Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Engineering

Elixir Comp. Engg. 78 (2015) 29978-29982

ISSN: 2229-712X

Both ended sorting algorithm & performance comparison with existing algorithm

Anuradha Brijwal[1], Arpit Goel[1], Anubhooti Papola[2] and Jitendra Kumar Gupta[3]

[1]Department of Computer Science & Engineering, Faculty of Technology, University Campus, Dehradun Uttarakhand, India.

[2]Department of Computer Science & Engineering, Uttarakhand Technical University, Dehradun Uttarakhand, India,

[3]Department of Computer Science & Engineering, GRD Institute of Technology, Dehradun Uttarakhand, India.

ARTICLE INFO

Article history:

Received: 3 June 2014;

Received in revised form:

19 November 2014;

Accepted: 29 November 2014;

Keywords

Algorithm,

Bubble Sort,

Selection Sort and Both-Ended Sorting,

Comparison.

ABSTRACT

One of the basic areas of the computer science is Data Structure. Sorting is an important issue in Data Structure which creates the sequence of the list of items. Although numbers of sorting algorithms are available, it is all the more necessary to select the best sorting algorithm. Therefore, sorting problem has attracted a great deal of research as sorting technique is very often used in a large variety of important applications so as to arrange the data in ascending or descending order. This paper presents a Both Ended Sorting Algorithm which is faster or better than the bubble sort& others algorithm. After having studied various sorting algorithms; I came to the conclusion that there is no such sorting algorithm which works on the basis of both end comparison right end as well as left end. The new algorithm so is then analysed, implemented & tested. The test results obtained are then presented and compared with the traditional Sorting Algorithm. Worst case complexity is also improved as a compare to bubble sort.

Introduction

Algorithms play a vital and key role to solve the computational problems, informally it is a well-defined computational procedure which takes input and produces output [6].Algorithm is a kind of tool or a sequence of steps so as to solve the computational problems. The existence of algorithms goes way back as they were in existence even before the existence of computers [9].

There are various methodologies and techniques based on which kinds of algorithms are designed [11]. Out of all these problem solving algorithms, let us talk about the sorting algorithms. So far as sorting is concern, it is required to arrange a sequence of numbers into a given order, generally non-decreasing. The sorting problem is countered with quite often in practice and it acts as a productive ground for the introduction of many standardized design techniques and analysis tools [13].

If the given input sequence is (89, 51, 31, 71, 3, 68), then the output sequence returned by a sorting algorithm will be (3, 31, 51, 68, 71, 89).

In order to be a good programmer, an excellent programming practice is required to be done & this can only be possible if we know about related topic theoretically as well as practically [10]. In initial stage when students learn to make program with array, they do exercise relevant to selection sort, and then with the help of required coding convert into code on their own. Many sorting algorithms we did in lab by the help of instructor and over their instructor work was always to check students who grabbed the code off the web but effective coding can be possible only when we do our own.

Sorting is a data structure operation, which is helpful for making searching and arranging of element or record. Here arrangement of sorting involves either into ascending or descending order. Everything in this world has some merits and demerits, some sorting algorithms are problem specific which means that they work well on some specific problem not on all the problems. It saves time and help searching data very quickly. Sorting algorithm performance varies on which type of data being sorted, it is not easier to say that which one algorithm is better than another. Here, also performance of different algorithm is in accordance to the data being sorted. Examples of some common sorting algorithms are the exchange or bubble sort, the selection sort, the insertion sort and the quick sort [15].

Bubble sort is a basic sorting algorithm that performs the sorting operation by iterative comparing the adjacent pair of the given data items and swaps the items if their order is reversed [15]. There is a repetition of the passes through the list is sorted and no more swaps are then required[15]. It is a simple algorithm but it is not much efficient when the given input data set is large.

But sometimes question arises in front of us, whether there any way through this sorting can be more effective and how to convert that algorithm into code. Then demonstrate a modification of this algorithm, and finally to assign the coding modification as a programming. This paper suggests one simple modification of sorting algorithm: Double-Ended Bubble Sort. One can argue as to whether the use of double sort for these small array partitions will provide improvement to this critical algorithm.

Therefore, to understand important concepts and programming practice, a good programming exercise plays a crucial role i.e. for using double-ended bubble sort in place of normal bubble sorting technique that raises the sorting skills. Well, there are two cases that occurred at the time of making double sorting code for bubble sort, when the size is odd or even.

An effort is done in positive direction and realizes coding technique for double sorting offer great improvements speed up to 25% to 35% over the single bubble sorting [15].

