyah et al., 2023).

By emphasizing the Brute Force algorithm with the application of Bubble Sort, this study aims to explore the optimization potential in addressing the problem of new student admissions selection. Previous studies have highlighted the effectiveness of hybrid algorithms in improving computational efficiency (Afirin & Lestanti, 2021). This study not only focuses on solving the problem of new student admission selection but also delves into optimizing the Brute Force algorithm by leveraging the simple nature and optimization of Bubble Sort. An indepth understanding of this solution is expected to contribute to the development of more efficient and accurate selection methods. For instance, studies by Rizvi et al. (2024) and Balogun et al. (2023) have demonstrated the performance enhancement of integrating simple sorting algorithms with more complex ones in various applications, including data mining and resource allocation. Similarly, this study seeks to extend these findings to the domain of educational admissions.

Furthermore, the optimization of algorithms like Brute Force through methods such as Bubble Sort is a growing area of interest in computer science, particularly in the context of big data and combinatorial problems (Sunandar, 2021). As data sizes increase and problems become more complex, the need for efficient algorithms becomes more critical (Adadi, 2021). This study addresses this need by proposing a novel approach to optimize a well-known algorithm.

In addition to improving the efficiency of the Brute Force algorithm, this research also aims to provide practical insights for educational institutions (Soleh, 2023). Efficiently processing large volumes of application data can save time and resources, allowing institutions to focus on other important aspects of the admission process (Hariski et al., 2023). By reducing the computational burden, schools and universities can implement more comprehensive and fair selection criteria without compromising on processing time (Kamalov et al., 2023).

In conclusion, this study contributes to the field of algorithm optimization by integrating Bubble Sort with the Brute Force algorithm in the context of new student admission selection. By addressing the high time complexity typically associated with Brute Force, this research aims to develop a more efficient approach to solve admission selection problems. The findings are expected to have broad implications not only for educational institutions but also for other fields where similar combinatorial problems are prevalent.

II. Literature Review

A. *Algorithms*

Algorithms are structured sequences of instructions designed to solve specific problems. Within the realm of big data processing, algorithms are essential tools for improving the efficiency and accuracy of data handling. They enable the organization, management, and analysis of vast quantities of data in a timely and effective manner. The choice of appropriate algorithms is critical in overcoming challenges inherent in big data processing, such as time constraints and resource utilization (M. Suraya, 2024).

In the context of this research, we aim to optimize the bubble sort algorithm through a brute force approach for the selection of new student admissions. Thus, a comprehensive understanding of various programming algorithms and their significance in big data processing is crucial for enhancing the efficiency of the new student admissions selection process through brute force optimization.

B. *Brute Force Algorithm*

The Brute Force Algorithm stands as a foundational method in the arsenal of problem-solving techniques, characterized by its straightforward and uncomplicated nature. This approach hinges on the problem statement itself and the fundamental concepts involved, making it accessible and easy to understand. When applied to string matching tasks, the Brute Force Algorithm serves as a robust tool for comparing a given pattern against a text, employing a direct and systematic approach (Siddiq Sumi, A et al., 2018).

In practical terms, the Brute Force Algorithm operates akin to its namesake, by exhaustively iterating through all possible combinations or solutions without employing any specific optimizations or shortcuts. In the realm of string matching, this algorithm methodically compares each character of the pattern against the corresponding characters in the text, moving forward character by character until a match is found or the entire text is traversed.

For implementation, consider its utilization within a search engine environment, leveraging the PHP programming language. Despite its simplicity, the Brute Force Algorithm proves its effectiveness in solving string matching challenges. This effectiveness is derived from its systematic and exhaustive nature, leaving no stone unturned in the pursuit of finding matching patterns within a given text.

While the Brute Force Algorithm may lack the sophistication of more complex algorithms, its simplicity holds inherent advantages. Its direct approach and reliance on fundamental principles make it easily adaptable and comprehensible, rendering it a valuable tool in scenarios where efficiency may not be the primary concern, but accuracy and reliability are paramount. Thus, the Brute Force Algorithm stands as a testament to the power of simplicity in problem-solving methodologies.

