International Research Journal of Computer Science (IRJCS)
Issue 10, Volume 5 (November 2018)
ISSN: 2393-9842
www.irjcs.com

IMPROVED SORT ALGORITHM FOR FILES MANAGEMENT

Raphael Agyo Baku*
Department of Information Technology
Federal University, Wukari, Nigeria
bakuralph@fuwukari.edu.ng

Adi Aji Useni
Department Computer Science
Federal University, Wukari, Nigeria
aadiajiuseni@gmail.com

Addas Matsunde
University Library
Federal University, Wukari, Nigeria
addasmatsunde@gmail.com

Manuscript History
Number: **IRJCS/RS/Vol.05/Issue10/NVCS10081**
Received: 08, November 2018
Final Correction: 14, November 2018
Final Accepted: 21, November 2018
Published: November 2018
Citation: Baku, Useni & Matsunde (2018). IMPROVED SORT ALGORITHM FOR FILES MANAGEMENT. IRJCS:: International Research Journal of Computer Science, Volume V, 501-506. **doi://10.26562/IRJCS.2018.NVCS10081**
Editor: Dr.A.Arul L.S, Chief Editor, IRJCS, AM Publications, India

Abstract— Sorting is one of the most common operations in computing, it has been defined as an activity which involves rearrangement of elements in either ascending or descending order. There are many sorting algorithms that have been proposed over the years, but this study focused on improvement of bubble sort algorithm. A new technique that enhanced files sorting has been proposed, structured system analysis and design was used for design and C++ platform with window 7 operating system were used for implementation of the prototype. The proposed technique, compared with the existing bubble sort, revealed 40% improvement in the time complexity of the sort algorithms. It's recommended that the technique should be deployed online for web-based files management.

Keywords— Algorithms; Bubble sort; Insertion; Time complexity; Technique;

I. INTRODUCTION

Sorting is referred as the systematic and logical way of re-arranging data and information in either ascending or descending order. There are number of sorting algorithm that have been proposed over the years, these include bubble, heap, median, insertion, merge, quick, bucket sort algorithms (Sharma, 2015). Qin (2008) pointed out that sorting is a mechanism that organizes elements of a list into a predefined order that is important for various reasons. He further opined that sorting algorithms such as Bubble, Insertion and Selection sort have quadratic time complexity that limits their use when the number of elements is too large. Aremu *et.al.* (2013) reported that to control accuracy in computation, scientists often apply sorting to actualize the goal, and in string processing, finding the longest common prefix in a set of string and the longest repeated substring in a given string is often based on sorting.

All sorting algorithms are applicable on specific kind of problems, while some are applicable on small number of elements, especially the sorting algorithm suitable for floating point numbers which fit for specific range like (0, 1). Many sorting algorithms are in existence and researchers are to find out an extensive analysis of the existing algorithms in order to reduce the time complexity. It is in the consideration of these, that this study focused on bubble sort algorithm by using a technique for files sorting in order to improve its performance and efficiency.

The specific objectives of the study include:

i. Study and analyze the time complexity of bubble sort algorithm.
ii. Design a new technique for sorting files.
iii. Compare the existing bubble sort and the new technique.

II. RELATED WORK

Awode *et.al*. (2017) reported halted complexity analysis of Bubble and Insertion sorting algorithms. Two sorting algorithms were implemented in BATLAB as shown in Table 1 and Figure 1.

Metrics	Value
c1	13
c2	7
C1	21
C2	30
LOC	27
Pvoc	17
P	43
Cd	21.70
PVol	41.23

Table 1: bubble sort implementation parameter (Awode et.al, 2017).

Table 1 described the parameter used for the simulation. The bubble sort parameter in table 1 was implemented in MATLAB and the analysis results are presented in Figure 1. The insertion sort parameters were implemented in MATLAB and the analysis results are presented.

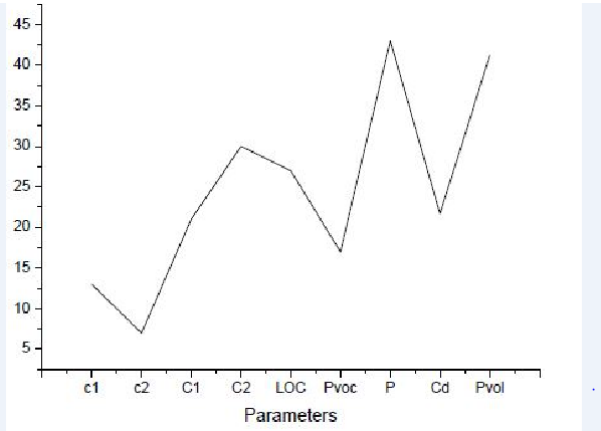

Figure 1: bubble sort algorithm analysis (Awode et.al, 2017).