Tele:

E-mail addresses: meetanubrijwal01@gmail.com

Bubble Sort

Bubble sort is a basic sorting algorithm that performs the sorting operation by iterative comparing the adjacent pair of the given data items and swaps the items if their order is reversed [10, 20].There is a repetition of the passes through the list is sorted and no more swaps are needed. It is a simple algorithm but it lacks in efficiency when the given input data set is large.

The worst case as well as average case complexity of bubble sort is $O(n^2)$, where n represents the total number of items in the given array to be sorted. When bubble sort is compared with other sorting algorithms, the result indicates that there are many of such algorithms which perform better in their worst case. That's why, it is not considered as a best sorting algorithm. Bubble Sort has the best performance i.e. complexity of O(n) in case of an already sorted list.

The algorithm for bubble sort having *ARRAY* as an array with N elements is as follows:

```
BUBBLE (ARRAY, N)
for(i=1 to N-1)
{
PTR = 1
while(PTR<= N-i)
{
if (ARRAY[PTR] >ARRAY[PTR+1])
{
Temp = ARRAY[PTR]
ARRAY[PTR]=ARRAY[PTR+1]
ARRAY[PTR]=Temp
}
Set PTR = PTR+1
}
}
```

Selection Sort

The selection sort works by selecting the smallest unsorted item remaining in the list, and then swapping it with the item in the next position to be filled. The selection sort has a complexity of $O(n^2)$ [14].

The worst case as well as average case complexity of Selection sort is $O(n^2)$, where n represents the total number of items in the given array to be sorted.

The selection sort is the unwanted stepchild of the n^2 sorts. It yields a 60% performance improvement over the bubble sort, but the insertion sort is over twice as fast as the bubble sort and is just as easy to implement as the selection sort. In short, there is not really any reason to use the selection sort-use the insertion sort instead [15].

The algorithm for selection sort having *ARRAY* as an array with N elements is as follows:

```
SELECTION (ARRAY, N)
for(i=1 to N-1)
{
Min = ARRAY[i]
for (k = i+1 to N)
{
if (min>ARRAY [k])
{
    Min = A[k]
    Loc = k
}
}
Temp = ARRAY [Loc]
ARRAY [Loc]=ARRAY [i]
ARRAY [i]=Temp
}
```

Both ended Sorting Algorithm

Introduction

Various authors had made continuous attempts for increasing the efficiency and performance of the sorting process. The proposed algorithm is based on bubble sort.

In this paper algorithm works in two phases:

1. In first phase, one element from the front end and one element from the rear end of the array is compared. If the front position element is greater than the rear position element, then swap the elements. The position of the element from front end and element from the rear end of the array are stored invariables which are increased (front end) and decreased (rear end) as the algorithm progresses.
2. In the second phase, two consecutive elements from the front and rear end of the array are taken and both elements are compared. Replacing of elements is done if required according to the order. Here four variables are taken which stores the position of two rights elements and two left elements which are to be sorted.

Algorithm

```
BESA (DATA, n)
i = 1;
j = MAX;
while(i< j)
{
    if(DATA(i) > DATA(j))
    {
        Temp← DATA[i]
DATA[i]←DATA[j]
DATA[j]← Temp
    }
    i++;
    j++;
}
for (i←1 to n/2 && FLAG)
{
FLAG←0
for (j←i to n-i)
{
if(DATA[j]>DATA[j+1])
{
Temp←DATA[j]
DATA[j]←DATA[j+1]
DATA[j+1]← Temp
FLAG←1
}
if(DATA[n-j]>DATA[n-j+1])
{
Temp←DATA[n-j]
DATA[n-j]←DATA[n-j+1]
DATA[n-j+1]← Temp
FLAG←1
}
}
}
```

Complexity Analysis

The general working of the proposed algorithm is already discussed in detail. Now discuss its complexity analysis.

Line 1-2 execute only one time in a single execution of the algorithm.

Line 3 executes n/2 + 1 times in a single execution of the algorithm.

Line 4 executes n/2 times in a single execution of the algorithm.