C. *New Student Admission Selection*

The process of selecting new student admissions is pivotal in ensuring the excellence of an educational institution's academic standards. As highlighted in the research conducted by Ahsanul Khair Asdar at STABN Sriwijaya Tangerang Banten in 2018 (Asdar, A. K. 2018), the identification of anomalous scores in written exams, particularly in fundamental Buddhist religious materials, emerges as a critical aspect. This emphasis on detecting irregular scores underscores the significance of refining the selection process to enhance overall quality.

Asdar's research serves to deepen our comprehension of the occurrence of unusual scores within the context of new student admissions selection. By delving into the intricacies of exam materials and aligning them with the objectives of the selection process, the study underscores the necessity of adopting a nuanced approach in evaluating the capabilities of prospective students. Moreover, it sheds light on the imperative to scrutinize the efficacy of the selection methodologies employed by educational institutions.

In essence, this research underscores the multifaceted nature of the new student admissions selection process. It prompts educators and stakeholders to reflect on the alignment between assessment tools, selection criteria, and educational objectives. By fostering a more discerning approach to evaluating prospective students and refining selection techniques, institutions can better fulfill their mandate of delivering high-quality education and nurturing academic excellence.

D. Optimization of Bubble Sort in the Context of New Student Admission Selection

In the realm of new student admission selection, optimizing the Bubble Sort algorithm becomes imperative to enhance the efficiency of the selection process. Although Bubble Sort stands as one of the elementary sorting algorithms, its time complexity tends to escalate notably when confronted with extensive datasets, as highlighted (N. Sari et al., 2022). This journal delves into the optimization of Bubble Sort, aiming to expedite the new student admission selection process through the employment of a brute force algorithm.

Bubble Sort fundamentally operates by iteratively shifting data into their appropriate positions. However, within the context of new student admission selection, where numerous criteria and variables necessitate evaluation for each prospective student, the sorting mechanism must be executed with greater efficiency and speed.

To elaborate on the optimization process, it involves a meticulous examination of the Bubble Sort's underlying mechanics and the identification of potential areas for enhancement. This may encompass scrutinizing the algorithm's comparison and swap operations, as well as exploring opportunities for parallelization or the integration of more sophisticated data structures.

Moreover, the optimization endeavor might entail the utilization of advanced techniques such as adaptive sorting strategies or hybrid approaches that combine the strengths of multiple sorting algorithms. For instance, incorporating techniques like early termination or optimizing the algorithm's traversal pattern could significantly alleviate its computational burden, particularly when processing large volumes of student data.

Furthermore, optimizing Bubble Sort in the context of new student admission selection necessitates a thorough understanding of the specific requirements and constraints inherent to the selection process. This involves considering factors such as the diversity of evaluation criteria, the size of the applicant pool, and the computational resources available for conducting the selection procedure.

In essence, optimizing Bubble Sort for new student admission selection transcends mere algorithmic refinement; it entails a comprehensive analysis of the unique challenges and objectives associated with the selection process. By leveraging advanced optimization techniques and tailoring them to the specific needs of the admission selection domain, significant improvements in efficiency and throughput can be achieved, ultimately enhancing the overall efficacy of the student admission process.

E. Integration of Brute Force and Bubble Sort Algorithms in New Student Admission Selection

In the context of new student admission selection, the use of the Bubble Sort algorithm can be integrated with the Brute Force algorithm to optimize the selection process. The Brute Force algorithm is often used for its ability to comprehensively evaluate all possible solutions (Gunawan & Tambunan, 2019). However, in the context of new student selection, especially in private schools with a large number of applicants, sorting student achievement data takes a considerable amount of time and resources. By integrating the Bubble Sort algorithm as part of the optimization strategy, the data sorting process can be performed more efficiently and quickly.

