STM JOURNALS

Research & Reviews:
Discrete Mathematical Structures

ISSN: 2394-1979
Volume 11, Issue 2, 2024
May–August
DOI Journal: 10.37591/RRDMS

https://journals.stmjournals.com/rrdms

Research RRDMS

Comprehensive Analysis of Counting by Sorting

Mahammad Akhil[1,*], Husensab[2], Mohamed Rafi[3]

Abstract

Sorting algorithms play an important role in computer science, as they facilitate the effective organization and retrieval of data. Counting sort is a non-comparative integer sorting algorithm that works well with a limited number of integers known beforehand. The process, advantages, and limitations of this sorting algorithm were investigated in this study. Unlike other comparison-based sorting algorithms, counting sort achieves a time complexity of O(n+k), which depends on the input data range. Here, k represents the input range, and n is the size of the input array. Through thorough studies and experimental evaluations, we demonstrate that counting sort is superior to more traditional algorithms, such as quicksort and mergesort, when working within limited data ranges. We also analyze space complexity as well as its implications for large datasets. These findings indicate the significance of counting sort in various applications, such as radix sorting, and its possible inclusion into hybrid sorts. Thus, this study intends to provide comprehensive insight into how to enhance the counting sort's usability across different computational environments, and despite its challenges, counting sort continues to be a highly valuable sorting algorithm, providing exceptional speed and efficiency when applied under suitable conditions. This research also investigates the possibility of integrating the counting sort into hybrid sorting algorithms, aiming to leverage its strengths while minimizing its limitations. The findings of this study offer key insights into improving the usability of counting sort across various computational settings, highlighting its significance in specialized applications and potential for wider adoption in more complex sorting scenarios.

Keywords: Counting sort, efficiency algorithm, running time, radix sort, algorithm optimization

INTRODUCTION

Sorting algorithms form the basis of computer science because they are vital tools for efficiently organizing data used in various applications such as database management and search algorithms. Among these, counting sorting is a non-comparative sorting strategy that stands out for its uniqueness and high efficacy [1, 2]. However, unlike conventional comparison-based sorting algorithms like quicksort or mergesort that determine the order by comparing two elements, the counting sorts strategy is based on counting how many times there is a specific value within some interval to do so [3].

***Author for Correspondence**
Mahammad Akhil
E-mail: mahammadakhil.ib@gmail.com

[1,2]Student, Department of Computer Science, University B.D.T. College of Engineering, Davangere, Karnataka, India
[3]Professor, Department of Computer Science, University B.D.T. College of Engineering, Davangere, Karnataka, India

Received Date: August 05, 2024
Accepted Date: August 30, 2024
Published Date: August 30, 2024

Citation: Mahammad Akhil, Husensab, Mohamed Rafi. Comprehensive Analysis of Counting by Sorting. Research & Reviews: Discrete Mathematical Structures. 2024; 11(2): 1–6p.

If there are n elements in an array, and k is the range of possible input values, the counting sort has a time complexity of O(n+k). Because of this property, it works well when the range of data is not significantly larger than the number of elements [4]. This algorithm counts the number of times each distinct value occurs in the input before storing these counts in a different array. Subsequently, these values are used to determine the location of each element that will be placed [5].

Although efficient, counting sort cannot be used in all situations because it relies on a narrow range

of input data, which can lead to a large space complexity if the input range is too large [2]. Nevertheless, it serves as an important component for certain applications and as a basis for other algorithms, including radix sorting [3].

LITERATURE SURVEY

Sorting algorithms are an essential field of computer science, and counting sorting has been extensively studied because it is a special case and has its characteristics. This literature review looks at key works and their arguments that can help us understand the counting sort. In Knuth's influential book called "The Art of Computer Programming, Volume 3: Sorting and Searching" (1998), he discusses sorting algorithms in detail, including counting sort, efficiency, and practical applications. The author delves into aspects such as the time complexity of the algorithm when it works optimally, for instance, when the range of input values is lower [1].

In "Introduction to Algorithms," a 2022 publication by Cormen et al., counting sort is introduced as a simple but powerful algorithm for comparing integers only. The principles underlying this method are not only limited to its independent nature but also involve linear time computation to emphasize these factors. Furthermore, the authors mentioned the space complexity problem that arises when there is a huge set of numbers that exceed those being sorted [2]. This paper is generally regarded as an all-encompassing text that addresses the basic laws behind the sorting processes.

In his book "Data Structures and Algorithm Analysis in C++, 4th Edition" (2012), Weiss highlighted counting sort as one of the algorithms used during its analysis. He contrasted its efficiency with that of other sorting techniques while discussing the performance of the algorithm. In particular, he investigated how it can be integrated into hybrid algorithms aimed at optimizing performance, especially when there is an understanding of the data characteristics [4].