Line 5-7 executes

Table I. Complexity Analysis

Line No.	Iteration
1.	i,j,n,Temp
2.	i = 1, j = n
3.	While(i<j)
4.	If(DATA[i] > DATA[j])
5.	Temp← DATA[i]
6.	DATA[i] ← DATA [j]
7.	DATA[j] ←Temp
8.	i++, j--
9.	FLAG ← 1
10.	for i←1 to n/2 && FLAG
11.	FLAG←0
12.	for j←i to n-i
13.	if(DATA[j]> DATA[j+1])
14.	Temp← DATA[j]
15.	DATA[j]← DATA[j+1]
16.	DATA[j+1]← Temp
17.	FLAG←1
18.	if(DATA[n-j]> DATA[n-j+1])
19.	Temp← DATA[n-j]
20.	DATA[n-j]← DATA[n-j+1]
21.	DATA[n-j+1]←Temp
22.	FLAG←1

$\sum_{K=0}^{n/2} L$

times in a single execution of the algorithm.

Note: L=1 when if statement is true, else L=0.

$\sum_{k=0}^{n/2} t$ = this evaluates to a constant value based on the value of t (either 1 or 0).

Line 8 executes n/2 times in a single execution of the algorithm.

Line 9-10 execute only one time in a single execution of the algorithm.

Line 11-12 executes n/2 times in a single execution of the algorithm.

Line 13-22 executes

$\sum_{K=0}^{n/2-1} (2k+1).L$

*Note :*L=1 when if statement is true, else L=0.

Suppose for L=1 we have

$$\sum_{K=0}^{n/2-1} (2k+1) = 2\sum_{K=0}^{n/2-1} k + \sum_{K=0}^{n/2-1} 1$$

$= 2(((n/2-1)*((n/2-1)+1))/2) + (n/2-1)$

[By Applying 1+2+3+…n = (n(n+1)/2)]

$= ((n^2-2n)/4) + (n/2-1)$

Now calculating total time taking by proposed algorithm,

$T(n) = 1 + n/2 + 1 + n/2 + 1 + n/2 + 1 + n/2 + (n^2-2n)/4 + (n/2-1)$

$= n^2/4 + 4(n/2) + 3$

Now taking only the largest term, i.e. n^2 so running time of the algorithm is,

$T(n) = O(n^2)$

Working

Let the given set of elements are 87, 33, 48, 74, 13, 66.

Bubble Sort

Table II. Working of Bubble Sort

Step No.	Elements					
1.	87	33	48	74	13	66
2.	33	87	48	74	13	66
3.	33	48	87	74	13	66
4.	33	48	74	87	13	66
5.	33	48	74	13	87	66
6.	33	48	74	13	66	87
7.	33	48	74	13	66	87
8.	33	48	74	13	66	87
9.	33	48	13	74	66	87
10.	33	48	13	66	74	87
11.	33	48	13	66	74	87
12.	33	13	48	66	74	87
13.	33	13	48	66	74	87
14.	33	13	48	66	74	87
15.	13	13	48	66	74	87
16.	13	33	48	66	74	87
Sorted	13	33	48	66	74	87

BESA Sort

Table III. Working of debs sort

Step No.	Pointer	Elements					
1.	i=1, j=6	**87**	33	48	74	13	**66**
2.	i=2, j=5	66	**33**	48	74	**13**	87
3.	i=3, j=4	66	13	**48**	**74**	33	87
4.	p=1,q=2,r=6,s=5	**66**	**13**	48	74	**33**	**87**
5.	p=2,q=3,r=5,s=4	13	**66**	**48**	**74**	**33**	87
6.	p=3,q=4,r=4,s=3	13	48	**66**	**33**	74	87
7.	p=4,q=5,r=3,s=2	13	**48**	**33**	**66**	**74**	87
8.	p=5,q=6,r=2,s=1	**13**	**33**	48	66	**74**	**87**
9.	p=2,q=3,r=5,s=4	13	**33**	**48**	**66**	**74**	87
10	p=3,q=4,r=4,s=3	13	33	**48**	**66**	74	87
11.	p=4,q=5,r=3,s=2	13	**33**	**48**	**66**	**74**	87
Sorted		13	33	48	66	74	87

Comparison

On the Basis of Passes

1)Worst Case Analysis: In a sorting algorithm, Worst case refers to the situation when the given set of numbers is in decreasing order while the required set has to be in increasing order.

