ISSN (e): 2250 – 3005 || Volume, 14 || Issue, 3|| May - June – 2024 ||
International Journal of Computational Engineering Research (IJCER)

CBSA: Clustering Based Sorting Algorithm

Omar Kettani
Scientific Institute, Mohammed V University
Rabat, Morocco

Abstract: *This paper introduces CBSA, a new deterministic sequential sorting algorithm based on clustering that addresses the challenge faced by existing sorting methods when dealing with large data sets. This algorithm achieves in linear space a complexity of O(nloglog(n+1-(logloglogn))) time for sorting n integer numbers and a complexity of O(n(log(n+1-(loglogn)$^{1/2}$)$^{1/2}$) time for sorting n real numbers, improving the computational complexity of the state of the art sorting algorithm, for both integer and real numbers. The paper concludes by discussing the potential of this new sorting algorithm and the future directions for its development and refinement.*

Keywords: *Sorting; Clustering; Han's algorithm; computational complexity.*

Date of Submission: 07-06-2024 Date of acceptance: 21-06-2024

I. Introduction

Sorting and clustering are two fundamental tools in the field of computer science. While sorting consists to order data according to some linear relationship concerning the data, clustering aims to group data objects into distinct groups, or clusters, such that the data objects within each cluster are more similar to one another than they are to data objects in other clusters. With the exponential growth of data, the need for efficient and scalable clustering and sorting algorithms has become increasingly pressing. Traditional sorting methods, while still widely used, often struggle with the complexity and sheer size of modern data sets. In an effort to address these challenges, a new sorting algorithm based on clustering has been developed. This algorithm utilizes a clustering approach to data organization as a preprocessing phase, resulting in improved speed. The purpose of this paper is to introduce this new sorting algorithm, provide an explanation of its inner workings, evaluate its computational complexity and compare with the state of the art sorting algorithm.

This paper is organized as follows: In section II, some related work are presented. Section III describes the pseudo-code of the proposed algorithm and provides an analysis of its computational complexity. Finally, section IV discuss some future work and concludes the paper.

II. Related work

Sorting algorithms have been an active area of research in computer science for several decades, and numerous sorting methods have been proposed to address the challenges of sorting large datasets efficiently and accurately. Some of the most widely used sorting algorithms include QuickSort, MergeSort, and HeapSort, RadixSort,BucketSort, TimSort and LibrarySort with complexities varying between O(nlogn) and O(n^2). These algorithms have different strengths and weaknesses in terms of time complexity, stability, and scalability.

QuickSort [1], for instance, is a fast sorting algorithm that uses the partitioning technique to sort data. However, it is not stable, meaning that equal elements may not retain their relative order in the sorted result. QuickSort is a divide-and-conquer algorithm that selects a pivot element and partitions the data around it. Its advantages are: a fast average case performance, and a simplicity in its implementation. Its limitations are: its worst-case performance can be slow, and it could be not stable.

MergeSort [2], on the other hand, is a stable sorting algorithm that uses the merging technique to sort data, but it has a higher time complexity compared to QuickSort. MergeSort is a divide-and-conquer algorithm that recursively splits the data into smaller sub-arrays, sorts them, and then combines them. Its strengths are: stability, efficiency for large data sets, and suitability for both linked lists and arrays. Its weaknesses are: slowness for small data sets due to overhead, non-efficiency for data that can't be easily divided into smaller parts.

RadixSort [3], a non-comparison-based sorting algorithm that sorts data based on the digits of each element. Its advantages are: efficiency for data with a large number of digits. Its limitations are: slowness for large data sets, inefficiency for data with a small number of digits, inadaptivity.

BucketSort [4]: is a distribution-based sorting algorithm that sorts elements into buckets and then sorts the buckets. Its strengths are: fastness for small data sets, efficiency for data with a limited number of values. Its weaknesses are: slowness for large data sets, inefficiency for data with a large number of values, inadaptivity.

TimSort [5]: TimSort is a hybrid, sorting algorithm, derived from merge sort and insertion sort.

LibrarySort [6]: is a sorting algorithm that uses an insertion sort, but with gaps in the array to accelerate subsequent insertions.

These are some of the commonly used sorting algorithms, and the choice of which one to use depends on the specific requirements of the problem being solved.

