Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 9, September 2015, pg.173 – 181

RESEARCH ARTICLE

Minimizing Execution Time of Bubble Sort Algorithm

Ms. Jyoti Mundra, Mr. B. L. Pal
M. Tech. Scholar, Department of CSE. , Mewar University, India
Department of Computer Science & Engg. , Mewar University, India
totlajyoti@gmail.com
Contat2bl@rediffmail.com

ABSTRACT

Sorting is an important data structure which finds its place in many real life applications. A number of sorting algorithms are in existence till date. In this paper the we have tried to improve upon execution time of the Bubble Sort algorithm by implementing the algorithm using a new algorithm .An extensive analysis has been done by us on the new algorithm and the algorithm has been compared with the traditional method of —Bubble Sort. Observations have been obtained on comparing this new approach with the existing approaches of Bubble Sort. The new algorithm was tested on random data of various ranges from small to large. It has been observed that the new approach has given efficient results in terms of execution time. Hence we have reached to the conclusion through the experimental observations that the new algorithm given in this paper is better than the traditional Bubble Sort.

Keywords: Bubble Sort, Optimized Bubble Sorting, CPU Time.

I. INTRODUCTION

Information growth rapidly in our world and to search for this information, it should be ordered in some sensible order. Many years ago, it was estimated that more than half the time on many commercial computers was spent in sorting. Fortunately this is no longer true, since sophisticated methods have been devised for organizing data, methods which do not require that the data be kept in any special order [1].
There are two basic categories of sorting methods:

1. INTERNAL SORTING: If all the data that is to be sorted can be adjusted at a time in main memory, then internal sorting methods are used.
The various internal sorting methods are : Bubble sort, Insertion sort, Quick Sort, Merge Sort, Heap Sort, Radix Sort

2. EXTERNAL SORTING: When the data to be sorted can't be accommodated in the memory at the time and some has to be kept in auxiliary memory (hard disk, floppy, tape etc) , then external sorting method are used. the most common used external sorting method is: Merge Sort

As stated in [2], sorting has been considered as a fundamental problem in the study of algorithms, that due to many reasons:

• The need to sort information is inherent in many applications.
• Algorithms often use sorting as a key subroutine.
• In algorithm design there are many essential techniques represented in the body of sorting algorithms.
• Many engineering issues come to the fore when implementing sorting algorithms.
Efficient sorting is important to optimize the use of other algorithms that require sorted lists to work correctly; it is also often in producing human-readable output. Formally, the output should satisfy two major conditions:
• The output is in non-decreasing order.
• The output is a permutation, or reordering, of the input.

Since the early beginning of computing, the sorting problem has attracted many researchers, perhaps due to the complexity of solving it efficiently. Bubble sort was analyzed as early as 1956 [5]. Many researchers considered sorting as a solved problem. Even so, useful new sorting algorithms are still being invented, for example, library sort was first published in 2004. Sorting algorithms are prevalent in introductory computer science classes, where the abundance of algorithms for the problem provides a gentle introduction to a variety of core algorithm concepts [1,13].

In [4], they classified sorting algorithms by:

• Execution time: Some algorithms takes less CPU time and some takes More CPU time.
• Number of swaps (for in-place algorithms).
• Stability: stable sorting algorithms maintain the relative order of records with equal keys (values). Ex. when there are two records R and S with the same key and with R appearing before S in the original list, R will appear before S in the sorted list.
• Usage of memory and other computer resources. Some sorting algorithms are "in place", such that only O(1) or O(log n) memory is needed beyond the items being sorted, while others need to create auxiliary locations for data to be temporarily stored.
• Whether or not they are a comparison sort. A comparison sort examines the data only by comparing two elements with a comparison operator.

In this paper we have proposed a slight variation to Bubble sort by introducing a new approach for implementing the bubble sort. We wanted to design a stable sorting algorithm which could sort Maximum Number of elements in every pass.

II. LITERATURE REVIEW

Astrachanm in 2003 [5]has investigated the origin of bubble sort and its enduring popularity despite warnings against its use by many experts.

Jehad Alnihoud and Rami Mansi in 2010 [7] have presented two new sorting algorithms i.e. enhanced Bubble Sort and Enhanced Selection Sort . ESS has O(n2) complexity, but it is faster than SS, especially if the input array is stored in secondary memory, since it performs less number of swap operations. EBS is definitely faster than BS, since BS performs O(n2) operations but EBS performs O(nlogn) operations to sort n elements.

Arora Nitin, Kumar vivek and Kumar Suresh in 2012 [13] provides a novel sorting algorithm Counting Position Sort which is based on counting position of each element in the array. This algorithm is based on counting the smaller elements in the array and fixes the position of the element.