The Bubble Sort algorithm can help speed up the sorting process of student achievement data before it is used for new student admission selection. Although it has less efficiency with very large data, the Bubble Sort algorithm remains a relevant choice for data sorting in this context. By sorting student achievement data using the Bubble Sort algorithm before applying the Brute Force algorithm, teachers and school staff can save time and resources needed for the selection process, as well as obtain accurate and efficient results in determining new student admissions.

F. Algorithm Optimization

Algorithm optimization refers to the deliberate effort aimed at enhancing the effectiveness and speed of an algorithm by mitigating its time complexity, space utilization, or other resource requirements (Anggreani, D., 2020). In the context of this study, we introduce the concept of utilizing Bubble Sort optimization to augment the efficiency of the Brute Force algorithm in the selection process for new student admissions.

The integration of an optimized Bubble Sort technique serves as a pivotal enhancement, specifically tailored to streamline the execution time complexity of the Brute Force algorithm. By strategically implementing this optimization strategy, we can substantially curtail the duration required for the selection process, thereby expediting the overall procedure without compromising on the accuracy of the outcomes.

Through the amalgamation of these methodologies, our research endeavors to make notable contributions towards the refinement of selection methodologies, striving towards a paradigm where efficiency and accuracy coalesce seamlessly. This proposed approach holds promise in not only optimizing the specific context of student admission selection but also in fostering a broader discourse surrounding algorithmic efficiency enhancement across various domains.

G. Execution Time Complexity

In the context of optimizing the selection process for new student admissions, particularly through the application of Bubble Sort optimization within the Brute

Force algorithm, it is imperative to delve into the intricacies of execution time complexity.

When discussing execution time complexity, we are essentially exploring the efficiency of the algorithm concerning the time it takes to complete its task. In the context of the Brute Force algorithm, which exhaustively evaluates every possible solution, this becomes a critical consideration.

The execution time complexity of the Brute Force algorithm is typically quantified by analyzing the number of computational steps needed to execute the algorithm relative to the size of the input, which, in this scenario, equates to the number of prospective students under evaluation. Mathematically, this complexity is often denoted as $T(n) = O(f(n))$, where $T(n)$ represents the algorithm's execution time complexity as a function of the input size n, and $f(n)$ signifies the function indicating the computational steps required to evaluate all potential solutions. Given the exhaustive nature of the Brute Force approach, $f(n)$ can involve a substantial number of steps as it entails evaluating every conceivable combination of prospective students.

However, the integration of Bubble Sort optimization introduces an intriguing facet to this discussion. Despite Bubble Sort traditionally exhibiting a time complexity of $O(n^2)$, its incorporation within the Brute Force algorithm can yield efficiency gains. Bubble Sort operates by iteratively swapping adjacent elements if they are in the wrong order, gradually "bubbling" the larger elements towards the end of the list. In the context of student admissions, this means that students with lower ranks are more likely to be correctly positioned earlier in the process.

By leveraging Bubble Sort within the Brute Force algorithm, the number of computational steps required can be reduced significantly. This is because Bubble Sort facilitates the expedited arrangement of students based on their merit, thereby streamlining the overall selection process. Consequently, the integration of Bubble Sort as an optimization mechanism enhances the algorithm's efficiency and performance, ultimately contributing to a more expedient and effective student admission procedure (Basir, R. R., 2020).

III. Research Method

This research method employs the Bubble Sort algorithm approach for selecting new student admissions by applying the Brute Force concept. The implementation steps in the research method are as follows:

A. *Data Collection*

The process begins by determining the total number of student applications to be processed. For each participant, specific data is collected, including the participant's name, their psychometric test score, and their final score.

B. *Data Processing and Sorting*

The process of score calculation and participant sorting involves several key steps. Initially, after collecting the necessary data from participants, a combined score is calculated for each individual. This is achieved by applying a specific formula (1)

$$Value = (Psychometric\ Test\ Score \times 0.5) + (Final\ Score \times 0.5) \qquad (1)$$

This formula (1) ensures that both the psychometric test score and the final score are given equal weight, each contributing fifty percent to the overall value. Once these combined scores are determined, the next step is to organize the participant data. This is done using the Bubble Sort algorithm, which is employed to arrange the participants in descending order based on their calculated "Value". The Bubble Sort algorithm repeatedly compares and swaps adjacent elements if they are in the wrong order, ensuring that higher scores rise to the top of the list. This methodical approach guarantees a structured and fair ranking of participants according to their combined scores.

