Volume 4, Issue 3, March 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Performance Analysis of Enhanced Selection Sort Algorithm Based on Even-Odd Elements for Positive Integers

Sarvjeet Singh
(M.Tech. Scholar)
Deptt. of Computer Science and Engineerin g
Lovely Professional University
Phagwara,Punjab, India

Surmeet Kaur
(Assistant Professor)
Deptt. of Computer Science and Engineering
Lovely Professional University
Phagwara,Punjab, India

Abstract— One of the most problem for the computer to mapping the data into databases. Nowadays many algorithms are introduced for the sorting. Generally Sorting is the operation that arranged the logically data into the database. There are the application of the sorting is searching, used a different method like linear, based on indexed sequentially to make access fast and efficiently. In this research paper introduced an enhancement of performance of the selection sort using the classification of data on the basis of even and odd elements. We conclude the result of the existing algorithm with the proposed sorting algorithm with the help of mathematics and practically implementation of the algorithms. Time complexity of enhanced selection sort in the best case is big O(n log n).

Keywords— Big O Notation, CPU, RAM, Complexity.

I. INTRODUCTION

Algorithm is the vital part of the operational research methodology. Basically algorithms involved the step by step procedures to solve the problem and it is terminated after the when statement either true or false vice-versa. So there are the the two method to analysis of the sort algorithms based on the mathematical analysis and second is the empirical analysis[2].There are the two factor based for the algorithms complexity:-

1.Time (how much time taken to execution)
2.Space (how much space algorithm require)

There are several factor which affected the algorithms complexity:-

1. Number of comparisons.
2. Number of swaps
3. Number of arithmetic operations.

Number of iterations.This document is a template. An electronic copy can be downloaded from the Journal website. For questions on paper guidelines, please contact the journal publications committee as indicated on the journal website. Information about final paper submission is available from the conference website.

II. EXISTING SORTING ALGORITHMS

1) Bubble Sort Algorithm:- This is simplest sorting algorithm, in which elements scan from starting index of an array and the compare with adjacent element if its greater than it swap if it's not than second element compare with third and so on till array is not equal to null, and this process repeated until list is being sorted. The time complexity is O(n) in best case and $O(n^2)$ in worst case and as well as space complexity is depends upon the number of inputs. Bubble Sort Algorithm is as follows[8]:-

```
BubbleSortAlgorithm( A , n)
1. for i ← 0 to n – 1 do
2.       for j ← 0 to n – i
3.                if A[j] < A[j+1] then
4.                swap( A[j] , A[j+1] )
5.       end for of step 2
6. end for of step 1
```

Figure 1.1 Bubble Sorting Algorithm

2) Selection Sorting Algorithm:-In this method of sorting , selection method is being used , basic formula to conclude the algorithm is that smallest element is selected from the array and placed in the proper location of the array in

particular pass, this is repeated until list is being sorted. The time complexity of the algorithm is O(n^2) in best and worst case of the input data size[2].

```
SelectionSortAlgorithm(A,n)
1. i ← 0
2. while i < n do
3. j ← i+1
4.        while j < n do
5.                  if A[i] > A[j] then
6.                  Swap(A[i],A[j])
7.                           j ← j+1
8.                  End if of step 5
8.         End While of step 4
9. i = i+1
10 end-while
```

Figure 1.2 Selection Sorting Algorithm

III. PROPOSED ENHANCED SELECTION SORTING TECHNIQUE

In the proposed sort algorithms we can take the input from the user equal number of the even and odd elements and after that we can take the temporary array of the four size and after the selection of the elements there are we can extract the even values as well as odd values from the inputted array. After the pick values from it store in temporary array and compare to each other and fixes into a fixed sized array of n number, where n equal to number of inputs.

Algorithms:-

Step 1:-Input the array.
Step 2:-Assign proper index to the array.
Step 3:-Assign temporary array of fixed size of four elements.
Step 4:- Select the minimum even and minimum odd value from the array and store into the temp array of fixed size four.
Step 5:- go to step 4 till temp array full.
Step 6:- Compare the values in temp array and store in the different array till temp array n/2!=0.
Step 7:- Go to step 4.
Step 8:-End.