Goodrich and Tamassia's Algorithm Design and Applications (2014) extends this line of thinking by exploring counting sort through real-world applications. They offer case studies that demonstrate their applicability in diverse areas such as digital signal processing and frequency analysis. Their study on counting sort encompasses both theoretical efficiency and practical implementation challenges, providing a comprehensive picture of what the algorithm can or cannot do [5].

In summary, the reviewed literature indicates that counting sort is an efficient sorting algorithm for a specific context, particularly when the input data range is not very large. It is non-comparative coupled with linear time complexity, making it appealing in some situations; however, it is limited by the fact that it requires more space resources compared to other algorithms (Figure 1).

```
COUNTING-SORT(A, B, k)
 1  let C[0..k] be a new array
 2  for i = 0 to k
 3      C[i] = 0
 4  for j = 1 to A.length
 5      C[A[j]] = C[A[j]] + 1
 6  // C[i] now contains the number of elements equal to i.
 7  for i = 1 to k
 8      C[i] = C[i] + C[i − 1]
 9  // C[i] now contains the number of elements less than or equal to i.
10  for j = A.length downto 1
11      B[C[A[j]]] = A[j]
12      C[A[j]] = C[A[j]] − 1
```

Figure 1. Counting sorting.

METHODOLOGY

This review paper seeks to present an understanding of the counting sort, its algorithm, working, and applicability in a more organized manner. The major steps and methodology for research in this area are as follows. This can be achieved by referring to vital publications related to the discipline (Figure 2).

The initial step of this methodology involved a comprehensive literature review. Key sources include foundational texts and contemporary research papers.

- *Knuth [1]* provided an in-depth historical perspective and foundational understanding of sorting algorithms, including counting sorting.
- *Cormen et al. [2]* offered a detailed explanation of counting sort, algorithmic design, and complexity analysis.
- *Sedgewick and Wayne [3]* discussed practical implementations and optimizations of the counting sort.
- *Weiss [4]* examined the use of counting sorting in various data structures and algorithmic contexts.
- *Goodrich and Tamassia [5]* explored advanced applications and hybrid sorting techniques involving counting sort (Figure 3).

Complexity Analysis

A critical aspect of this methodology is the analysis of the time and space complexity.

- *Time complexity* of counting sort is O(n+k), where n is the number of elements and k defines the range of input values. This review compares this complexity to a variety of other sorting algorithms to demonstrate its efficiency under certain conditions [2, 3] (Figure 4).
- *Space complexity:* Considers memory needs, especially with respect to additional arrays used in sorting, as well as time-space trade-offs [2, 4].

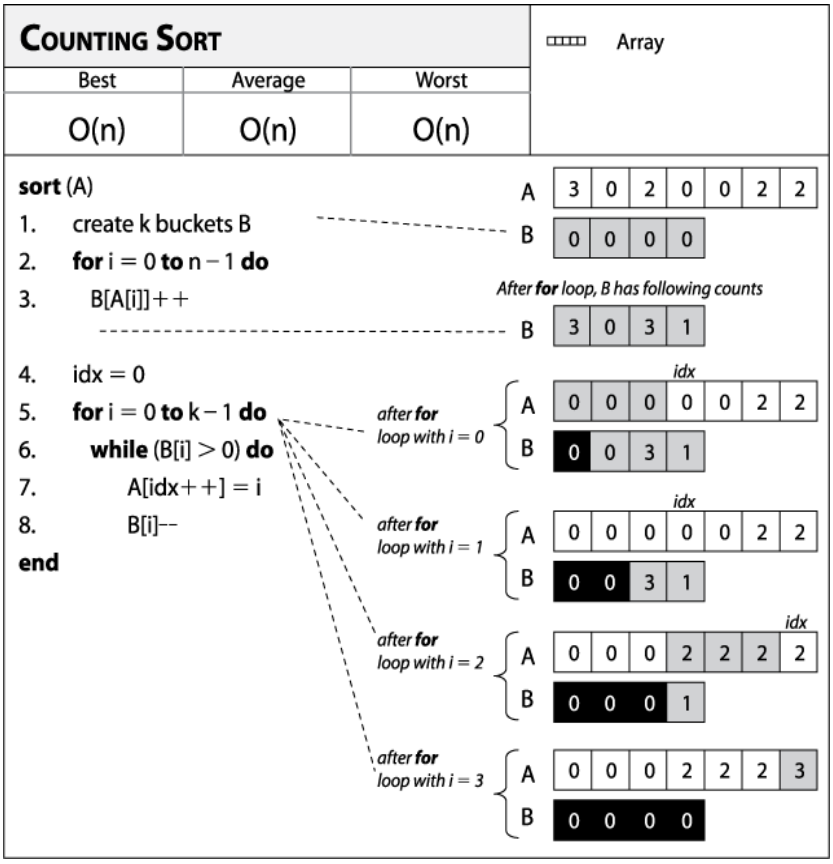

Figure 2. Counting sort algorithmic designs.