Recently, researchers have proposed various hybrid sorting algorithms that aim to combine the strengths of different sorting methods to provide a more efficient and stable sorting solution. In [7], Han suggested a deterministic sorting algorithm with complexity O(nloglogn) time and linear space, which is the state of the art for deterministic integer sorting. Recently, Han proposed in [8], an $O(n(\log n)^{1/2})$ time and linear space algorithm for sorting n real numbers, breaking the O(nlogn) time bound for sorting real numbers. Currently, this algorithm is considered to have the lowest computational complexity. In the present paper, a new sorting method based on clustering and using Han's algorithm as a subroutine is proposed in order to improve slightly these bounds.

III. Proposed approach

The proposed CBSA algorithm consists first to set k, the number of clusters to an appropriate value (depending if it sorts integers or real numbers), aiming to improve the overall computational complexity. Then, an O(nk) clustering subroutine like those proposed in [9, 10, 11] is applied to an one dimensional input dataset A. The second phase consists to sort the k cluster centers c_j obtained in the first phase, by using Han's algorithm as a subroutine. In the third phase, each cluster C_j is sorted by using Han's algorithm as a subroutine. Finally, CBSA outputs, in a sorted array SA, the clusters according to the order of the index J found in the second phase. A pseudo-code of the proposed algorithm for sorting both integers and real numbers is depicted below:

Pseudo-code of the proposed CBSAalgorithm for sorting integers:

Input: An array A of n integers.
Output: A sorted array SA of these n integers.
k ← round(logloglogn)
[C,c]← Cluster(A,k)
[Sc, J]← Han_real_sort(c)
SA←[ ]
For i=1 to k **do**
j ← J(i)
SC_j←Han_integer_sort (C_j)
SA←SA ∪SC_j
end For

Pseudo-code of the proposed CBSAalgorithm for sorting real numbers:

Input: An array A of n real numbers.
Output: A sorted array SA of these n real numbers.
k ← round $((\log n)^{1/2})$
[C,c]← Cluster(A,k)
[Sc, J]← Han_real_sort(c)
SA←[ ]
For i=1 to k **do**
j ← J(i)
SC_j←Han_real_sort (C_j)
SA←SA ∪SC_j
end For

Computational complexity:
Since the algorithms used in the clustering subroutine [9, 10, 11] and the sorting subroutine [7, 8] run in linear space, then CBSA has a linear space complexity.
Let $T_i(n)$ be the time complexity of phase i=1,2,3 of CBSA.
For the integer case, since the time complexity of algorithms used in the clustering subroutine [9, 10, 11] is O(nk) then $T_1(n)=O(nk)=O(n(loglogn))$.
On the other hand, $T_2(n)=O(k(logk)^{1/2})$ because of the time complexity of Han's real sorting subroutine [8]. Thus $T_2(n)\leq O(n)$, because $loglogn(loglogloglogn)^{1/2}\leq n$.
Let n_j be the size of clusters C_j, for j=1,...,k. Since the time complexity of Han's integer sorting subroutine [7] is O(n(loglogn)), then

$$T_3(n)=\sum_{j=1}^{k} n_j(loglogn_j) \leq loglogM\sum_{j=1}^{k} n_j = n\ loglogM, \text{ since} \sum_{j=1}^{k} n_j = n \text{ and } M=max(n_j)$$

M reaches its maximal value when all the k clusters C_j, expect one (whose size is M) are singletons.
Then, $M\leq n-(k-1)$ and
$T_3(n)\leq nloglog(n+1-(loglogloglogn))$. Therefore,T(n) the overall time complexity of CBSA in the integer case is:
$T(n)=T_1(n)+T_2(n)+T_3(n)=O(nloglog(n+1-(loglogloglogn)))$ since $T_1(n)\leq T_3(n)$ and $T_2(n)\leq T_3(n)$.
Thus the time complexity of CBSA is a slightly improvement over the last time complexity bound [7].