Pusdocode of fetch values from the algorithms:-

1. For Fetch Even Values From List:-

```
for(int k=0;k<evenl;k++)
{
        for(int p=k+1;p<evenl;p++)
        {
                if(even[p]<even[k])
                {
                        int sw=even[k];
                        even[k]=even[p];
                        even[p]=sw;
                }
        }
}
```

2. For Fetch Odd Values From List:-

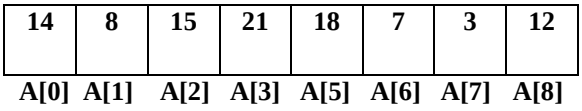
```
for(int k1=0;k1<oddl;k1++)
{
        for(int p1=k1+1;p1<oddl;p1++)
        {
                if(odd[p1]<odd[k1])
                {
                        int sw1=odd[k1];
                        odd[k1]=odd[p1];
                        odd[p1]=sw1;
                }
        }
}
```


After fetch the values from list then we compare the values which is temporary stored in the values in the four size fixed array and then compare with all an freeze it after the comparison of values from fixed size array, there are in third separated list of an elements which which used to store the values comes from after comparisons.

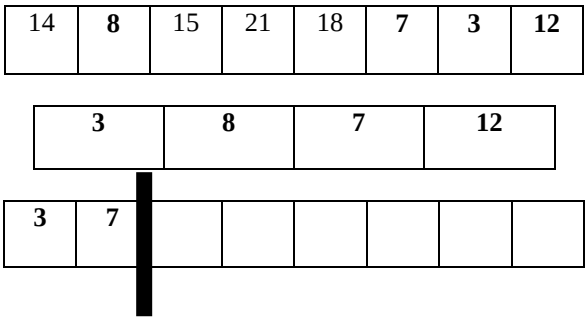
3. Comparisons of fetched Even and Odd Values:-

```
for(int jl=0;jl<length;jl++)
{
    if((even[e2]<odd[o2] &&e2!=e1)||o2>=o1)
    {
                final1[jl]=even[e2];
                e2++;
                continue;
    }
    if((even[e2]>odd[o2]&&o2!=o1) || e2>=e1)
    {
        final1[jl]=odd[o2];
        o2++;
        continue
    }
}
```

