An Efficient Optimized Sorting Technique Using Combination of Other Techniques

Nitin Mishra, Pranjal Srivastava, Khushi Gupta and Kunal Singh Teotia

December 29, 2020

An Efficient Optimized sorting technique using combination of other techniques

Dr. Nitin Mishra
School of Computing Science and Engineering
Galgotias University
Greater Noida , India
Email: drnitinmishra10@gmail.com

PRANJAL SRIVASTAVA
School of Computing Science and Engineering
Galgotias University
Greater Noida , India

Khushi Gupta
School of Computing Science and Engineering
Galgotias University
Greater Noida , India

Kunal Singh Teotia
School of Computing Science and Engineering
Galgotias University
Greater Noida , India

***Abstract*—The problem of sorting is a problem that arises frequently in computer programming and though which is need to be resolved. Many different sorting algorithms have been developed and improved to make sorting optimized and fast. As a measure of performance mainly the average number of operations or the average execution times of these algorithms have been compared. There is no one sorting method that is best for every situation. Some of the factors to be considered in choosing a sorting algorithm include the size of the list to be sorted, the programming effort, the number of words of main memory available,the size of disk or tape units, the extent to which the list is already ordered, and the distribution of values.**

***Index Terms*—Sorting, complexity lists, comparisons, movement sorting algorithms, methods, stable,unstable, internal sorting**

I. Introduction

Sorting is one of the very most important and well-studied problems in computer science. Many great algorithms offer a variety of trades for efficient, easy-to-use, memory usage, and other items. However, these algorithms do not think features of modern computer technology that greatly influence performance. A large number of suggested filtering algorithms and their asymptotic complexity, depending on the comparative value or multiplication value, be careful updated. In the past, there has been a growing interest in the development of filtering algorithms that do not affect their asymptotic complexity but nonetheless improve performance by improving the data center. Since the advent of the computer, the filtering issue has been very interesting research, perhaps because of the problem of solving it well without being an easy, standard statement. It cannot always be said that a single filtering algorithm is better there is another filtering algorithm. The performance of various filtering algorithms depends on the data that is filtered. Sorting is often understood as a reorganization process of a given set of items in a particular order. In particular, filtering is a relevant topic in the show a wide variety of algorithms, all with the same purpose, many of them are good in a sense and most of them are more valuable than others. The filter model is well-suited to show how much profit it is in performance can be achieved by building a complex algorithm their obvious methods are readily available. In our research we are going to optimize the way of doing sorting through algorithms. We are going to propose a new algorithm which acts as an orchestrator for our function. Like, when the array is given to us it goes in the orchestrator method and here the main task comes. The orchestrator automatically selects from the different sorting techniques going to be used after the completion of one step. Keeping the view on their effectiveness at every case i.e. Worst case, Average case, and best case, the sorting technique is chosen. In this newly proposed algorithm, we can use multiple sorting techniques in a single problem, thus getting the solution more optimized.

II. OBJECTIVE

The aim of this optimization technique is to reduce the certain complexities such as time complexity space complexity and make the sorting algorithm technique more feasible and easy to solve. The following objectives are designed to fulfill the aim of this optimization technique:

1) Automatic selection of sorting algorithm according to the need and requirement of the particular step.
2) It will select sorting technique which has less time complexity and less space complexity and which best complements the sorting array.

These function will help the array to sort faster with less complexity and it will more efficient. In our research paper we will mention all the advantages of this optimization that how it will be beneficial and all the drawbacks or problems for which it lacks to help.