C. *Bubble Sort Brute Force Operation*

BubbleSortBruteForce Function is a function implements the Bubble Sort algorithm with an additional threshold for the admission quota. It utilizes the bubble_sort function (not explained here) to perform the sorting in descending order based on the "Value". After sorting, participants who meet the score and quota criteria are selected using the Brute Force approach.

D. *Result Display*

The display_result function presents a structured overview of participant selections, incorporating both accepted and rejected candidates. Each entry includes vital details such as the participant's name, psychometric test score, final score, and generated value. Accepted participants are prominently featured, sorted in descending order based on their generated value, showcasing the most promising candidates first. Furthermore, the display provides transparent insight by including information on rejected participants, ensuring thorough coverage of all outcomes.

E. *Implementation of Functions in Code*

The implementation of functions within the code is crucial for the seamless execution of various tasks inherent to the program. The 'bubbleSortBruteForce' function, for instance, integrates the Bubble Sort algorithm, accompanied by a threshold mechanism based on the quota capacity. This function streamlines the participant selection process by sorting them using Bubble Sort and subsequently filtering participants who meet the specified criteria. It optimizes the selection process, ensuring efficiency and accuracy in participant selection.

Moving forward, the 'display_result' function plays a pivotal role in presenting the outcomes of the participant selections. It distinguishes between accepted and rejected candidates, structuring the information in a clear and organized manner. Accepted participants are sorted based on the derived value, enhancing clarity and facilitating comprehension of the displayed data.

Additionally, the 'import_from_csv' function facilitates the seamless integration of participant data from a CSV file. It ensures the preservation of existing data while accommodating the inclusion of unique entries, contributing to the overall robustness of the program's data handling capabilities.

Furthermore, the 'read_participant_data' function retrieves participant information from a CSV file utilizing the pandas library. It incorporates error handling to address scenarios where the file may be inaccessible, ensuring the smooth execution of the program under varying circumstances.

Moreover, the 'view_participants' function offers a structured presentation of participant details, encompassing essential information such as ID, name, psychometric test scores, final score, and the generated value. This comprehensive display enhances user understanding and facilitates informed decision-making.

Lastly, the 'main' function serves as the orchestrator of the program flow, providing users with the flexibility to either view participant details or execute the selection process using the Bubble Sort algorithm with a predefined threshold. User inputs are processed to trigger the appropriate functions, ensuring a user-friendly and intuitive program interface.

Complementing this comprehensive explanation, the utilization of the pyvis.network module for visualizing the program flow furnishes a lucid depiction of the research methodology's progression. This visualization (Figure 1) amalgamates the elucidated functions to furnish a coherent portrayal, augmenting comprehension and aiding in grasping the intricacies of the implemented system.

