# Run Time Bubble Sort – An Enhancement of Bubble Sort

Harish Rohil [#1], Manisha [#2]

[#1] *Assistant Professor, Department of Computer Science and Applications, Chaudhary Devi Lal University, Sirsa-125055, Haryana, India*

[#2] *M. Tech. Scholar, Department of Computer Science and Applications, Chaudhary Devi Lal University, Sirsa-125055, Haryana, India*

***Abstract*- In many real applications, sorting plays an important role as it makes easy to handle the data by arranging it in ascending or descending order, according to requirements. Bubble sort is most common sorting algorithm, but not efficient for many applications. There are number of other sorting algorithms and still, research is going for new sorting algorithms to improve the complexities of existing algorithms whether it is space complexity or it is time complexity. This paper introduces a new algorithm named RTBS (Run Time Bubble Sort). This sorting technique is very simple to implement. It is helpful to reduce the elapsed time when to sort the data. RTBS is compared with original bubble sort algorithm. This sorting technique is tested for average case, best case and the worst case.**

***Keywords- Sorting algorithm, Bubble sort.***

## I. INTRODUCTION

An algorithm is a finite set of steps defining the solution of a particular problem. An algorithm can be expressed in English like language, called pseudo code, in a programming language, or in the form of a flowchart. Every algorithm must satisfy the following criteria:

*A. Input*

There are zero or more values which are externally supplied.

*B. Output*

At least one value is produced**.**

*C. Definiteness*

Each step must be clear and unambiguous.

In general, sorting means rearrangement of data according to a defined pattern. The task of sorting algorithm is to transform the original unsorted sequence to sorted sequence. Different sorts are classified in different categories: complexity, memory usage, stability, comparison/non- comparison etc.

*1) Complexity:* There are mainly two types of complexity: **Time Complexity** is the running time of the program as a function of the size of input. **Space Complexity** is the amount of computer memory required during the program execution, as a function of the input size. There are three cases when measuring the complexity of algorithms: **Best case** if the resources are at least used. **Average case** if the resources are at average used and **Worst case** if the resources are at most used.

2) *Memory Usage:* Computer scientists differentiate between internal and external sorting algorithms. While external ones need a large amount of extra memory, the internal algorithms manipulate the original array and only needs a few byte for stack and temporary variables. With the exception of merge sort, almost basic sorting algorithms belong to internal sorting algorithms.

3) *Stability:* Stable algorithms maintain the relative order of records with equal sort key values. For example, whenever there are two records X and Y with the same key and with X appearing before Y in the original list, X will appear before Y in the sorted list. Bubble sort, insertion sort, Merge sort, Bucket sort Radix sort-LSD are stable algorithms, while Selection sort, Heap sort, Quick sort, Shell sort, Radix sort- MSD are unstable sorting algorithms.

*4) Comparison/non comparison sorts:* A comparison sort reads the list elements through a single abstract comparison operation (like greater than or equal to). This determines which of two elements being compared should occur first in the final sorted list. There are many used in comparison sorts:

**Exchanging:** used by Bubble sort

**Selection:** used by Selection sort and Heap sort

**Insertion:** used by Insertion and Shell sort

**Merging:** used by Merge sort

**Partitioning:** used by Quick sort

Non- comparison sorts include:

**Bucket sort:** examines bit of keys

**Radix sort:** examines individual bit of keys.

Comparison sorts being more popular.

## II. RELATED WORK

Astrachanm has investigated the origin of bubble sort and its enduring popularity despite warnings against its use by many experts [1].

Jehad Alnihoud and Rami Mansi have presented two new sorting algorithms i.e. Enhanced Bubble Sort and Enhanced

Selection Sort [2]. ESS has $O(n^2)$ 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 $O(n^2)$ operations but EBS performs O(nlogn) operations to sort n elements.