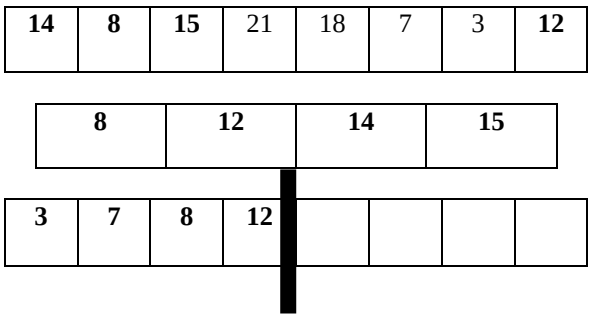
Example with Proposed Algorithm:-



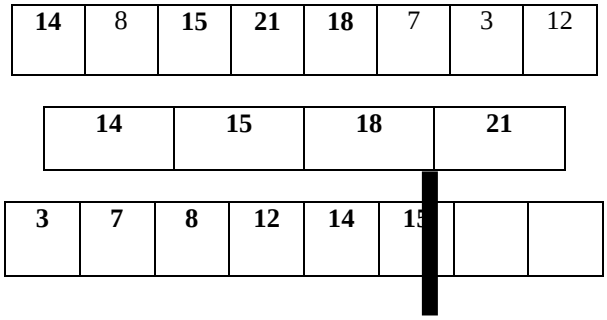
After first Pass :-



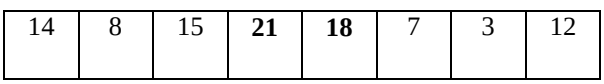
After Second Pass:-



After Third Pass:-



After Fourth Pass:-



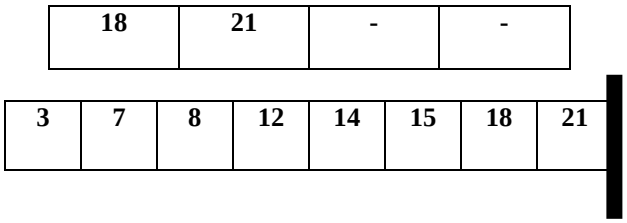


Table 1.1 Comparisons on the basis of complexity and other factors

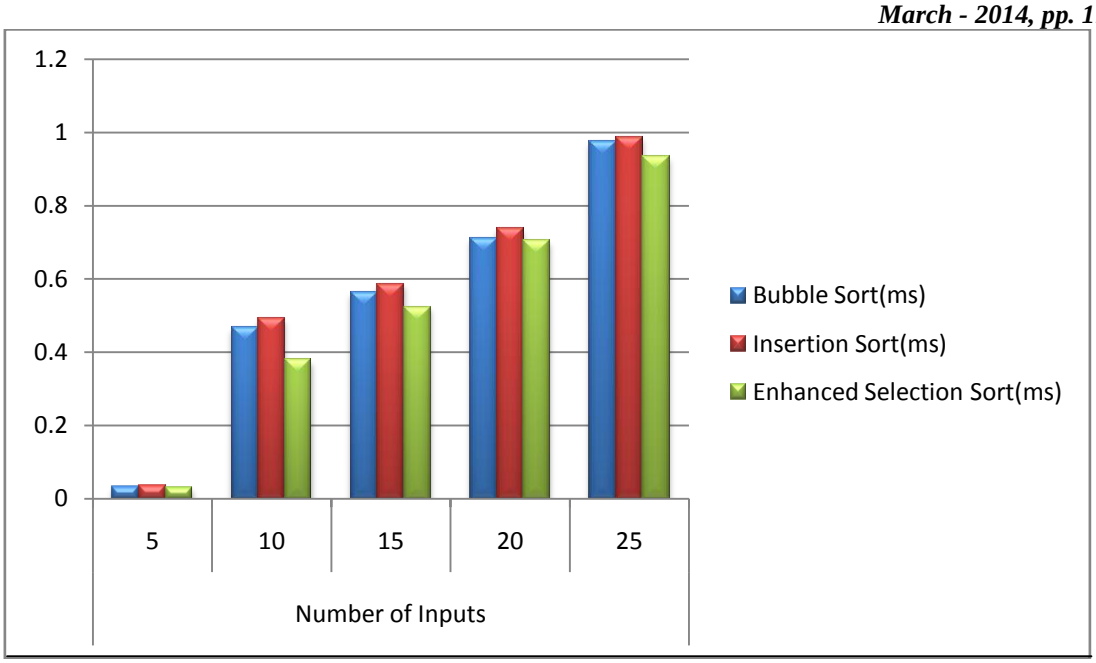
	Bubble Sort Algorithm	Selection Sort Algorithm	Insertion Sort Algorithm	Merge Sort Algorithm	Enhanced Selection Sort Algorithm
Time Complexity:- Best Case Average Case Worst Case	O(n) $O(n^2)$ $O(n^2)$	$O(n^2)$ $O(n^2)$ $O(n^2)$	O(n) $O(n^2)$ $O(n^2)$	O(n log n) O(n log n) O(n log n)	O(n log n) $O(n^2)$ $O(n^2)$
Passes used in above Example	7	6	6	5	7
Space Complexity	O(1)	O(1)	O(1)	O(n)	O(1)
Method used	Exchange	Selection	Incremental	Merging	Selection
In Place Algorithm	Yes	Yes	Yes	No	Yes
Stable Algorithm	Yes	No	Yes	Yes	Yes
Type of Memory used	Internal Memory	Internal Memory	Internal Memory	Internal and External Memory	Internal Memory