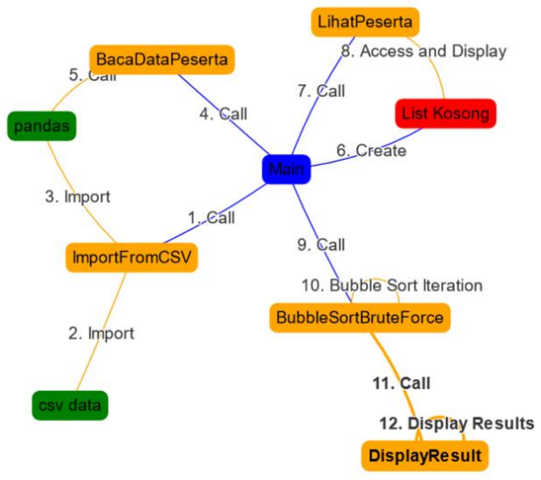

Figure 1. Program Flow

IV. Results and Discussion

A. Results of Selection using Brute Force Bubble Sort

The selection results using the brute force bubble sort algorithm demonstrate the algorithm's ability to sort and select participants based on their psychometric scores and final scores. The application of the brute force bubble sort algorithm involved iterating through the list of participants, comparing each pair of adjacent elements, and swapping them if they were in the wrong order. This process continued until the entire list was sorted. Through this method, the algorithm effectively arranged participants in ascending or descending order, depending on the specified criteria. The efficiency of the brute force bubble sort algorithm, however, may vary depending on the size of the participant pool and the complexity of the sorting criteria. Nevertheless, the obtained selection results showcase the algorithm's functionality in handling sorting tasks within this context.

1. Accepted Participants

The following Table 1, lists the participants who meet the criteria of psychometric scores and final scores as determined by the algorithm. These participants are ranked based on their scores from highest to lowest.

Table 1. Results of Bubble Sort Brute Force

Quota Capacity : 80 participants			
Result of bubbleSort Brute Force			
Participant Name	**Psychometric Test Score**	**Final Score**	**Value**
Fitri	100	99	99.5
Rabia	95	100	97.5
Dianna	90	100	95
Enisa	100	90	95
...	...	...	...
Khansa	90	70	80
Fauzan	100	100	100
Bahira	100	100	100
Eshal	100	100	100
Number of Accepted Participants			: 65 orang

2. Rejected Participants

Table 2 show the lists the participants who did not meet the selection criteria or for whom the quota for accepted participants had already been filled.

Table 2. Rejected Participants

Rejected Participants			
Participant Name	**Psychometric Test Score**	**Final Score**	**Value**
Adillah	90	65	77.5
Zimraan	80	75	77.5
Yasmin	95	60	77.5
Amira	75	80	77.5
...	...	...	...
Cyra	60	60	60
Gamila	60	60	60
Laila	60	60	60
Number of Rejected Participants			: 66 orang

B. Discussion

Bubble sort, despite its simplicity and minimal space requirements, presents both advantages and limitations that must be considered in practical applications.

1. Advantages

The bubble sort algorithm offers several advantages, making it a popular choice in various contexts. Firstly, its simplicity stands out as a key feature. Unlike more complex sorting algorithms, bubble sort is easy to grasp and implement, even for individuals with limited programming experience. This straightforward nature makes it an ideal starting point for learning about sorting algorithms, contributing to its widespread use in educational settings.

Another notable advantage of bubble sort is its minimal space requirements. Unlike some sorting methods that necessitate additional memory allocation, bubble sort operates in-place. This means it doesn't need extra space proportional to the input array's size. Instead, it only requires a constant amount of memory (O(1)) to store temporary variables during the sorting process. This efficient memory usage makes bubble sort well-suited for systems with constrained memory resources, enhancing its practicality in various computing environments.

2. Limitations

Bubble sort, despite its straightforward implementation, encounters significant drawbacks primarily due to its high time complexity, especially in scenarios where sorting large datasets is required. Its time complexity of $O(n^2)$, where 'n' represents the number of elements in the array, implies that the algorithm's performance degrades rapidly as the input size increases. This quadratic growth in computational effort stems from the algorithm's need to perform multiple comparisons and swaps during each iteration, resulting in an inefficient sorting process.

Furthermore, bubble sort exhibits inefficiency when confronted with nearly sorted data. While it can achieve its best-case time complexity of O(n) when the array is already sorted, it struggles to efficiently handle data that is even slightly disordered. In cases where only a few elements are out of order, bubble sort continues to perform unnecessary element swaps until the entire array is sorted. This behavior prolongs the sorting process unnecessarily, making bubble sort unsuitable for datasets where elements are frequently positioned close to their final sorted positions. Consequently, bubble sort may not be the optimal choice for sorting partially sorted lists or datasets with minimal variations, as its inefficiency becomes more pronounced in such scenarios.