Table IV. Worst case analysis

	N = 10	N = 49	N = 100	N = 499	N = 1000
Bubble	9	48	99	498	999
Insertion	9	48	99	498	999
BESA	1	1	1	1	1

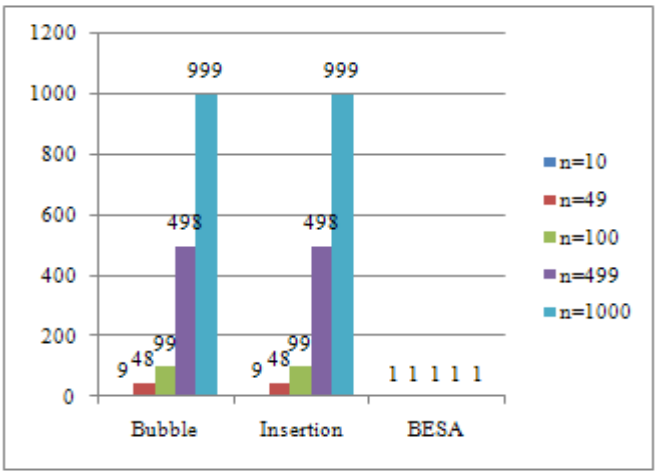

Fig. 1 Worst Case Analysis

2)Average Case Analysis: Average case in a sorting algorithm refers to a random manner.

Table V. Average Case Analysis

	N = 10	N = 49	N = 100	N = 499	N= 1000
Bubble	9	48	99	498	999
Insertion	9	48	99	498	999
BESA	3	9	19	77	157

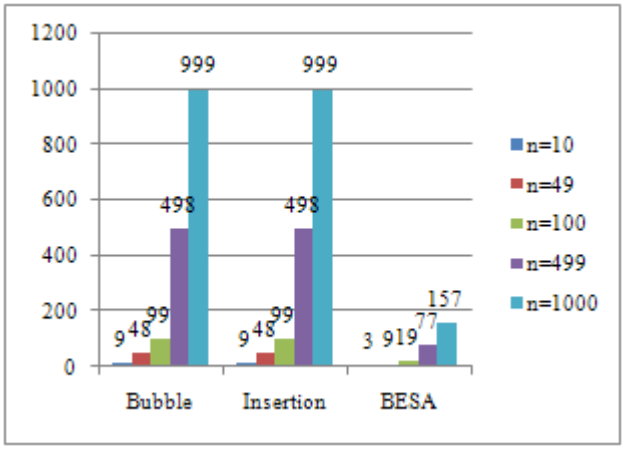

Fig. 2 Average Case Analysis

3) Best Case Analysis: Best case in a sorting algorithm refers to the situation when the given set of numbers is in already in increasing order.

Table VI. Best Case Analysis

	N = 10	N = 49	N = 100	N = 499	N= 1000
Bubble	9	48	99	498	999
Insertion	9	48	99	498	999
BESA	1	1	1	1	1

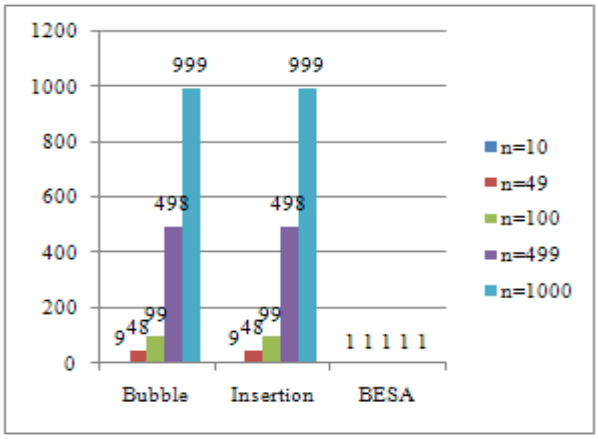

Fig. 3 Best Case Analysis

On the Basis of Complexity

BESA works exceptionally well in worst case scenarios. The 1st step of the algorithm places each element as its proper position with minimum number of comparisons and swaps.

Table VII. Analysis on the basis of complexity

	Worst Case	Average Case	Best Case
Bubble	$O(n^2)$	$O(n^2)$	O(n)
Insertion	$O(n^2)$	$O(n^2)$	$O(n^2)$
BESA	O(n)	$O(n^2)$	O(n)

Conclusion & Future Scope

In this research paper we have studied about different sorting algorithms along with their comparison. Every sorting algorithm has advantage and disadvantage. The fundamental sorting algorithms are basic sorting algorithm and we have try to show this how disadvantage of fundamental sorting algorithm have removed in advance sorting algorithm. Various Sorting algorithms have been compared on the basis of different factors like complexity, number of passes, number of comparison etc. After the study of all various sorting algorithms we observed that there is no such algorithm, which works in this way that to sort the elements on both ends. So we have proposed sorting algorithm, which work on the basis of both end comparison front as well as rear. For implementation to the proposed algorithm we have to use MATLAB.

My first target is to remove the demerits of various sorting algorithms. It is also seen that many algorithms are problem oriented so we will try to make it global oriented. Hence we can say that there are many future works which are as follows.