Figure 1.1 Bubble Sort Example with same data

i=0	j	0	1	2	3	4	5	6	7
	0	14	8	15	21	18	7	3	12
	1	8	14	15	21	18	7	3	12
	2	8	14	15	21	18	7	3	12
	3	8	14	15	21	18	7	3	12
	4	8	14	15	18	21	7	3	12
	5	8	14	15	18	7	21	3	12
	6	8	14	15	18	7	3	21	12
i=1	0	8	14	15	18	7	3	12	**21**
	1	8	14	15	18	7	3	12	
	2	8	14	15	18	7	3	12	
	3	8	14	15	18	7	3	12	
	4	8	14	15	7	18	3	12	
	5	8	14	15	7	3	18	12	
i=2	0	8	14	15	7	3	12	**18**	
	1	8	14	15	7	3	12		
	2	8	14	15	7	3	12		
	3	8	14	7	15	3	12		
	4	8	14	7	3	15	12		
i=3	0	8	14	7	3	12	**15**		
	1	8	14	7	3	12			
	2	8	7	14	3	12			
	3	8	7	3	14	12			
i=4	0	8	7	3	12	**14**			
	1	7	8	3	12				
	2	7	8	3	12				
i=5	0	7	8	3	**12**				
	1	7	3	8					
i=6	0	7	3	**8**					
		3	**7**						
		3							

1) Practical Results:-

In the comparisons of the other algorithms execution time is calculated. We conclude the result from the selection, bubble,insertion,freeze sorting algorithm. The algorithm efficiency is measured on the basis of the CPU time based on the time taken by the running process at background area of the n number of inputs data. The algorithm was obtained results from the C Language with the help of by in built function Clock().It runs on Sony Vaio E-Series laptop with the following specification:-Intel(R) Core™ i5-2450M CPU at 2.50GHz with the 4 GB RAM. In first table result will shows of the clock() function[2].



IV. CONCLUSIONS

In this proposed algorithms, There are O (n log n) time complexity. The number of element is select by n/2 inputs after comparisons in particular pass is occur, and it is lesser than the Selection sort algorithm. It is called as stable and in place algorithm also because it selects the element from left to right in array. It needs only O (1) space complexity. So the performance analysis is based upon the two different scenario one is mathematically and second practical implementation. In different term of the sort it support the stable sorting because selection of the elements is directly linear form.

REFERENCES

[1] Knuth D.E,"*The art of programming- sorting and searching*,".2nd edition Addison Wesley.
[2] 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.
[3] 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.
[4] Wang Min "*Analysis on 2-Element Insertion Sort Algorithm*", International Conference on Computer Design And Appliations (ICCDA), 2010.
[5] Kaur Surmeet,"*Freezing Sort*",International Journal of Applied Information Systems (2249-0868), Volume 2,No.4 May 2012.
[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] Cormen T., Leiserson C., Rivest R., and Stein C., "*Introduction to Algorithms*," McGraw Hill, (2001).
[9] Singh Tarundeep, Kaur Surmeet, Kaur Snehdeep," *Enhanced Insertion Sort Algorithm*", international Journal of Computer Applications (0975-8887) ,Volume 64,No.21, February 2013.
[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] Agarwal Aayush, Vikas Pardesi, Namita Agarwal," *A New Approach To Sorting: Min-Max Sorting Algorithm*", May-2013 IJERT.
[13] Bhardwaj Shagun," *Mark-Set{} Sort*", 2013 IEEE.
[14] Devi S. Gayathri , K. Selvam , Dr. S. P. Rajagopalan, " *An Abstract to Calculate Big O Factors of Time and Space Complexity of Machine Code*",2011 SEISCON.
[15] Maurya V. N., Bathla R. K., Kaur Diwinder, Maurya Avadhesh, Gautam Ram Asrey, "*An Alternate Efficient Sorting Algorithm Applicable for Classification of Versatile Data*", April-2013 IJMMAC.
[16] Sareen Pankaj, "*Comparison of Sorting Algorithms (On the Basis of Average Case)*", March 2013 IJRCSSE.