Folabi (2017) presented performance evaluation of some selected sorting algorithms by the use of halted complexity metrics. The study compared three sorting algorithms and was implemented in Microsoft C# programming language, with bubble, insertion and selection sort.

Obed and Ezekiel (2015) presented magnetic Bubble sort algorithm. The paper proposed a new bubble sort algorithm which performs better than the existing bubble sort and in most cases have a run time in order of O(n) which is ideal for sorting relatively large set of data. Olabiyisi and Adetunji (2013) reported an evaluation of the critical factors affecting the efficiency of some sorting techniques. In the study, a random number generator was incorporated into the Java program written for each of five sorting techniques: Bubblesort, Quick sort, Shell sort, Merge sort, and Heap sort. The numbers generated were varied from 2000, 4000, 6000, 8000 and 10000 for each of the sorting techniques.

Yelami (2013) presented Bidirectional Bubble sort approach to improving the performance of Introsort in the worst case for large input size. The result obtained from the experiments carried out when the programs were tested on a system running windows 7 ultimate using Bloodshed dev-C++ 4.9.9.2 are presented in Table 2.

	Introsort				Improved Introsort			
Size	Comparison	Swapping	Swapping-based Assignment Operations	Total Operations	Comparison	Swapping	Swapping-based Assignment Operations	Total Operations
900	6,364	450	1,350	7,714	8,159	450	1,350	9,509
5,000	50,512	2,500	7,500	58,012	60,507	2,500	7,500	68,007
10,000	111,024	14,999	44,997	156,021	131,019	5,000	15,000	146,019
20,000	242,048	29,999	89,997	332,045	282,043	10,000	30,000	312,043
80,000	1,128,192	119,999	359,997	1,488,189	1,288,187	40,000	120,000	1,408,187

Table 2: performance of introsort and improved introsort in the worst case scenario (Yelami and Olufemi ,2013).

Oyelami (2007) proposed an improvement on performance of bubble sorting using a modified diminishing increment sorting. The proposed algorithms has fewer number of comparison and swap compared with both Batcher's odd-even sort and Bitonic sort as shown in Table 3. The result also show that as the size of the input increases, the proposed algorithm tends to be more efficient as both Batcher's odd-even and bitonic sort are not good for large values of input.

	Batcher's Odd-Even Sort		Bitonic Sort		Oyelami's Sort	
Size of Input	Number of Comparisons	Number of Swaps	Number of Comparisons	Number of Swaps	Number of Comparisons	Number of Swaps
4	5	4	6	4	5	2
8	19	12	24	14	11	4
16	63	32	80	44	23	8
32	191	80	240	128	47	16
64	543	192	672	312	219	41
128	1471	448	1792	928	191	64
256	3839	1024	4608	2368	383	128

Table 3: comparison of batchers sort, bitonic sort and Oyelami's sort performances (Oyelami, 2007).

Sharma (2015) presented a new approach to improve worst case efficiency of Bubble Sort. In the study, the proposed algorithm has been compared with normal bubble sort and insertion sort on different size of input elements in reverse order. Results obtained are shown in the Table 4.

Input size	Bubble sort	Insertion sort	Proposed Algorithm (Improved bubble sort)
	No. of comparisons	No. of comparisons	No. of comparisons
15	105	105	22
55	1485	1485	82
220	24090	24090	329
350	61075	61075	524
512	130816	130816	767
405	81810	81810	607
850	360825	360825	1274
600	179700	179700	899
750	280875	280875	1124
890	395605	395605	1334
910	413595	413595	1364
1200	719400	719400	1799

Table 4:Comparison of proposed algorithm with existing ones (Sharma, 2015)

1. Methodology

A. Development Framework

The study used system analysis and design (SSADM) methodology and analyzed /designed the model of the new system.

B. Design tool

Microsoft visio was used for drawing of the model and data flow diagram of the proposed system.

C. Implementation

C++/Qt framework. Integrated Development Environment (IDE) was used for implementing the model.

2. Design

D. Existing Bubble sort Model

Figure 2 shows an existing Bubble sort, it was designed to sort integers, it sort based with redundant comparison, it compare two adjacent data or input and swap them based on ascending order if the first element is larger or descending if the first element is smaller.