Sultanullah Jadoon, Salman faiz Solehria, Prof. Dr. Salim Ur Rehman, Prof. Hamid Jan in 2012[8] proposed A new sort algorithm namely "An End-to-End Bi-directional Sorting (EEBS) Algorithm" is to address the shortcomings of the most popular sorting algorithms. The proposed algorithm works in two steps. In first step, the first and the last element of the array is compared. If the first element is larger than the last element, then the swapping of the elements is required. The position of the element from front end and element from the rear end of the array are stored in variables which are increased (front end) and decreased (rear end) as the algorithm progresses. In the second step, two adjacent elements from the front and rear end of the array are taken and compared. Swapping of elements is done if required according to the order. Four variables are taken which stores the position of two front elements and two rear elements to be sorted. The results of the analysis proved that EEBS is much more efficient than the other algorithms like bubble, selection and insertion sort .

Sareen Pankaj], Sareen Pankaj, In March 2013 [15]present a paper " Comparison of Sorting Algorithms (On the Basis of Average case)", International Journal of Advanced Research in Computer Science and software Engineering, in which various sorting algorithms are discussed and compared.

Abdul Wahab Muzaffar, Naveed Riaz, Juwaria Shafiq and Wasi Haider Butt In 2012 [18], has proposed a new sorting algorithm, relative Split and Concatenate sort. This algorithm is implemented and compared with some existing sorting algorithms.

Dhwaneel Trivedi, Prathmesh Trivedi, Suraj Singh,"Min-Max Select Bubble Sorting Algorithm" ,in 2013 [9] In initially "TWO" elements are placed at proper positions in each iteration, and then it uses neighborhood property to swap elements, hence it is advantageous .

Harish Rohil , Manisha In 2014 [14] introduces a new algorithm named RTBS (Run Time Bubble Sort). the user doesn't need to compare all last entered elements, but he needs only to compare the entire list with next coming element, because the list is already sorted before entering next element. It requires less execution time than existing bubble sort.

Vipul Sharma In 2015 [10] proposed a new approach to improve the worst case performance of bubble sort by reducing the number of comparisons.

Ramesh M. Patelia, Shilpan D. Vyas, Parina S. Vyas In 2015 [11] proposed enhancement algorithm has a significant improvement on reducing the number of pass iteration

III. EXISTING BUBBLE SORT ALGORITHM

The bubble sort is an exchange sort. It involves the repeated comparison and, if necessary, the exchange of adjacent elements. The elements are like bubbles in a tank of water- each seeks its own level.

Algorithm

```
Bubble_Sort ( A [ ] , N )
Step 1 : Repeat For P = 1 to N – 1 Begin
Step 2 : Repeat For J = 1 to N – P Begin
Step 3 : If ( A [ J ] < A [ J – 1 ] )
                Swap ( A [ J ] , A [ J – 1 ] )
                End For
                End For
Step 4 : Exit
```

For example, if the Bubble Sort were used on the array, 7,9 ,4,5,6,2,1,8 each pass would be like as shown in Table 1:

Pass 1	7	4	5	6	2	1	8	9
Pass 2	4	5	6	2	1	7	8	9
Pass 3	4	5	2	1	6	7	8	9
Pass 4	4	2	1	5	6	7	8	9
Pass 5	2	1	4	5	6	7	8	9
Pass 6	1	2	4	5	6	7	8	9
Pass 7	1	2	4	5	6	7	8	9

Table 1. Bubble Sort for the input values 7,9 ,4,5,6,2,1,8

With the Bubble Sort, the number of comparisons is always the same because the two for loops repeat the specified number of times whether the list is initially ordered or not. This means that the bubble sort always performs $\frac{1}{2}(n^2 - n)$ Comparisons, where n is the number of elements to be sorted.

IV. PROBLEM STATEMENT

The bubble sort always performs $\frac{1}{2}(n^2 - n)$ Comparisons. So main disadvantage of the bubble sort method is that time it requires to sort data is more. And it is highly inefficient for large data sets.

V. OBJECTIVE

The main objective of this paper is to minimize the execution time required to sort an unordered list of data. Proposed algorithm require less execution time compared to Existing bubble sort algorithm.

VI. PROPOSED OPTIMIZED BUBBLE SORT ALGORITHM

In this sorting algorithm, the sorting is done in two phases:

Phase 1: initially Leftmost bound is set at 0th position and Rightmost bound is set at last position i.e. Length of array - 1. Leftmost and rightmost elements are compared if leftmost is larger than rightmost then swapped .then leftmost bound get increased and right most bound get decreased .The main loop gets repeated "n/2" times provided "n" is the Length of array. After every iteration the distance between Leftmost bound and Rightmost bound goes on decreasing. In this array get optimized.