The authors have tried to improve upon the Bubble Sort technique in [3], by implementing the algorithm using a new approach of implementation. The comparison is done with traditional Bubble sort as well as Bi-directional Bubble sort. It is observed that Bi-directional Bubble sort algorithm is the best algorithm to be used with regard to worst case analysis where the data elements are in reverse order and the proposed algorithm can be very effective for the small as well as large data files.

[4] 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.

An enhancement of quick sort is presented in [6]. This sorting algorithm is named SMS (Scan, Move and Sort) algorithm.

A new sort algorithm namely "An End-to-End Bi-directional Sorting (EEBS) Algorithm is proposed in [7] to address the shortcomings of the most popular sorting algorithms. The proposed algorithm is based on bubble sort as its working in the second step of the operation is somewhat, similar to the bubble sort. 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 having $O(n^2)$ complexity, like bubble, selection and insertion sort.

A new sorting algorithm is presented in [8], namely, "Optimized Selection Sort Algorithm (OSSA)". OSSA are designed to perform sorting quickly and more effectively as compared to the existing version of Selection sort. Some key ideas are given in [9]. Some enhancements are suggested for insertion sort.

The sorting algorithm "Sorting in Linear Time?" is presented in [10]. With this algorithm, the sorting is performed in two phases. In the first phase, the size of the numbers is reduced using radix sort techniques. In second phase, merge sort is performed on shortened numbers.

In [11], the author has investigated the complexity values researchers have obtained and observed that there is scope for fine tuning in present context. They aim to provide a useful and comprehensive note to researcher about how complexity aspects of sorting algorithms can be best analyzed.

In [15], various sorting algorithms are discussed and compared.

[16] demonstrates how an unstable in-place algorithm sorting algorithm, the ALR algorithm, can be made stable by temporary changing the sorting keys during the recursion.

[17] proposes a novel parallel sorting algorithm based on exact splitting that combines excellent scaling behavior with universal applicability.

In [18], author has proposed a new sorting algorithm, relative Split and Concatenate sort. This algorithm is implemented and compared with some existing sorting algorithms.

In [19], the author uses the stacks to solve the sorting problem. Two stack based sorting algorithms are introduced. The first is based upon sorting by insertion technique, whereas other is based upon sorting by exchange technique.

## III. PROPOSED WORK

### A. *Concept*

The proposed algorithm for this research paper is RTBS (Run Time Bubble Sort). In this sorting technique, the sorting is started as the user gives first two elements. The first two elements are sorted before user gives the third element. The first three elements are sorted before user gives the fourth element and so on up to last element to be sorted. In fact, the Run Time Bubble Sort is an enhancement to the bubble sort algorithm in decreasing the time for traversing the elements from beginning. RTBS is a comparison sort with the property of stability.

### B. *Algorithm*

RTBS(DATA,N)

Step I. Input DATA[0]

Step II. Repeat steps III and IV for k=1 to N-1

Step III. Set PTR=1

Step IV. Repeat while PTR<=N-K

a. Input DATA[PTR]
b. If DATA[PTR-1]>DATA[PTR], then
c. Swap DATA[PTR-1] and DATA[PTR]
d. Set PTR=PTR+1

Step V. Exit

### C. *Experimental Results*

In order to test this proposed algorithm for its efficiency the algorithm was implemented in C language on core i3 machine with operating system window 8.

To calculate the execution time of both algorithms, clock() function is used. 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:

*TABLE I*

*Elapsed Time for the Proposed RTBS algorithm and the Existing Bubble Sort Algorithm*

| No. of elements | RTBS (in ms) | Bubble sort (in ms) |
|---|---|---|
| 10 | 40 | 40 |
| 100 | 39 | 52 |
| 1000 | 36 | 51 |
| 10000 | 164 | 191 |

As shown in Table 1 above, 10 unordered elements have been taken and sort them using proposed RTBS 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 upto 100, efficiency of proposed algorithm has been shown. The elapsed time using Run Time 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 RTBS algorithm and Bubble sort.