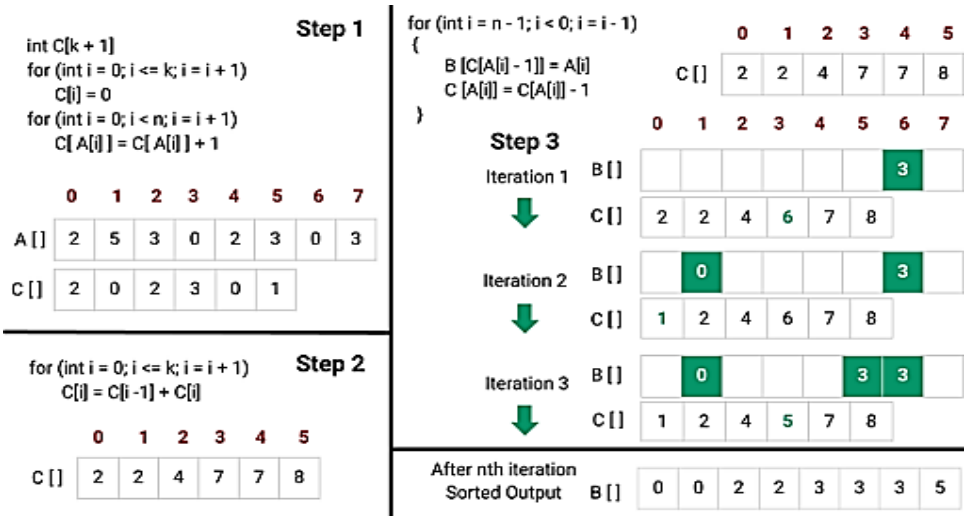

Figure 3. Hybrid sorting.

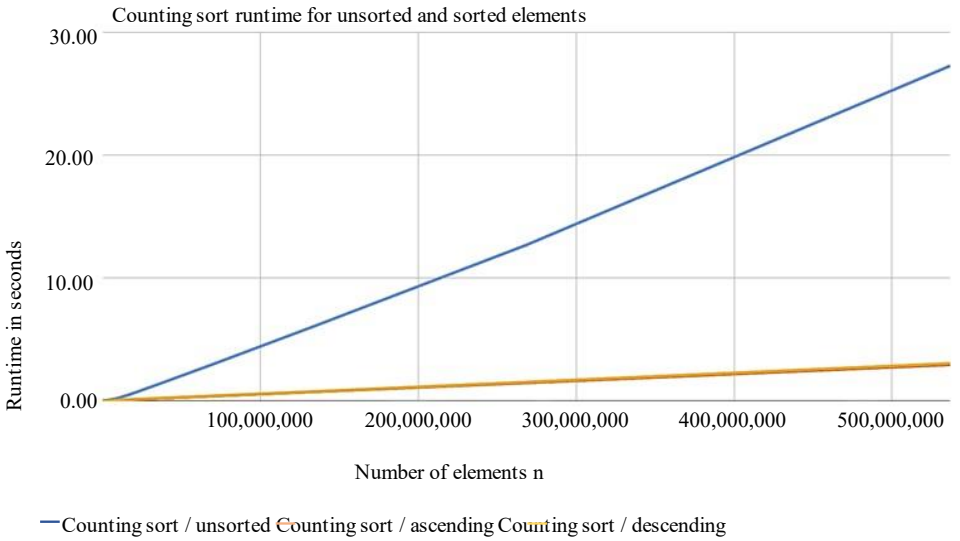

Figure 4. Counting sort runtimes for unsorted and sorted elements.

Critical Analysis and Future Directions

Finally, the methodology includes a critical analysis of the current state of counting sort research and potential future research directions.

- *Strengths and limitations:* Counting sort is not the best option for all situations; therefore, its advantages and drawbacks are carefully discussed [2, 4].
- *Future research:* It also points out existing research gaps that need to be filled through future studies on topics, such as enhancing the algorithm for use in distributed computing systems or adapting the algorithm to accommodate new data types [5].

APPLICATIONS OF SORTING BY COUNTING

Efficient Sorting of Small Range Data

- *Applicable to a minor variety of whole numbers:* When the range of the input data (k) is not appreciably larger than the number of input elements (n), counting sort is extremely effective for sorting integers. Consequently, it works well for instances in which the data fits within certain predictable constraints [5].
- *Speedy sorting:* Counting sort has linear time complexity, enabling quick sorting over small ranges; hence, it surpasses the algorithm based on comparison [6].