In a similar way for the real numbers case,
$T_1(n)=O(nk)=O(n(loglogn)^{1/2})$.
$T_2(n)=O(k(logk)^{1/2})=O((loglogn)^{1/2}(log((loglogn)^{1/2}))^{1/2})$ and $T_2(n)\leq O(n)$,
because $(loglogn)^{1/2}(log((loglogn)^{1/2}))^{1/2}\leq n$.
On the other hand, $T_3(n)\leq nlog(n+1-(loglogn)\leq^{1/2})^{1/2}$
Therefore, T(n) the overall time complexity of CBSA for the real numbers case is:
$T(n)=T_1(n)+T_2(n)+T_3(n)=O(nlog(n+1-(loglogn)^{1/2})^{1/2})$ since $T_1(n)\leq T_3(n)$ and $T_2(n)\leq T_3(n)$.
Thus, again the time complexity of CBSA is slightly improved over the last time complexity bound in this case [8].

IV. Conclusion

In conclusion, the new clusteringbased sorting algorithm CBSA introduced in this paper has shown an improvement in term of computational complexity. Its approach to data organization based on clustering makes it a highly efficient solution for sorting vast amounts of information. Despite being a relatively new technique, early evaluations suggest that it has the potential to outperform traditional sorting algorithms, especially in complex data structures.

In terms of future work, there is potential to further extend and optimize CBSA. For example, the algorithm could be modified to handle different types of data, such as sparse data sets or data with a large number of missing values. CBSA could also be optimized for specific types of hardware, such as GPUs or multi-core processors, to take full advantage of the computational resources available. A parallel version of this algorithm is under consideration. Another possible improvement will consist to find a lower computational complexity bound by considering a recursive version of this algorithm where instead of using Han's algorithm as a subroutine, the algorithm uses CBSA in a recursive call. Finally, it will be useful to develop an efficient implementation of this algorithm in order to conduct some experiments aiming to evaluate its performance and speed on several different datasets. As more research is conducted and the algorithm is further refined, it is likely to become a valuable tool for various applications.

References

[1]. Sedgewick, Robert (1 September 1998). Algorithms In C: Fundamentals, Data Structures, Sorting, Searching, Parts 1-4 (3 ed.). Pearson Education. ISBN 978-81-317-1291-7. Retrieved 27 November 2012.
[2]. Ajtai, M.; Komlós, J.; Szemerédi, E. (1983). An O(n log n) sorting network. STOC '83. Proceedings of the fifteenth annual ACM symposium on Theory of computing. pp. 1–9. doi:10.1145/800061.808726. ISBN 0-89791-099-0.
[3]. Goodrich, Michael T.; Tamassia, Roberto (2002). "4.5 Bucket-Sort and Radix-Sort". Algorithm Design: Foundations, Analysis, and Internet Examples. John Wiley & Sons. pp. 241–243. ISBN 978-0-471-38365-9.
[4]. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001) [1990]. Introduction to Algorithms (2nd ed.). MIT Press and McGraw-Hill. ISBN 0-262-03293-7.

[5]. Peters, Tim. "[Python-Dev] Sorting".
[6]. Bender, Michael A.; Farach-Colton, Martín; Mosteiro, Miguel A. (1 July 2004). "Insertion Sort is O(n log n)". arXiv:cs/0407003.
[7]. Yijie Han "Deterministic sorting in O(nloglogn) time and linear space" Journal of Algorithms Volume 50, Issue 1, January 2004, Pages 96-105
[8]. Yijie Han "Sorting Real Numbers in $O(n(\log n)^{1/2})$) Time and Linear Space. " Algorithmica 82, 966–978 (2020). https://doi.org/10.1007/s00453-019-00626-0
[9]. Kel'manov, A., Khandeev, V. (2020). Exact Linear-Time Algorithm for Parameterized *K*-Means Problem with Optimized Number of Clusters in the 1D Case. In: Sergeyev, Y., Kvasov, D. (eds) Numerical Computations: Theory and Algorithms. NUMTA 2019. Lecture Notes in Computer Science (), vol 11974. Springer, Cham. https://doi.org/10.1007/978-3-030-40616-5_35.
[10]. J. Marshall, Lawrence C. Rafsky "Exact clustering in linear time" Published 17 February 2017 Computer Science ArXiv 1702.05425.v2
[11]. A. Jørgensen, Kasper Green Larsen, +1 author J. Nielsen "Fast Exact k-Means, k-Medians and Bregman Divergence Clustering in 1D' Published 25 January 2017 Computer Science ArXiv arXiv:1701.07204