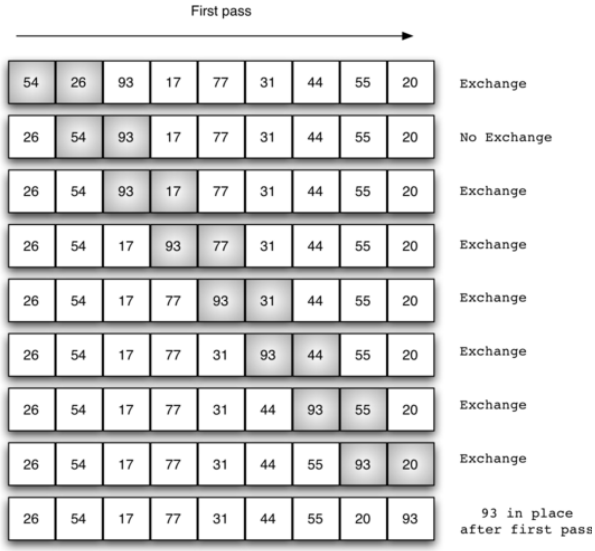

Figure 2: Existing Bubble Sort Algorithms

The bubble sort makes multiple passes through a list, it compare adjacent items and exchange those that are out of order. Each shaded items are being compared to see if they are out of order. If there are **n** items in the list, then there are **n-1** pairs of items that need to be compared on the first pass.

E. High Level Model Of The Proposed New System

The proposed system improves the existing bubble sort, a new technique is applied on the design to sort and manage files rather than integers. The new system reduces the quadratic time complexity of the algorithms and checks each step of the sorting level to avoid unnecessary loop if the preceded files are sorted in the necessary order.

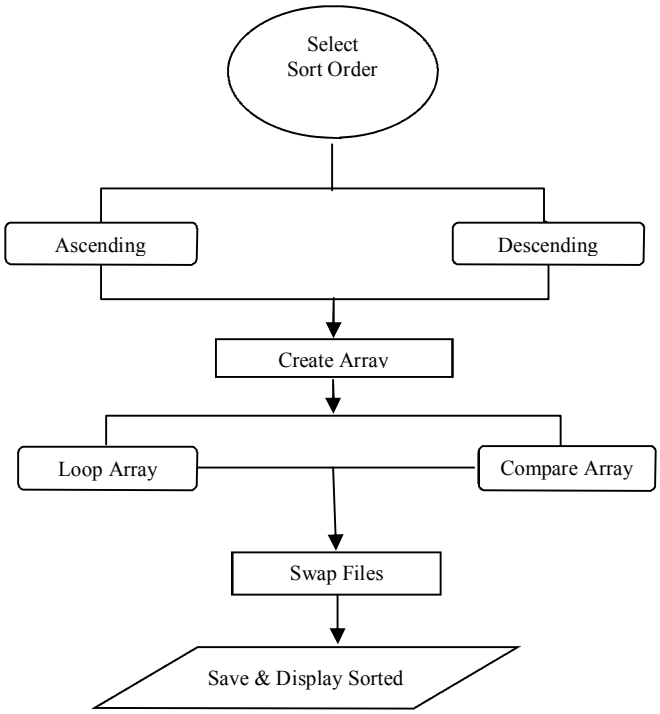

Figure 3: System Flow-Diagram

F. Result

The result of this study revealed an improved sort algorithm. The proposed improved bubble sort algorithm was able to sort files; the new system checks each step of the loop to avoid unnecessary loops especially if the preceding files are already sorted.

