International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 2– No.4, May 2012 – www.ijais.org

Freezing Sort

Surmeet Kaur
2,Partap Nagar Near Zila Parishad, Sangrur(Pb)
Assistant Professor, Department of Computer Science and Engineering. Lovely Professional University, Jalandhar, India

Tarundeep Singh Sodhi
H.No 599 Mota Singh Nagar Jalandhar(Pb)
Assistant Professor Department of Computer Science and Engineering. Arni University, Kathgarh-Kangra, Himachal Pradesh, India

Parveen Kumar
HNo 484/5M,New Govind Puri Kanker Khera Meerut Cantt.
Assistant Professor, Department of Computer Science and Engineering, Lovely Professional University, Jalandhar, India

ABSTRACT
The term algorithm is now applied to many kinds of problem solving, such as in setting up a computer program. An algorithm is as a finite sequence of steps, that can be used for solving many problems.copied Algorithms are used for calculation, data processing, and many other fields. In computer science and mathematics, a sorting algorithm is an algorithm that puts elements of a list in a certain order, not necessarily in increasing order, it may be in decreasing order as well. Efficient sorting is important to optimizing the use of other algorithms that require sorted lists to work efficiently; it is also often useful for producing human- readable output.[1]

In this paper we present a new sorting algorithm, named as freezing algorithm, which uses the methodology of bubble sort efficiently to give a much better performance than the existing sorting algorithms of the O (n^2) class. We prove the correctness of the algorithm and give a detailed time complexity analysis of the algorithm.

General Terms
Algorithm, Stability, Freezing, Comparisons

Keywords
Algorithm, Complexity, Bubble sort, Selection sort.

1. INTRODUCTION
Sorting algorithms are classified by several other criteria such as Computational complexity (worst, average and best number of comparisons) in terms of the size of the list, Stability (Memory usage and use of other computer resources). The difference between best, worst case and average behavior, behaviors are practically important data sets.

There are many sorting algorithms, among which is Bubble Sort. The basic sorting algorithms are mostly iterative, and thus probably easier to understand. [8]The simplest algorithm, but not the most efficient algorithm, is Bubble sort. Bubble Sort is not known to be a very good sorting algorithm when compared to other sorting algorithms. This sort is still used a lot because it is easier and shorter to type than the other sorts. However, efforts have been made to improve the performance of the algorithm.

This paper presents a new algorithm called Freezing Sort that enhances the performance of bubble sort by reducing the number of swaps and the number of comparisons. The results from the implementation of the algorithm compared with the Bubble sort and other recently used algorithms showed that the algorithm performed better than the others in the worst case scenario.

In general $O(n^2)$ sorting algorithms run slower than O(nlogn) algorithms, but still their importance cannot be ignored. Since $O(n^2)$ algorithms are non recursive in nature, they require much less space on the RAM.[5]

The paper is organized as follows: section – 2 gives the basic algorithm and section -3 shows an example to illustrate the working of the algorithm. Section – 4 proves the correctness of the algorithm using the pseudo code, section – 5 gives a detailed time complexity analysis of the algorithm, section – 6 gives number of swaps and also shows the stability of the proposed algorithm, section – 7 shows the comparison of freezing algorithm with recently used and enhanced algorithms, section – 8 concludes, - 9 gives acknowledgement and finally section – 10 gives important references.

2. FREEZING ALGORITHM
Here we can sort the list of elements in an array by a new freezing technique.

STEP1: We start with first element (BEG) and compare it with the last element (END) in the list. Swap the values if the last element is smaller than the first one.i.e. If END<BEG

STEP2: Now BEG=BEG+1 AND END=END-1

STEP 3: Now repeat step1 for new BEG and END

STEP4: Now freeze the last element in the list and repeat step1 to 3 for the remaining elements and then freeze the first element and repeat the process for the remaining list.

STEP 5: Each time increase the no of freezing elements by 1.

International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 2– No.4, May 2012 – www.ijais.org

So at the end we will have a SORTED LIST of elements.