- Remove disadvantage of various fundamental sorting and advance sorting.
- Make problem oriented sorting to global oriented.

In the end we would like to say that there is huge scope of the sorting algorithm in the near future, and to find optimum-sorting algorithm, the work on sorting algorithm will go on forever.

Acknowledgment

I would like to express my sincere thanks to Asst. Prof. Anubhooti Papola for his advice during my work. As my supervisor, she has constantly encouraged me to remain focused on achieving my goal. I extend my thanks to Asst. Prof. Nitin Arora for his valuable advices and encouragement. I must acknowledge the academic resources that I have got from UTU Dehradun.

References

[1] Y.Han "Deterministic sorting in O(nloglogn) time and linear space", Proceeding of the thirty-fourth annual ACM symposium on theory of computing, Monteral Quebec, Canada, **(2002)**, p.602-608.

[2] Y.Han, M.Thorup, "Integer Sorting in O(nloglogn) time and linear space" proceesings of the 43rd symposium on foundations of Computer Science, **(2003)**, p.135-144.

[3] J.L. Bentley and R.Sedgewick. "Fast Algorithms for Sorting and Searching Strings", ACM-SIAM SODA " **(2003)**, 360–369.

[4] G. Franceschini and V. Geffert, "An In-Place Sorting with O(n log n) Comparisons and O(n) Moves", Proceedings of 44th Annual IEEE Symposium on Foundations of Computer Science, **(2003)**, pp. 242-250.

[5] M. A. Bender, M. Farach-Colton and M. A. Mosteiro, "Insertion Sort is O(n log n)", Proceedings of the Third International Conference on Fun With Algorithms (FUN), **(2004)**, pp. 16-23.

[6] A. D. Mishra and D. Garg, "Selection of the best sorting algorithm", International Journal of Intelligent Information Processing, vol. 2, no. 2, (2008) July-December, pp. 363-368.

[7] O. O. Moses, "Improving the performance of bubble sort using a modified diminishing increment sorting", Scientific Research and Essay, vol. 4, no. 8, **(2009)**, pp. 740-744.

[8] J. Alnihoud and R. Mansi, "An Enhancement of Major Sorting Algorithms", International Arab Journal of Information Technology, vol. 7, no. 1, **(2010)**, pp. 55-62.

[9] Sultanullah Ja doon et al., "Design and Analysis of Optimized Selection Sort Algorithm", International Journal of Electric & Computer Sciences IJECS-IJENS, **(2011)**Vol: 11 No: 01.

[10] Savina & Surmeet Kaur, "Study of Sorting Algorithm to Optimize Search Results", International Journal of Emerging Trends & Technology in Computer Science, **(2012)** Volume 2, Issue 1.

[11] Md. Khairullah, "Enhancing Worst Sorting Algorithms", International Journal of Advanced Science and Technology, **(2013)** Vol. 56.

[12] Ibrahim M. Al Turani, Khalid S. Al-Kharabsheh & Abdallah M. AlTurani, "Grouping Comparison Sort", Australian Journal of Basic and Applied Sciences, **(2013)**, 7(7): 470-475.

[13] Nitin Arora, Anil Kumar & PramodMehra, "Two Way Counting Position Sort", International Journal of Computer Application **(2013)**.

[14] Partha Sarathi Dutta, " Design and Analysis of Hybrid Selection Sort Algorithm", International Journal of Applied Research and Studies **(2013)**.
[15] Surender Lakra & Divya, "Improving the performance of selection sort using a modified double-ended selection sorting", International Journal of Application or Innovation in Engineering & Management (IJAIEM) **(2013)**.
[16] Pankaj Sareen, "Comparison of Sorting Algorithms", International Journal of Advanced Research in Computer Science and Software Engineering **(2013)**.
[17] R.Srinivas & A.RagaDeepthi, "Novel Sorting Algorithm", International Journal on Computer Science and Engineering **(2013)**.
[18] Partha Sarathi Dutta, "An Approach to Improve the Performance of Insertion Sort Algorithm", International Journal of Computer Science & Engineering Technology **(2013)**.
[19] Khalid Suleiman Al-Kharabsheh et al., "Review on Sorting Algorithms A Comparative Study", International Journal of Computer Science and Security (IJCSS), **(2013)**, Volume (7) : Issue (3).
[20] Savina & Surmeet Kaur, "Study of Sorting Algorithm to Optimize Search Results", International Journal of Emerging Trends & Technology in Computer Science, **(2013)** Volume 2, Issue 1.