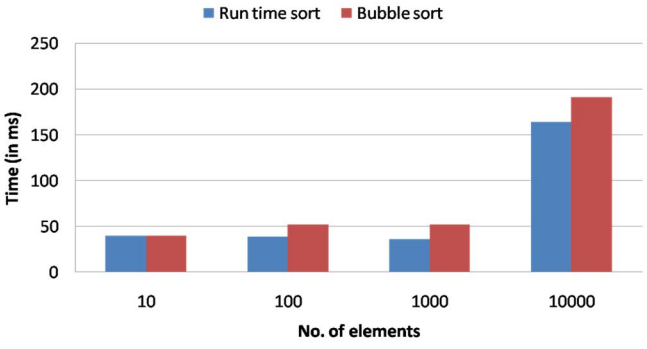


*Figure 1 Comparison of the Proposed RTBS algorithm and the Existing Bubble Sort Algorithm based on the Elapsed Time*

In Figure 1, X-axis shows number of elements and Y-axis shows the elapsed time in milliseconds.

### *D. Analysis*

When RTBS algorithm is compared with bubble sort, the only difference is that 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.

## IV. CONCLUSION

In this paper, efforts are made to point out some deficiencies in earlier work related to sorting algorithms. By going through all the experimental results and their analysis it is concluded that the proposed algorithm is efficient. In all existing algorithms, first complete list is entered, then the list is processed for sorting, but in case of proposed approach, the list is sorted simultaneously. The proposed sorting technique saves the time for traversing the list after entering all the elements, as it sorts all elements before entering any new. Although the complexity is same as the other algorithm being compared i.e. $O(n^2)$, but the performance is exceptionally well. RTBS works well when the list of numbers to be sorted is large.

## REFERENCES


[1] Astrachanm O.,*Bubble sort: An Archaeological Algorithm Analysis,* Duk University, 2003.

[2] Jehad Alnihoud and Rami Mansi, "An Enhancement of Major Sorting Algorithms," *The International Arab Journal of Information Technology,* Vol.7, No. 1, January 2010.

[3] V. Mansotra and Kr. Sourabh, "Implementing Bubble Sort Using a New Approach," *in proceedings of 5th National Conference;INDIACom-2011.*

[4] 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.*

[5] Dmitri Mihhailov, Valery Sklyarov, Iouliia Skliarova and Alexander Sudnitson, "Optimization or Recursive Sorting Algorithms for Implementation in Hardware," *22nd International conference on Microelectronics (ICM 2010).*

[6] Rami Mansi, "Enhanced Quicksort Algorithm ," *The International Arab Journal of Information Technology, Vol.7, No. 2, April 2010.*

[7] Kapur Eshan, Kumar Parveen and Gupta Sahil,"Proposal Of A Two Way Sorting Algorithm And Performance Comparison With Existing Algorithms," *International Journal of Computer Science, Engineering and Applications (IJCSEA) vol. 2, No. 3, June 2012.*

[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] source www.softpanorana.org/Algorhms/Sorting/ insertion_sort shtml.

[10] Source:-www.drdobbs.com/architecture-and-design/the- fastest-sorting-algorithm/184404062.

[11] Parag Bhalchandra, Nilesh Deshmukh, Sakharam Lokhande and Santosh Phulari, "A Comprehensive Note On Complexity Issues in Sorting Algorithms," *Advance in Computational Research, ISSN: 0975-3273,* Volume 1, Issue 2, 2009 pp. 01-09.

[12] Shahzad B. and Afzal M., "Enhanced Shell Sorting Algorithm," *Computer Journal of Enformatika,* vol.21, no. 6, pp. 66-70, 2007.

[13] Shell D, "A High Speed Sorting Procedure," *Computer Journal of Communications of the ACM* vol. 2, No. 7 pp. 30-32, 1959.

[14] 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.

[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.