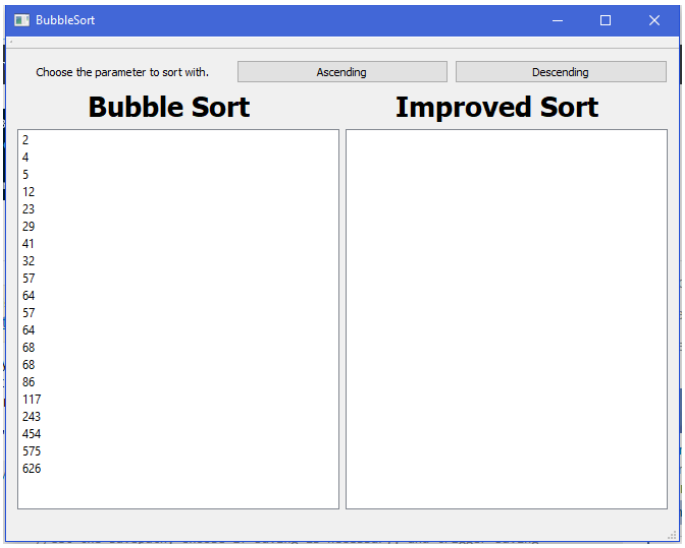

Figure 4: Output of the Improved Sort Algorithms

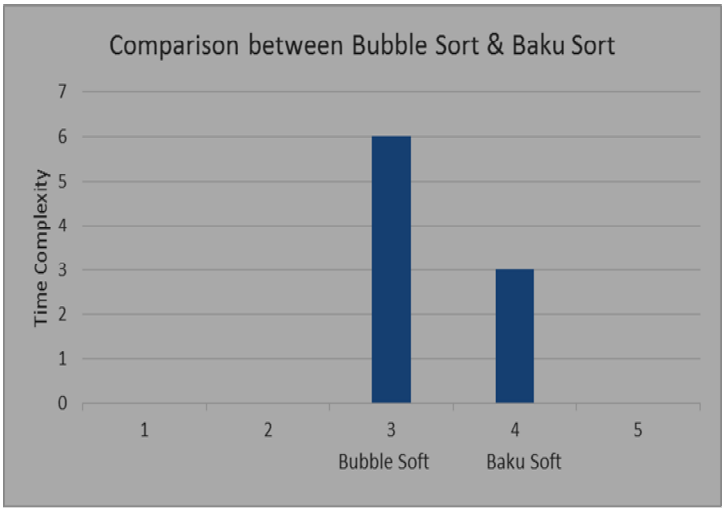

Figure 5: Comparison between Bubble Sort & Improved Baku Sort.

G. Discussions

The result of the study found a new way of sorting files which reduce the time complexity by 40% compared to the existing sort algorithm. The improved bubble sort is designed to sort files in either ascending or descending order depending on the choice of the user, the new system as captured in the result of the study, get a folder from the directory, and create an array, loop through the files, sort the files according to size in ascending or descending order. The result of the research work adds to the knowledge of the sorting algorithm.

H. Conclusion

Observation and analysis so far has clearly shown that the importance of sorting and its study cannot be over emphasized. Data needed to be organized in a specific order to promote orderliness and easy access. In every sorting, complexity and execution processes determine the efficiency of such sorting techniques. Each sorting techniques has its advantage and disadvantage to some extent, so the choice of a sorting technique depends on the order of the data, the data structure involved and the programmer's ability. However, the implementation of sorting techniques in a programming language should be dependent on how that sorting techniques effectively utilizes the memory and time in such language.

I. Recommendations

The result of the study should be applied in an organization that deals with files management such as educational institutions. It is also recommended that the application should be deployed online for web-based files management. Further research can be conducted based on online file management using different methodology.

REFERENCES

1. D.R Aremu, O.O Adesina, O.E Makinde, O. Ajibola & O.O Agbo-Ajala (2013). *A Comparative Study of Sorting Algorithms.*African Journal of Computing & ICT. Vol. 6, No.5. Pp 199-206
2. T.R Awode, D.D Olatinwo, O.Shoewu, S.O Olatinwo, O.O Omitola and Mary Adedoyin. (2017). *Halstead Complexity Analysis of Bubble and Insertion Sorting Algorithms.* The pacific Journal of Science and Technology, Vol.18 No. 1 Pp.125
3. A.O Afolabi (2017). Performance Evaluation of Some Selected Sorting Algorithms by the Use of Halstead Complexity Metrics. Transactions on Machine Learning and Artificial Intelligence,Volume 5 No 2 April (2017); pp: 9-17.
4. S.O. Olabiyisi and A.B .Adetunji. (2013). *An Evaluation of the Critical Factors Affecting the Efficiency of Some Sorting Technques.* I.J.Modern Education and Computer Science, 2013, 2, 25-33.
5. O.M. Oyelami. (2013).*Bidirectional Bubble Sort Approach to Improving the Performance of Introsort in the Worst Case for Large Input Size.* International Journal of Experimental Algorithms (IJEA), 4 (2). pp. 17-24.
6. Obed, A. and Ezekiel, M. M (2015). *Magnetic Bubble Sort Algorithm*. International Journal of Computer Applications (0975 – 8887) Volume 122 – No.21.
7. Sharma, V. (2015). A New Approach to Improve Worst Case Efficiency of Bubble Sort. International Research Journal of Computer Science (IRJCS) ISSN: 2393-9842Volume 2, Issue 6.
8. Qin, S. (2008). *Merge Sort Algorithm,* Department of Computer Sciences, Florida Institute.