III. RELATED WORK

Four-dimensional (4D) computed tomography (CT) has been widely used as a visual aid in radiotherapy. Two of the most widely used 4D CT algorithms classify images with a corresponding respiratory phase or transfer to a predefined number of drums, and tend to look at art objects for switching between sleeping areas. The purpose of this work is to show how to reduce the activity of motion in 4D CT by incorporating anatomic similarities in phase or migration planning agreements. Methods: Ten patient data sets are rearranged using both migration and phase-level layout planning. Standard filtering methods allow for the selection of a close-up photo of a neighbor at a time or a move within each barrel. In our approach, in each bed area or migration or section defines the center of the drum distance in terms of which multiple selected images are selected. The coefficients of the intersection of the two ends between the boundary pieces of the connector between the adjacent sofa areas are calculated for all pairs. The two pieces have a high cohesiveness when they are similar in shape. Nominees for each bin are selected to maximize slide integration across all preset data using the shortcut Dijkstra method algorithm. To examine the reduction of archeology, two thoracic radiation oncologists independently compare 4D datasets using standardized dasets, blinded to the filter path, to select which less moving objects. Agreement between reviewers was assessed using points weighing in kappa. Results: Anatomically based image selection resulted in 4D CT data sets with significantly reduced reductions for all migrations (P = 0.0063) and phase classification (P = 0.00022). There was a good agreement between the two analysts, with a total agreement 34 times and a complete disagreement six times. Conclusions: Prepared editing using anatomic similarities significantly reduces 4D CT moving objects compared to standard phase or locally-based filtering. This advanced filtering algorithm is a direct extension of the most common 4D CT filtering algorithms.

IV. Optimizing Sorting with Genetic Algorithms

The growing complexity of modern processors has made the development of more efficient code more difficult. The production of manual code takes a lot of time, but it is often the only option as the code generated by modern compiler technology often works much lower than hand-coded codes. A promising code-generating strategy, used by programs such as ATLAS, FFTW, and SPIRAL, uses powerful search to find startup parameter values, such as tile size and tutorials, that bring the closest performance to a particular machine. However, this approach is already effective in scientific codes whose performance does not depend on input data. In this paper we learn about machine learning techniques to extend the dynamic search in the construction of filtering practices, its effectiveness depends on the characteristics of the installation and construction of the intended machine. We build on previous research that selects the "pure" filtering algorithm at the beginning of the computer as a standard deviation function. The approach discussed in this paper uses genetic algorithms and a classification system to create highly structured hybrid algorithms that are able to adapt to input data. Our results show that such algorithms generated using the method presented in this paper are very effective in addressing the complex interactions between building and input data features and that the resulting code works much better than standard programming and code developed in our previous study. In particular, the routes made using our method work better than all the commercial libraries we have tried to install IBM ESSL, INTEL MKL and C ++ STL The best algorithm we have been able to do is at an average of 26% and 62% faster than IBM ESSL in IBM Power 3 and IBM Power 4, respectively.

V. Optimum Sorting Algorithm

By comparing internal filtering algorithms to determine the best for a certain number of elements, we face problems defining what we mean by fine-tuning the algorithm. , first, we will explain the best filtering algorithm. Although there is not the "best" definition of the best filtering algorithm, however, we will describe the best to set the algorithm as its estimated value for a comparable, dynamic number, and the exchange is small. Indeed, all filtering algorithms are problematic means they do well on special types as others are useful for a small number items, some are on a larger list, some are worth duplicates. Therefore, it is difficult function to say which filter algorithm is best. But we are showing in this paper other internal filtering algorithms and compare them. In this paper we see that shortcomings of our meaning, but feel that its simplicity provides both honesty the basis for comparing and understanding the nature of the algorithm. We feel like ours rating indicates better filtering effort than most commonly used number of to compare. Studies of natural filtration difficulties are often guided in terms of reducing the number of times.

VI. Problem Definition

All the sorting algorithms are problem-specific. Each sorting algorithms work well on specific kind of problems. In this section, we described some problems and analyses which sorting algorithm is more suitable for that problem. As the problem of sorting arises frequently in computer programming and though need to be resolved in an optimizing and efficient way. While the problem is assigned it is having some size, so depending on that and more other user's choices we are proposing a method in which the user can get his problem sorted in a most optimizing manner.

VII. Optimized Orchaestrator

To dynamically optimize the sorting we have created a function which acts as our orchaestrator function. As we are working on the optimization of sorting . we have divided the whole function in few modules.

When the user enters the array to be sorted it will gets in the main method and it will get sorted choosing the best sorting technique matched with the user's requirements. We have divided the ways to be sorted on some keypoints i.e Depends on the size of the array user enters if its little larger lets say more than 1000 elements then it might not get ther stability. Always in that case user could also gets the option that if he wanted the top n sorted elements or the whole array to be sorted. We have 2 main modules of our function:

A. Main Method

In main method we are mainly processing the user's input. By getting the information from the user about the size of the array , and if the array is very large he wants the whole array to be sorted or wants to get some top n elements.