Overall, while bubble sort offers simplicity and minimal space requirements, its high time complexity and inefficiency for nearly sorted data limit its practical utility in real-world applications. Alternative sorting algorithms with better performance characteristics, such as quicksort or mergesort, may be preferred in situations where efficiency and scalability are paramount. However, bubble sort remains a valuable teaching tool and a point of reference for understanding fundamental sorting concepts in computer science.

C. Performance Evaluation

1. Time Complexity

By delving into asymptotic notation analysis, we uncover the inherent time complexity of the bubble sort algorithm, particularly focusing on its behavior in the worst-case scenario. This exploration reveals that bubble sort exhibits a relatively high time complexity, particularly as the size of the dataset increases. The implication here is significant: in scenarios where the dataset comprises a substantial number of participants, employing bubble sort might not be the most efficient approach due to its time-intensive nature.

2. Selection Capability

Despite its rudimentary nature, the brute force implementation of the bubble sort algorithm demonstrates commendable selection capabilities. It effectively sifts through the dataset and identifies participants based on predetermined criteria. This aspect underscores the algorithm's innate ability to fulfill the basic requirements of participant selection, even though it lacks the sophistication of more advanced sorting techniques.

3. Capacity Limitations

One notable drawback of the brute force bubble sort algorithm is its failure to account for intricate capacity quota limitations pertaining to participants. As such, it may necessitate further refinement or adjustments to adequately address scenarios where detailed capacity constraints come into play. This limitation underscores the algorithm's inherent simplicity and highlights the need for additional considerations when dealing with complex participant selection criteria.

In summary, while the brute force bubble sort algorithm can indeed serve as a viable option for participant selection tasks, it is imperative to acknowledge and address its limitations, particularly concerning time and space efficiency. For scenarios demanding optimal performance and scalability, more sophisticated methodologies such as greedy algorithms or dynamic programming present themselves as superior alternatives, offering enhanced efficiency and effectiveness in tackling participant selection challenges.

D. Program Flow Visualization

Program Flow Visualization plays a crucial role in comprehending the intricacies of algorithms, such as the Brute Force Bubble Sort. By employing the pyvis.network module, a graphical representation has been crafted (as depicted in Figure 2). This visualization serves as a dynamic medium to elucidate the flow of the program and the sequential execution of the Brute Force Bubble Sort algorithm.

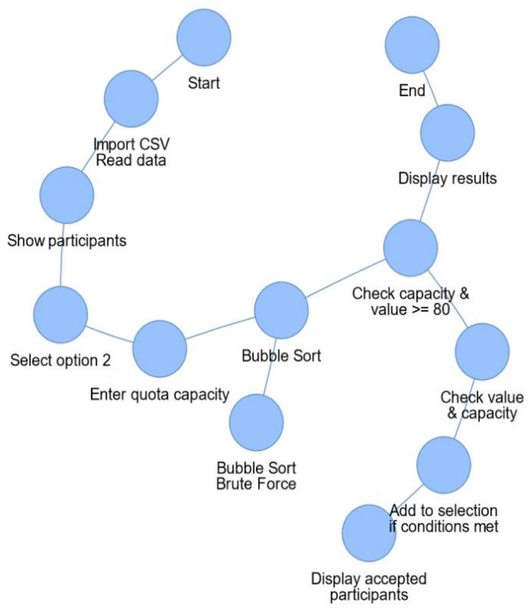

Figure 2. Program Output Flow

Through this graphical visualization, each step of the algorithm is visually portrayed, showcasing the interactions between elements and the progression of the sorting process. It provides a clear depiction of how the algorithm iterates through the dataset, compares elements, and rearranges them accordingly to achieve the desired sorted outcome.

Moreover, visualization aids in elucidating the underlying logic and decision-making process involved in the algorithm's execution. By visually mapping out the sequence of operations, observers can gain a deeper insight into the algorithm's inner workings and better grasp its efficiency and effectiveness in sorting data.