Phase 2: In this phase bubble sort algorithm is applied from left to right up to middle element and from right to left up to middle element. After these array get sorted.

Algorithm

```
proposed_Bubble_Sort ( A [ ] , N )
Step1:   Set Mid=N/2, S=0
Step 2 : Repeat For J= 0 to Mid-1  Begin
Step 3:  If ( A [ J ] >A [ N- J -1] )
              Swap ( A [ J ] , A [ N-J – 1 ] )
           End For
Step  4: Repeat For P = 0 to Mid-1  Begin
Step  5 : Repeat For J = 0 to Mid   Begin
Step  6: If ( A [ J ] > A [ J +1 ] )
                  Swap ( A [ J ] , A [ J +1 ] )
                  S=1
           If ( A [ N-J -1] < A [ N-J -2 ] )
                  Swap ( A [ N-J -1] , A [ N-J -2] )
                  S=1
           End For
                IF (S=0 )  Break
                Set   S=0
          End For
Step 7 : Exit
```

Data flow Representation :

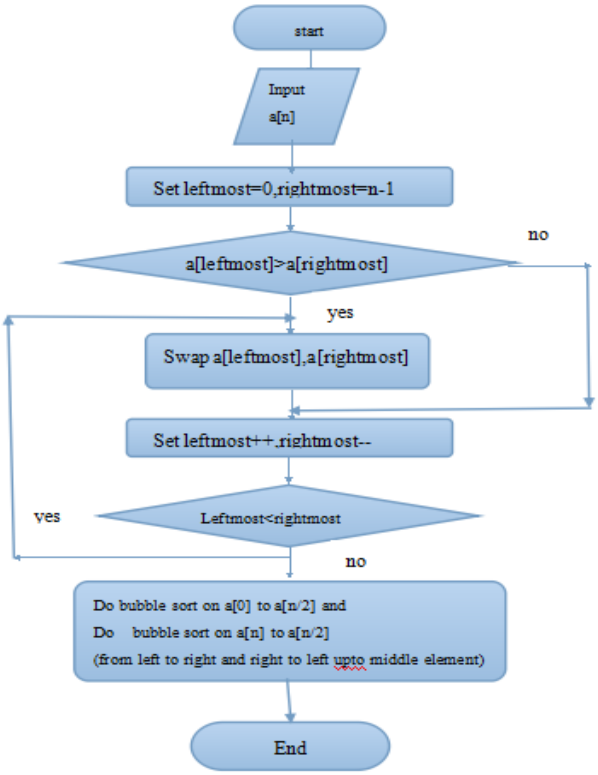

Fig 1: Data flow diagram of new bubble sort algorithm

For example, if the proposed optimized algorithm were used on the array, 7,9 ,4,5,6,2,1,8 each phase would be like as following :

Phase1 (step 1-3)	7	1	2	5	6	4	9	8
Phase2 (step 4-7) Pass 1	1	2	4	5	6	7	8	9

Table 2.proposed optimized bubble sort for the input values 7,9 ,4,5,6,2,1,8

For given example above No. Of comparisons made are only 12 which are very less.

VII. IMPLEMENTATION

In order to test this proposed algorithm for its efficiency the algorithm was implemented in C language on Turbo C++ with operating system window 7. But it was sufficient to sort less no of data item up to 15000 because of memory problem .
So we next implemented it in C language on GCC complier on Linux operating system .We tested up to 1,00,000 data items.
To calculate the execution time of both algorithms, clock() function is used.

VIII. RESULT AND ANALYSIS

Both algorithms are compared on the same elements of unordered list. In order to make a comparison of the proposed algorithms with the existing Bubble sort, a number of tests were conducted for small as well as large number of elements. Comparison is shown in tabular form as following:

No. Of Elements	Bubble sort (in sec)	Optimized Bubble sort(in sec)	Difference(in sec)
1000	0.00	0.00	0.00
2000	0.04	0.02	0.02
3000	0.05	0.025	0.025
4000	0.06	0.03	0.03
5000	0.10	0.06	0.04
6000	0.15	0.08	0.07
7000	0.20	0.11	0.09
8000	0.26	0.15	0.11
9000	0.33	0.19	0.14
10000	0.40	0.24	0.16

Table 3: Comparison of CPU time of Bubble sort and Proposed optimized Bubble sort

As shown in Table 3 above, 1000 unordered elements have been taken and sort them using proposed Bubble sort as well as Bubble sort. The elements are selected randomly using random() function. The elapsed time was same for both sorts. As the number of elements increases up to 2000, efficiency of

proposed algorithm has been shown. The elapsed time using optimized Bubble sort was less than using existing Bubble sort. The difference of elapsed time grows as number of elements increases.

