IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Volume-5, Issue-6 E-ISSN: 2347-2693

An Enhancement of Bubble Sorting Algorithm

Harsh N. Nankani[1*], Mukesh Bhandari[2]

[1*]IT department, Vadodara Institute of Engineering, Gujarat Technological University, Vadodara, India
[2]CE/IT department, Vadodara Institute of Engineering, Gujarat Technological University, Vadodara, Gujarat, India

**Corresponding Author: harshnankani196@gmail.com, Mobile no.: +919033622478*

Available online at: www.ijcseonline.org

Received: 20/May/2017, Revised: 03/Jun/2017, Accepted: 20/Jun/2017, Published: 30/Jun/2017

Abstract: Sorting is an important technique of data structure which finds its place in many real-life applications. There are various sorting algorithms are in existence till date. In this paper, we have tried to improve upon execution time of the Bubble Sort algorithm by implementing the algorithm using an enhancement of it. 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 proposed approach was tested for Average Case analysis, Best Case analysis and Worst case analysis. It has been analysed that the new approach has given very good results on Average Case and Worst Case analysis. The new approach 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— *Sorting, Bubble sort*

I. INTRODUCTION

An algorithm is a finite set of instructions defining the solution of a particular problem. An algorithm can be expressed in simple language what we call a pseudo code, in a programming language, or in the form of a flowchart.

Every algorithm must satisfy the following criteria:

A. Input - Zero or more values, externally supplied.
B. Output - At least one value must be produced**.**
C. Definiteness – It should be clear and unambiguous.

Information growth rapidly in this world and to search this info, it should be ordered in meaningful manner. Earlier, it was estimated that more than half the time on many commercial machines were spent for 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].

In computer science, a sorting algorithm is an efficient algorithm which perform an important task that puts elements of a list in a certain order or arrange a collection of items into a particular order. Sorting data has been developed to arrange the array values in various ways for a database. For instance, sorting will order an array of numbers from lowest to highest or from highest to lowest, or arrange an array of strings into alphabetical order. Typically, it sorts an array into increasing or decreasing order. Most simple sorting algorithms involve two steps which are compare two items and swap two items or copy one item. It continues executing over and over until the data is sorted.

Classification of sorting algorithm:

- System complexity of computational.
- Computational complexity in terms of number of swaps.
- Memory usage is also a factor in classify the sorting algorithms.

The different cases that are popular in sorting algorithms are:

- O(n) is fair, the graph is increasing in the smooth path.
- O(n log n): this is considered as efficient, because it shows the slower pace increase in the graph as we increase the size of array or data.
- O(n square) : this is inefficient because if we input the larger data the graph shows the significant increase

In mathematics, computing, linguistics, and related disciplines, an **algorithm** is a finite list of well defined instructions for accomplishing some task that, given an initial state, will proceed through a well-defined series of successive states, possibly eventually terminating in an end-state. No generally accepted *formal* definition of "algorithm" exists yet. We can, however, derive clues to the issues

involved and an informal meaning of the word from the following quotation given by **Boolos and Jeffrey (1974, 1999) [2]**: "No human being can write fast enough, or long enough, or small enough to list all members of an innumerably infinite set by writing out their names, one after another, in some notation. But humans can do something equally useful, in the case of certain innumerably infinite sets: They can give **explicit instructions for determining the nth member of the set**, for arbitrary finite n. Such instructions are to be given quite explicitly, in a form in which **they could be followed by a computing machine**, or by a **human who is capable of carrying out only very elementary operations on symbols**".

Efficient sorting is important to optimize the use of other algorithms (such as search and merge algorithms) that require sorted lists to work correctly; it is also often useful for canonicalizing data and for producing human-readable output. More formally, the output must satisfy two conditions:
1. The output is in nondecreasing order each element is no smaller than the previous element according to the desired total order;
2. The output is a permutation, or reordering of the input. Since a significant portion of commercial data processing involves sorting large quantity of data, efficient sorting algorithms are of considerable economic importance. There is a good collection of algorithms on sortings techniques categorized under their execution behaviour which is known as complexity. Some algorithms like Bubble Sort, Selection Sort, Insertion Sort have complexity O(n^2) where as other algorithms like Quick Sort, Heap Sort have complexty O(nlogn).

II. RELATED WORK

The bubble sort is the oldest and simplest sorting method in use. Unfortunately, it's also the slowest. Thebubble sort works by comparing each item in the list with the item next to it, and swapping them if required. The algorithm repeats this process until it makes a pass all the way through the list without swapping any items (in other words, all items are in the correct order). This causes larger values to "bubble" to the end of the list while smaller values "sink" towards the beginning of the list. The total number of comparisons, is (n - 1) + (n - 2)...(2) + (1) = $n(n - 1)/2$ or $O(n2)$. The bubble sort is generally considered to be the most inefficient sorting algorithm in common usage. Under best-case conditions (the list is already sorted), the bubble sort can approach a constant $O(n)$ level of complexity. General-case is an abysmal $O(n2)$. While the insertion, selection, and shell sorts also have $O(n2)$ complexities, they are significantly more efficient than the bubble sort**. [2].**