Additionally, the graphical representation offers a platform for interactive exploration, allowing users to dynamically interact with the visualization, such as zooming in on specific steps, pausing or resuming the animation, or even altering parameters to observe the algorithm's behavior under different conditions.

In essence, the Program Flow Visualization serves as a valuable tool for both learners and practitioners alike, facilitating a deeper understanding of the Brute Force Bubble Sort algorithm and enhancing their ability to comprehend and analyze algorithms through a visual lens.

V. CONCLUSION

The selection process for new student admissions is pivotal in ensuring the quality and sustainability of educational institutions. This process necessitates a thorough evaluation of various criteria to admit the most suitable candidates. The Brute Force algorithm, while comprehensive in considering all possible solutions, encounters significant challenges due to its increasing time complexity with larger problem sizes. This study addresses these challenges by integrating Bubble Sort optimization into the Brute Force algorithm to enhance its efficiency.

The integration of Bubble Sort significantly improves the initial arrangement of data, facilitating a more efficient Brute Force computation. This optimization reduces the execution time complexity, making the algorithm more suitable for handling larger datasets. The Brute Force Bubble Sort algorithm demonstrates effective participant selection based on psychometric and final scores. The selection results indicate that this hybrid approach maintains accuracy while enhancing computational efficiency.

By optimizing the Brute Force algorithm, educational institutions can process large volumes of application data more efficiently. This allows for more comprehensive and fair selection criteria without compromising processing time, ultimately saving time and resources. Despite its improvements, the Brute Force Bubble Sort algorithm has limitations in handling complex capacity constraints and high time complexity for large datasets. Future research could explore more advanced algorithms like Greedy or Dynamic Programming to further enhance efficiency and scalability.

This study contributes to the field of algorithm optimization by demonstrating the potential of integrating Bubble Sort with the Brute Force algorithm in new student admission selection. The findings suggest that this hybrid approach can significantly reduce the computational burden and improve the efficiency of the selection process. As educational institutions continue to face increasing data sizes and complexity in admissions, this research provides valuable insights and a practical solution to enhance the accuracy and efficiency of their selection methods.

The optimization techniques discussed in this study have broader implications beyond educational admissions. Similar combinatorial problems in various fields, such as data mining and resource allocation, can benefit from the integration of simple sorting algorithms with more complex ones. By addressing the high time complexity associated with traditional Brute Force methods, this research opens avenues for further exploration and application of hybrid algorithms in diverse domains. Overall, this study underscores the importance of algorithmic optimization in big data contexts and offers a novel approach to improving selection processes, making it a valuable contribution to both academic research and practical applications in educational settings and beyond.

REFERENCES

Aini, Q., Pratama, A. M. P., & Yasmin, F. D. (2019). Supply Chain Performance Analysis With Supply Chain Operation Research And Analytical Hierarchy Process (Case Study of MSME Tempo Susu Malang). Poltanesa Bulletin, 23(1), 20-27.

Anggreani, D., Wibawa, A. P., Purnawansyah, & Herman. (2020). Perbandingan Efisiensi Algoritma Sorting dalam Penggunaan Bandwidth [Comparison of

Sorting Algorithm Efficiency in Bandwidth Usage]. ILKOM Jurnal Ilmiah, 12(2), 96-103. https://doi.org/10.33096/ilkom.v12i2.538.96-103

Asdar, A. K. (2018). Ketidakwajaran Skor Seleksi Penerimaan Mahasiswa Baru [The Unfairness of New Student Admission Scores]. Jurnal Vijjacariya, 5(2), 1-16.

Balogun, G. B., Abdulraheem, M., Sadiku, P. O., Taofeek, O. D., & Adewale, A. (2023). Halstead's Complexity Measure of a Merge sort and Modified Merge sort Algorithms. Jambura Journal of Informatics, 5(2), 141-151. https://doi.org/10.37905/jji.v5i2.21995

Basir, R. R. (2020). Analisis Kompleksitas Ruang dan Waktu terhadap Laju Pertumbuhan Algoritma Heap Sort, Insertion Sort, dan Merge dengan Pemrograman Java [Space and Time Complexity Analysis of Heap Sort, Insertion Sort, and Merge Algorithms with Java Programming]. STRING (Satuan Tulisan Riset dan Inovasi Teknologi, 5(2), 109-118.