Following is the graphical representation of comparison of proposed algorithm and

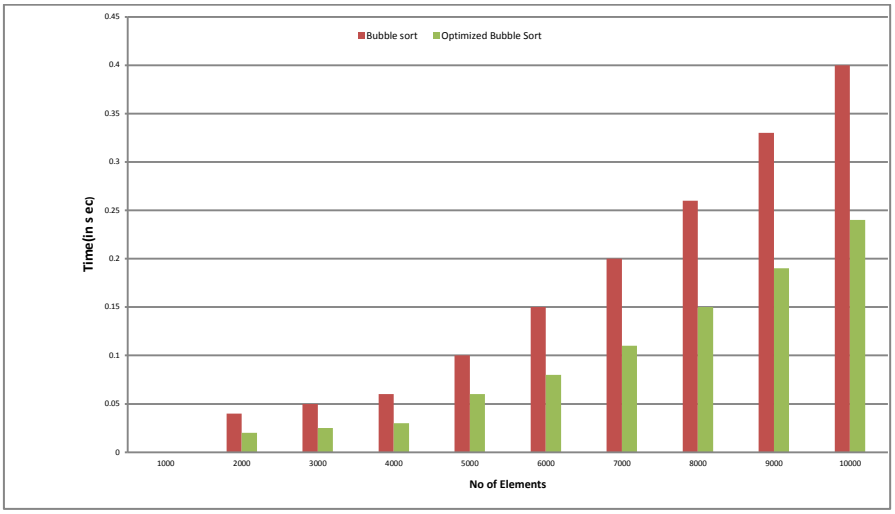

Fig.2 Graph of Number of Input vs CPU time(in sec)

In Figure 2, X-axis shows number of elements and Y-axis shows the elapsed time in milliseconds.
We tested our proposed bubble sort algorithm on large number of data also like 10k,20k,30k up to 100k(one lakh). the proposed algorithm is very efficient than existing bubble sort algorithm on these data. It is seen that Comparison for large number of elements is shown in tabular form as following:

No.Of Elements	Bubble sort (in sec)	Optimized Bubble sort (in sec)	Difference(in sec)
10k	0.40	0.24	0.16
20k	1.59	0.96	0.63
30k	3.58	2.15	1.43
40k	6.36	3.83	2.53
50k	9.9	5.97	3.93
60k	14.28	8.59	5.69
70k	19.46	11.70	7.76
80k	25.44	15.30	10.14
90k	32.21	19.41	12.8
100k	39.77	23.96	15.81

Table 4: Comparison of CPU time of Bubble sort and Proposed optimized Bubble sort

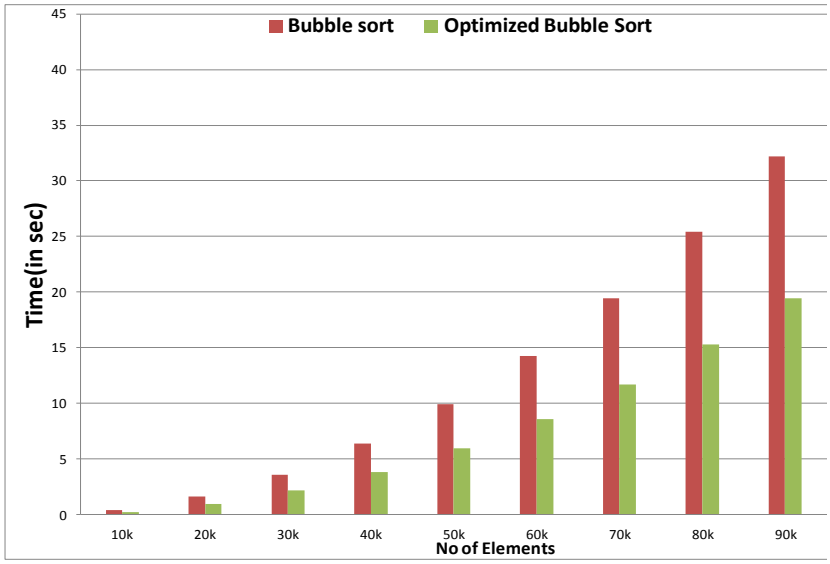

Fig.3 Graph of Number of Input vs CPU time(in sec)

It is seen that the proposed new algorithm is 40% faster than existing bubble sort algorithm on every range of data in terms of execution time.