Don Knuth, in his famous *The Art of Computer Programming*, concluded that "the bubble sort seems to have nothing to recommend it, except a catchy name and the fact that it leads to some interesting theoretical problems", some of which he discusses therein. Bubble sort is asymptotically equivalent in running time to insertion sort in the worst case, but the two algorithms differ greatly in the number of swaps necessary. Insertion sort needs only O(n) operations if the list is already sorted, whereas naïve implementations of bubble sort (like the pseudocode below) require O(n^2) operations. (This can be reduced to O(n) if code is added to stop the outer loop when the inner loop performs no swaps.) **[3].** Owen says that Bubble sort's prime virtue is that it is easy to implement, but whether it is actually easier to implement than insertion or selection sort is arguable **[5].** For example, in **[6]** we find: **—**The bubble sort is worse than selection sort for a jumbled array—it will require many more component exchanges—but it's just as good as insertion sort for a pretty well-ordered array. More important, it's usually the easiest one to write correctly. Authors have tried to bring the Bubble Sort closer to other sorts by using a new variation.

Astrachanm in 2003 **[8]** 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 **[9]** have presented two new sorting algorithms i.e. enhanced Bubble Sort and Enhanced Selection Sort. 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.

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 i = 1 to n – 1 Begin
Step 2: Repeat For j = 1 to n – i Begin
Step 3: If (A[ j ] > A[ j + 1 ])
            Swap (A[ j ] , A[ j + 1 ])
            End For
            End For
Step 4: Exit
```

IV. PROPOSED WORK

Algorithm

```
proposed_Bubble_Sort(A[ ] , n)
Step1: Set c=0
```



```
Step 2: Repeat For i= 0 to n – 1 Begin
Step 3: Repeat For j = 0 to n – 2 Begin
Step 4: If (A[ j ] > A[ j + 1])
          Set c=1
            If(j != n – 2 && A[ j + 1 ] > A[ j + 2 ])
                  Swap (A[ j ] , A[ j + 2 ])
            Else if(j != n – 2 && A[ j ] > A[ j + 2 ])
                  Swap (A[ j ] , A[ j + 1 ], A[ j + 2])
            Else
                  Swap (A[ j ] , A[ j + 1 ])
          End For
          If(c==0)
           Set i = n
          End For
Step 7: Exit
```

V. CONCLUSION

In this paper, efforts are made to find out some deficiencies in earlier work related to bubble sorting algorithms. By going through all the experimental results and their analysis one can easily conclude that the proposed algorithm is better for the data elements generated which are randomly ordered. In an existing algorithm, first complete list is entered, then the list is processed for sorting, by comparing each element with its successive element but in case of proposed approach, in addition, the list is sorted by comparing each element also with its successive element's immediate element. The proposed algorithm can process three elements in each iteration. The proposed algorithm saves the time for traversing and comparing the list after obtaining all the elements.

VI. REFERENCES

[1] Kruse R., and Ryba A., Data Structures and Program Design in C++, Prentice Hall, 1999.

[2]Boolos, George & Jeffrey, Richard (1974, 1980, 1989, 1999), *Computability and Logic* (4th ed.), Cambridge University Press, London, ISBN 0-521-20402-X: cf. Chapter 3 *Turing machines* where they discuss "certain enumerable sets not effectively (mechanically) enumerable".

[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] Cormen T., Leiserson C., Rivest R., and Stein C., Introduction to Algorithms, McGraw Hill,2001

[5] Owen Astrachan Bubble Sort: An Archaeological Algorithmic Analysis, *SIGCSE '03*, February 19-23, Reno, Nevada, USA. Copyright 2003 ACM 1-58113-648-X/03/0002.

[6] Cooper, D. *Oh My! Modula-2!* W.W. Norton, 1990.

[7] Aho A., Hopcroft J., and Ullman J., The Design and Analysis of Computer Algorithms, Addison Wesley, 1974.

[8] Astrachanm O., Bubble Sort: An Archaeological Algorithmic Analysis, Duk University, 2003.

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

[10] Knuth, D. *The Art of Computer Programming: Sorting and Searching*, 2 ed., vol. 3. Addison-Wesley, 1998.

[11] Iverson, K. *A Programming Language*. John Wiley,1962.

[12] http://linux.wku.edu/~lamonml/algor/sort/bubble.html

Authors Profile

Mr. Harsh N Nankani pursued Diploma in Information technology from Polytechnic of Maharaja Sayajirao University of Baroda, India in year 2015. He is currently pursuing Bachelor of Information technology from Vadodara Institute of Engineering of Gujarat Technological University, India since 2015. He is a member of Indian Society of Technical Education(ISTE) since 2015. His main research work focuses on Sorting and Searching algorithm.

Mr. Mukesh Bhandari pursued Bachelor of Technology in Information Technology from Graphic Era University of Dehradun, India in 2008 and Master of Technology in Computer Engineering from Government Engineering College of Rajasthan Technical University, India in 2011. He is currently working as an Assistant Professor in Department of CE and IT of Vadodara Institute of Engineering, Gujarat Technological University, India. He is a member of member of ISTE, CSI and IEEE computer society. He has published more than 13 research papers. His main research work focuses on Algorithms, Artificial Intelligence, Deep learning. He has 6 years of teaching experience, 2 years of research experience and 1.5 years of industry experience.