Chen, X. (2022). A Comparison of Greedy Algorithm and Dynamic Programming Algorithm. SHS Web of Conferences, 144, 03009. https://doi.org/10.1051/shsconf/202214403009

Fauzan, M., Ilham, N., & Saputra, A. (2023). Seleksi Penerimaan Mahasiswa Baru dengan Metode Pemecahan Masalah Algoritma Greedy Menggunakan Python [Selection of New Student Admission with Greedy Algorithm Problem Solving Method Using Python]. JURTI, 7(1), 32-38.

Gunawan, I., & Tambunan, H. S. (2019). Penggunaan Algoritma Sorting Bubble Sort untuk Penentuan Nilai Prestasi Siswa [Use of Bubble Sort Algorithm for Determining Student Achievement Scores]. Jurnal Sistem Informasi, 8(2), 296-304.

Hariski, F., Triayudi, A., & Soepriyono, G. (2023). Komparasi Metode Weighted Product (WP) dan Simple Additive Weighting (SAW) pada Sistem Pendukung Keputusan dalam Menentukan Pembangunan Infrastruktur Kelurahan. Journal of Computer System and Informatics (JoSYC), 4(3), 701-709. https://doi.org/10.47065/josyc.v4i3.3520

Kamalov, F., Santandreu Calonge, D., & Gurrib, I. (2023). New Era of Artificial Intelligence in Education: Towards a Sustainable Multifaceted Revolution. Sustainability (Switzerland), 15(16), 27. https://doi.org/10.3390/su151612451

Rizvi, D. Q., Rai, H., & Jaiswal, R. (2024). Sorting Algorithms in Focus: A Critical Examination of Sorting Algorithm Performance.

Sari, N., Gunawan, W. A., Sari, P. K., Zikri, I., & Syahputra, A. (2022). Analisis Algoritma Bubble Sort Secara Ascending dan Descending serta Implementasinya Menggunakan Bahasa Pemrograman Java [Analysis of Bubble Sort Algorithm in Ascending and Descending Order and Its Implementation Using Java Programming Language]. ABDI JURNAL, 3(1), 16-23.

Siddiq Sumi, A., Purnawansyah, & Syafie, L. (2018). Analisa Penerapan Algoritma Brute Force dalam Pencocokan String [Analysis of Brute Force Algorithm Application in String Matching]. Prosiding Seminar Nasional Ilmu Komputer dan Teknologi Informasi, 3(2), 88-92.

Soleh, M. ibnu. (2023). Use of Big Data in Education Management: Building Data-Powered Decision Making. Promis, 4(2), 113-138. https://doi.org/10.58410/promis.v4i2.735

Sunandar, E. (2021). Implementation of Bubble Sort Algorithm on 2 Fruit Models of Data Selection Using The Java Program Language. PETIR, 14(2), 159-169. https://doi.org/10.33322/petir.v14i2.946

Suraya, M. (2024). Algoritma Pemograman: Kunci Efisiensi dalam Pengelolaan Data Besar [Programming Algorithm: Key to Efficiency in Big Data Management]. Jurnal Pendidikan Berkarakter, 2(1), 308-314.

Thomas, P. (2023). Enhanced Efficiency in Sorting: Unveiling the Optimized Bubble Sort Algorithm. J Robot Auto Res, 4(3), 424-428.

Vierdansyah, D. Z., Al Ghafiqi, G., Dwi Putra, M. T., & Pradeka, D. (2023). Comparison Analysis of Bubble Sort and Insertion Sort Algorithm on the Selection of a Shop According to the Criteria. Journal of Computer Engineering, Electronics and Information Technology, 2(1), 39-52. https://doi.org/10.17509/coelite.v2i1.57091