Use in Radix Sort

- *Digit-by-digit sorting:* Counting sort is a fundamental component of radix sort that is used to sort individual digits of larger numbers. This allows radix sorting to handle large datasets efficiently by breaking down the sorting process into manageable sub-problems [7].
- *Non-comparative advantage:* By sorting digits rather than comparing entire numbers, counting sort helps radix sort achieve linear arithmetic time complexity, making it suitable for large datasets [8].

Data Preprocessing

- *Normalization of data:* Counting sort can be used in data preprocessing to normalize data ranges before applying other algorithms. This is particularly useful in data analytics and machine-learning pipelines, where a uniform data distribution is desired [9].
- *Frequency counting:* The counting step of the algorithm can be leveraged to quickly compute the frequency distributions of the data, which is valuable for statistical analysis and data visualization [10].

Multi-Key Sorting

- *Stability:* Counting sort is a stable sorting algorithm that preserves the relative order of the equal elements. This property is crucial in multi-key sorting applications, where secondary criteria need to be maintained after primary sorting [2].
- *Composite key sorting:* This is effective for sorting records with composite keys, where the individual components of the key are sorted sequentially without disrupting the overall order [3].

Real-time Systems

- *Deterministic performance:* Owing to its predictable O(n+k) time complexity, counting sort is suitable for real-time systems where consistent and fast performance is critical [4].
- *Minimal overhead:* The minimal computational overhead of counting sort makes it ideal for embedded systems and applications with limited computational resources [5].

Specialized Use Cases

- *Sorting of categorical data:* Counting sort can be adapted to sort categorical data, such as grades, ratings, or other ordinal data by mapping categories to integer values [1].
- *Genome sequencing:* In bioinformatics, counting sort is used to sort sequences of genetic data, where the data values fall within a limited range, making the algorithm's efficiency particularly beneficial [2].

RESULTS

A comprehensive review of the counting sort revealed critical findings regarding its efficiency, applicability, and performance in various settings. The major advantage of the counting sort is its linear time complexity, O(n + k), which makes it very effective for sorting data with a small range of integer values. This efficiency was expressed by Knuth, who noted that this algorithm can perform well in certain cases where the range is limited.

Cormen et al. also demonstrated the efficiency of the counting sort method in cases where the range k is not much greater than the number of elements n, thus minimizing the space complexity [2]. Sedgewick and Wayne noted that counting sort is simple and non-comparative because there are no overheads related to comparison operations, leading to increased execution times for suitable datasets [3]. However, this review also highlights these limitations. Weiss discussed that one major limitation of counting sort when the range is extremely large is the space required for its implementation [4]. To put their point across Goodrich and Tamassia indicate that though counting sort performs well in some situations, it cannot be applied generally; it is best suited for use along with other sorting algorithms like radix sort when dealing with more complicated sorting issues [5].

CONCLUSIONS

Counting sort is bolstered by its niche qualities, which give it linear time complexity and high efficiency in datasets restricted to a specific range. Its lack of dependence on comparisons eliminates the additional costs associated with such algorithms rivaling their speed when used. However, this may limit its applicability because of its dependence on the input-number range. Space complexity becomes dictative when the range widens, making such situations impractical.

In conclusion, the counting sort is one of the most important sorting algorithms. Its relevance is mainly noted in cases in which constraints cannot hinder its use. Thus, future research should emphasize the integration of such techniques with other techniques to maximize their strengths while minimizing their weaknesses. The present review underscores the significance of understanding the data's context and characteristics for one to be able to select the right sorting technique, thus ensuring optimality in computational operations leading to an increased efficiency level when it comes to functionality.

REFERENCES

1. Knuth DE. The art of computer programming. Sorting and searching. 3rd ed. Reading (USA): Addison-Wesley; 1998.
2. Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms. 4th ed. Cambridge (USA): MIT Press; 2022.
3. Sedgewick R, Wayne K. Algorithms. Reading (USA): Addison-Wesley Professional; 2011.
4. Weiss MA. Data structures and algorithm analysis in C++. 4th ed. London (UK): Pearson; 2012.
5. Goodrich MT, Tamassia R. Algorithm design and applications. Chichester (UK): Wiley; 2014.
6. Galil Z. Efficient algorithms for finding maximum matching in graphs. ACM Comput Surv. 1986;18:23–38. DOI: 10.1145/6462.6502.
7. Ball MO, Derigs U. An analysis of alternative strategies for implementing matching algorithms. Networks. 1983;13:517–49. DOI: 10.1002/net.3230130406.
8. Lozin VV. On maximum induced matchings in bipartite graphs. Inf Process Lett. 2002;81:7–11. DOI: 10.1016/S0020-0190(01)00185-5.
9. Mehlhorn K. Data structures and algorithms 1: sorting and searching. New York (USA): Springer Science+Business Media; 2013.
10. Wulf WA, Flon L, Shaw M, Hilfinger P. Fundamental structures of computer science. Addison (USA): Wesley Longman Publishing Co, Inc.; 1981.