IX. CONCLUSION

In this paper, efforts are made to point out some deficiencies in earlier work related to bubble sorting algorithms. By going through all the experimental results and their analysis it is concluded that the proposed new algorithm is very efficient than existing bubble sort algorithm .It is reducing the Execution time approx 40%. The benefits of this algorithm can be used in various areas like sorting of contact lists, Web Browsers, etc. One of the advantages is that it is a stable sort which can be used in all applications where similar valued keys are used in databases, same last name but different first name of people.

REFERENCES

[1] Kruse R., and Ryba A., Data Structures and Program Design in C++, Prentice Hall, 1999.
[2]Cormen T., Leiserson C., Rivest R., and Stein C., Introduction to Algorithms, McGraw Hill,2001
[3] Knuth, D. The Art of Computer Programming, Vol. 3: Sorting and Searching, Third edition. Addison-Wesley, 1997. ISBN 0-201-89685-0. pp. 106-110 of section.
[4] Aho A., Hopcroft J., and Ullman J., The Designand Analysis of Computer Algorithms, Addison Wesley, 1974.
[5] Astrachanm O., Bubble Sort: An Archaeological Algorithmic Analysis, Duk University, 2003.
[6] D.Sharma,V.Thapar,R.A.Ammar,S.Rajasekaran,M.Ahme d, "Efficient sorting algorithms for the cell broadband engine,"Computers and Communications,2008.ISCC 2008.IEEE Symposium.
[7] Jehad Alnihoud and Rami Mansi, "An Enhancement of Major Sorting Algorithms," The International Arab Journal of Information Technology, Vol.7, No. 1, January 2010.
[8] Sultanullah Jadoon, Salman faiz Solehria, Prof. Dr. Salim Ur Rehman, Prof. Hamid Jan, "Design and Analysis of Optimized Selection Sort Algorithm," International Journal of Electric and computer sciences IJECS-IJENS VOL. 11, No. 01 pp. 16- 22.
[9] Dhwaneel Trivedi, Prathmesh Trivedi, Suraj Singh,"Min-Max Select Bubble Sorting Algorithm" ,International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868 Foundation of Computer Science FCS, New York, USA International Conference & workshop on Advanced Computing

2013 (ICWAC 2013) – www.ijais.org .
[10] Vipul Sharma "A New Approach to Improve Worst Case Efficiency of Bubble Sort" International Research Journal of Computer Science (IRJCS) ISSN: 2393-9842 Issue 6, Volume 2 (June 2015) www.irjcs.com
[11] Ramesh M. Patelia, Shilpan D. Vyas, Parina S. Vyas "An Analysis and Design of Optimized Bubble Sort Algorithm".IJRIT International Journal of Research in Information Technology, Volume 3, Issue 1, January 2015, Pg. 65-68
[12] Data Structures by Seymour Lipschutz, Schaum's outlines, The MacGraw Hill Companies.
[12] Levitin A., "Introduction to the Design & Analysis of Algorithms", 2nd Ed. Pearson Educational, 2007.
[13] Arora Nitin, Kumar vivek and Kumar Suresh. "A Novel Sorting Algorithm and Comparison with Bubble Sort and Insertion Sort," International Journal of Computer Applications (0975-8887) vol. 45, No. 1, May 2012.
[14] Harish Rohil , Manisha."Run Time Bubble Sort – An Enhancement of Bubble Sort". International Journal of Computer Trends and Technology (IJCTT) V14(1):36-38, Aug 2014. ISSN:2231-2803. www.ijcttjournal.org. Published by Seventh Sense Research Group.
[15] Sareen Pankaj, " Comparison of Sorting Algorithms (On the Basis of Average case)", International Journal of Advanced Research in Computer Science and software Engineering ISSn: 2277128x, volume 3, Issue 3, March 2013, pp. 522-532
[16] heim.ifi.uio.no/~arnem/sorting/ARLStable2006 /NIK-2006-.pdf.
[17] www.stanford.edu/~poulson/CME194/papers/splitting.pdf.
[18] Abdul Wahab Muzaffar, Naveed Riaz, Juwaria Shafiq and Wasi Haider Butt, " Relative Sp lit and Concatenate Sort (RSCS-VI)", International Journal of Computer Theory and Engineering vol. 4, No. 2, April 2012.
[19] www.liacs.nl/~jvrijn/ds2012/assests/stacksorting.pdf.