3. EXAMPLE

Consider the list as below:

60 5 55 10 22 **35**

BEG **END**

- Compare BEG and END element. Swap them as END<BEG
- BEG=BEG+1 and END=END-1

35 **5** 55 10 **22** 60

BEG **END**

- Compare BEG and END element. No need to swap them as END>BEG
- BEG=BEG+1 and END=END-1

35 5 **55** **10** 22 60

BEG **END**

- Compare BEG and END element. Swap them as END<BEG

35 5 55 10 **22** ***60***

BEG **END**

- Freeze the last element from the list and repeat the process with the remaining list. We get…

22 5 55 10 35 ***60***

- Now freeze the first element in the list and repeat the process with the remaining list. We get…

22 **5** 55 10 35 **60**

BEG **END**

We get…

22 **5** 35 10 55 **60**

BEG **END**

- Now freeze the last two elements from the list and repeat the process with the remaining list. We get…

10 5 25 22 ***55*** ***60***

- Now freeze the first two elements from the list and repeat the process with the remaining list and so on….

4. PSEUDOCODE

In simple pseudo code, freezing sort algorithm might be expressed as:

```
1.  Calculate length n
2.  var  k, x=0, i=1, j=n
3.  k=n*mod2
4.  If k=0, then
5.  k=n/2,
6.  else k=(n+k)/2
7.  endif
8.  i= i+x , j=n
9.  do while( i=k-x)
10. compare a[i] with a[j] and swap,
11. i++,j--
12. do while (x<=n-2)
13. x =x+1
14. i=1,j=n-x
15. do while (i=k-x)
16. compare a[i] with a[j] and swap
17. i++,j--
18. goto 8
19. end
```

5. COMPLEXITY

Best Case, Average Case and Worst Case.
The complexity of freezing algorithm is O(n) in best as well as average case when all the elements are either in increasing and decreasing order respectively. So we can say that we just need to compare n/2 or (n-1)/2 times as we do not need to freeze any element (lines 8-11 of pseudo code).[10]

The complexity of this freezing algorithm is $O(n^2)$ in average case which is the same as that of selection sort and bubble sort. For even or odd number of elements, no of comparisons are n(n-1)/2. For example if:

n	no of comparisons
5	10
6	15
7	21
8	28

6. NUMBER OF SWAPS AND STABILITY

6.1 Swapping Operations

In best case, there is no swapping operation as all the elements are already sorted.

In average and worst case, when all the elements are unsorted or in decreasing order then the number of swaps are n/2 or (n-1)/2 depending upon the odd or even number of elements. For example:

50 30 20 10 **5**

5 **30** 20 **10** 50

5 10 20 30 50

Number of swaps=2 i.e (n-1)/2

6.2 Stability

Stablility means maintaining the relative order of records with equal values, i.e a sorting algorithm is stable if whenever there are two records M and N with the same key and with M appearing before N in the original list, M will appear before N in the sorted list. Freezing algorithm being data dependent is a stable algorithm.

7. COMPARISON

When compared the proposed algorithm with recently used as well as enhanced algorithms, it has been found that the freezing algorithm shows the best results. The two tables below shows the comparisons of the number of swaps and complexities of different sorting algorithms with the freezing algorithm.

Table 1-Comparison with recently used algorithms

	Freezing Sort	**Bubble Sort**	**Selection Sort**	**Insertion Sort[12]**
Best Case Complexity	O(n)	$O(n^2)$	$O(n^2)$	O(n)
Average Case Complexity	O(n)	$O(n^2)$	$O(n^2)$	$O(n^2)$
Worst Case Complexity	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$
Number of Swaps	0,n/2 or (n-1)/2	0,n/2,n	Always n	

Table 2: Comparison with Enhanced algorithms

	Freezing Algorithm	**Enhanced Selection Sort**	**Enhancd Bubble Sort**
Best Case Complexity	O(n)	$O(n^2)$	O(nlogn)
Average Case Complexity	O(n)	$O(n^2)$	O(nlogn)
Worst Case Complexity	$O(n^2)$	$O(n^2)$	O(nlogn)
Number of Swaps	0,n/2 or (n-1)/2	Depends on data:0,n/2 or n	Always n/2

Here we can also present the efficiency of Freezing sort by showing the number of comparisons in average case for equal number of elements with the help of graphs also:

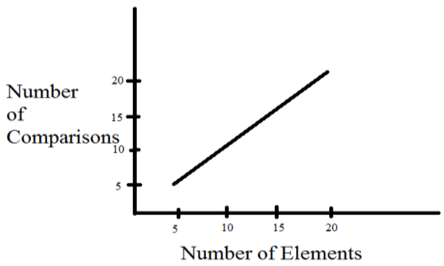

Figure1: Comparisons in Freezing Sort

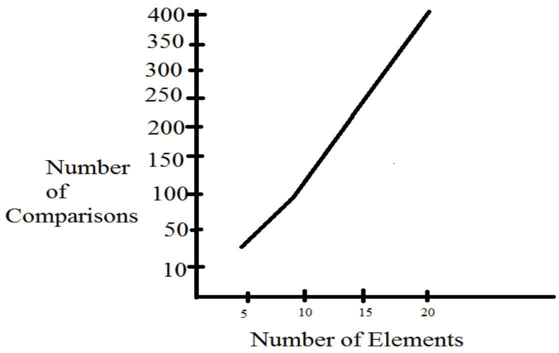

Figure2: Number of Comparisons in Bubble Sort, Insertion Sort, Selection Sort

8. CONCLUSION

From the results in Table 1 and 2 and Figure1 and Figure2, freezing algorithm is very good in average case as it reduces the number of comparisons as well as number of swaps in comparison to the other algorithms. It is basically an enhancement to bubble sort as it always compare two elements at a time like bubble sort. The implication of these is that the proposed algorithm is faster and therefore, more efficient.

The Freezing Algorithm has been compared with recently used as well as enhanced algorithms and it has been found that the freezing algorithm shows the best results.

9. ACKNOWLEDGMENT

I would like to thank my mentor, Mr. Parveen Kumar for providing support and material related to the area of this research, and for his suggestions and helpful comments. I am thankful to all of my friends and colleagues who have extended their helping hand in one way or another and provided several valuable suggestions. I am also thankful to all those who appreciated this work and provided me a boost to think and to work more effectively in this area of sorting.

10. REFERENCES

[1] Knuth D.E,"The art of programming- sorting and searching,".2nd edition Addison Wesley.

[2] D.E.Knuth, Sorting and Searching, volume 3 of "The Art of Computer Programming". Addison Wesley, Reading, MA, (1973)

[3] Cormen T., Leiserson C., Rivest R., and Stein C., “Introduction to Algorithms,” McGraw Hill, (2001)

[4]RupeshSrivastava,TarunTiwari,Sweetes Singh,”Bidirectional Expansion – Insertion Algorithm for Sorting,”Second International Conference on Emerging Trends in Engineering and Technology, ICETET-09. (2009).

[5] Rami Mansi,”Enhanced Quicksort Algorithm,” The International Arab Journal of Information Technology, Vol. 7, No. 2. (2010).

[6] Jehad Alnihoud and Rami Mansi,” An Enhancement of Major Sorting Algorithms,” The International Arab Journal of Information Technology, Vol.7.(2010)

[7] Muhammad Anjum Qureshi,” Qureshi Sort: A new Sorting Algorithm,”.(2010)

[8] Oyelami Olufemi Moses,” Improving the performance of bubble sort using a modified diminishing increment sorting,”. Scientific Research and Essay Vol. 4 (8), pp.740-744.(2009)

[9] Vandana Sharma, Satwinder Singh and Dr.K.S.Kahlon, “Performance Study of Improved Heap Sort Algorithm and Other Sorting Algorithms on Different Platforms,”IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.4.(2008)

[10] Seymour Lpischutz, G A Vijayalakshmi Pai (2006) “Data Structures”, Tata McGraw-Hill Publishing Company Limited, p.4.11, 9.6, 9.8.

[11] You Yang, Ping Yu, Yan Gan, “Experimental Study on the Five Sort Algorithms”, International Conference on Mechanic Automation and Control Engineering (MACE), 2011.

[12]Wang Min “Analysis on 2-Element Insertion Sort Algorithm”, International Conference on Computer Design And Appliations (ICCDA), 2010.

[13] 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 & Computer Sciences IJECS-IJENS Vol: 11 No: 01, February 2011.

[14]Sultanullah Jadoon, Salman Faiz Solehria, Mubashir Qayum, “Optimized Selection Sort Algorithm is faster than Insertion Sort Algorithm: a Comparative Study” International Journal of Electrical & Computer Sciences IJECS-IJENS Vol: 11 No: 02, April 2011.

[15]Wang Min “Analysis on 2-Element Insertion Sort Algorithm”, International Conference on Computer Design And Appliations (ICCDA), 2010.