B. Router method

This is considered to be our decisive module. In this we are assigning the different sorting functions we have created in the main class depending on the input given by the user. When all the props needed by the method comes in it will assign the most optimized sorting applicable in that condition. For ex If the user needs stability and the array is very small then the most optimized condition for that situation will be bubble sort . In the above manner we have created it for all other sorting techniques.

C. Print function

This will print the final output on the user's screen.

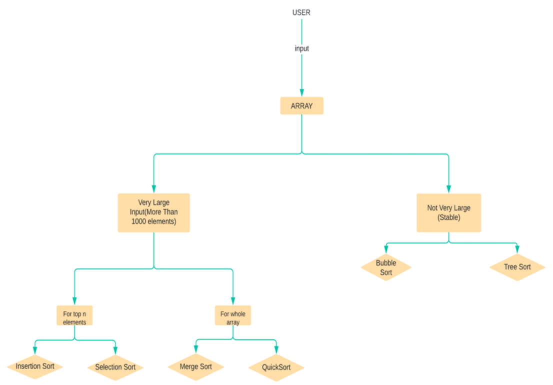

Fig. 1. Architecture of our Method

VIII. CONCLUSION

In this paper, try to summarize nearly all the sorting algorithms. Therefore, to sort a list of elements, first of all we analyzed the given problem i.e. the given problem is of which type (small numbers, large values). After that we apply the sorting algorithms but keep in mind minimum complexity, minimum comparison and maximum speed. In this paper we also discuss the advantages and disadvantages of sorting techniques to choose the best sorting algorithms for a given problem. finally, the reader with a particular problem in mind can choose the best sorting algorithm

IX. ACKNOWLEDGMENT

I cannot express enough thanks to my committee for their continued support and encouragement: Dr. Nitin Mishra.I offer my sincere appreciation for the learning opportunities provided by my committee. My completion of this project could not have been accomplished without the support of my project partners, Khushi Gupta and Kunal Singh Teotia. – thank you for allowing me to lead you and this wouldn't have been possible if you people wouldn't have had my back at stressful times.

Finally, to our caring parents who took great care of us in these tough times. Your encouragement when the times got rough are much appreciated and duly noted. It was a great comfort and relief to know that you were willing to provide management of our household activities while I completed my work. My heartfelt thanks! []

REFERENCES

[1] V. Kulalvaimozhi, M. Muthulakshmi, R. Mariselvi, G. S. Devi, and C. Rajalakshmi, "Performance analysis of sorting algorithm," *Journal of Modern Science*, vol. 7, no. 1, p. 63, 2015.

[2] A. D. Mishra and D. Garg, "Selection of best sorting algorithm," *International Journal of intelligent information Processing*, vol. 2, no. 2, pp. 363–368, 2008.

[3] H. Dannelongue and P. Tanguy, "Efficient data structures for adaptive remeshing with the fem," *Journal of computational physics*, vol. 91, no. 1, pp. 94–109, 1990.

[4] A. N. Trofimchuk, V. A. Vasyanin, and L. P. Ushakova, "Overview of methods and algorithms for constructing the shortest paths and prospects of their development," *Journal of Automation and Information Sciences*, vol. 52, no. 8, 2020.

[5] N. H. Beebe, "A complete bibliography of publications in science of computer programming," 2019.

[6] P. E. Battistella, C. G. Von Wangenheim, A. Von Wangenheim, and J. E. Martina, "Design and large-scale evaluation of educational games for teaching sorting algorithms." *Informatics in Education*, vol. 16, no. 2, pp. 141–164, 2017.

[7] B. Karsin, V. Weichert, H. Casanova, J. Iacono, and N. Sitchinava, "Analysis-driven engineering of comparison-based sorting algorithms on gpus," in *Proceedings of the 2018 International Conference on Supercomputing*, 2018, pp. 86–95.

[8] T. Bingmann, "Scalable string and suffix sorting: Algorithms, techniques, and tools," *arXiv preprint arXiv:1808.00963*, 2018.