ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 10, October 2013

A New Approach for Selection Sorting

[1]Mrs.P.Sumathi, [2]Mr.V.V.Karthikeyan
[1]Assistant Professor (Senior Grade), Department of Information Technology,
[2]Associate Professor, Department of Electronics and Communication Engineering,
SNS College of Engineering, Coimbatore

***Abstract*-One of the fundamental areas of the computer science is Data Structure. Sorting is a major issue in Data Structure which creates the sequence of the list of items. Though numbers of sorting algorithms are available, it is necessary to select the best sorting algorithm. So sorting problem has attracted a great deal of research as sorting technique is frequently used in a large variety of important applications to arrange the data in ascending or descending order. This paper presents a Double Ended Selection Sort Algorithm which is faster than the selection sort. The new algorithm is analyzed, implemented, tested. The test results are presented and compared with the traditional selection sort.**

Keywords: ***Data structures, Sorting, Double Ended Selection Sort Algorithm, Selection sort.***

1. Introduction

In this computational world, sorting plays an important role. All information need or follow a sorting order which is either descending or ascending. In the earlier methods, sorting takes longer time than the processing time of data. But now many sophisticated sorting algorithms are introduced and also the research is going on this area to develop new sorting algorithms. There are many well known sorting algorithms for unsorted list, such as selection sort, Quick sort, Insertion sort, Merge sort, Bubble sort and so on. There are many factors to evaluate the performance of the sorting algorithms which are given below. [5]

1. Computational complexity (worst, average and best behavior) (*n*).
2. Usage of memory and other computer resources.
3. Stability
4. Number of swaps (for in-place algorithms).
5. Number of comparisons

Choosing the appropriate sorting algorithm for a particular application is important. Every sorting algorithm has its own advantages and disadvantages. The performance factors of the algorithm help to know whether the chosen algorithm is the most suited one. The existing selection sort algorithm is best for already sorted list, average case for unsorted list neither ascending nor descending and Worst case for unsorted list in reverse[1][12]. This paper presents an algorithm, DESSA, to enhance the performance of selection sort.

2. Double Ended Selection Sort Algorithm

2.1. Concept

DESSA inserts an array of elements and sorting these elements in the same array (in-place) by finding the maximum element, minimum element and exchanges them with the last element, first element respectively and then decreases the size of the array by two for next call.

2.2. Procedure

The procedure of the algorithm can be described as follows:

- Insert all elements in the array.
- Call the "DESSA" function with passing the array and its size as parameters.
- Find the maximum element and minimum element in the array among first two elements.
- Max and min position are assigned.
- Min position value is compared with third position element. If it is less than min position element min position is replaced with current min element position.
- Otherwise it is compared with Max position element. If Current position element is greater than max position element, Max position is updated as current position.
- Otherwise, min and max positions remain the same.

- The above steps are continued until last position is reached.
- Swap max and min position elements in first and last positions.
- Decrease the size of the array by two for next iteration from 1 to n-2 position elements.
- Repeat the same procedure for 1 to n-2 position elements.
- Operationally, the (n-*2*) after the first call becomes (n-*4*), and after the second call becomes (n-*6*), and so on.
- Iteration continues up to <n/2 times.

The algorithm is illustrated with the following:

Array position → *0* *1* *2* *3* *4*
Array Element→ 31 52 27 5 68

Iteration 1:
Fix Min=0,Max=1

Min Max Curr
Array position → *0* *1* *2* *3* *4*
Array Element→ 31 52 27 5 68

Array [Min]>Array [curr]=?
Yes
Change Min as curr position
Min=2
Max=1

Max Min Curr
Array position → *0* *1* *2* *3* *4*
Array Element→ 31 52 27 5 68

Array [Min]>Array [curr]=?
Yes
Change Min as curr position
Min=3
Max=1

Max Min Curr
Array position → *0* *1* *2* *3* *4*
Array Element→ 31 52 27 5 68

Array [Min]>Array [curr]=?
No

Max Min Curr
Array position → *0* *1* *2* *3* *4*
Array Element→ 31 52 27 5 68

Array [Max]<Array [curr]=?
Yes
Change Max as curr position
Min=3
Max=4
Swap Array [Max] & Array [4]
Swap Array [Min] & Array [0]
After iteration1
Array position → *0* *1* *2* *3* *4*
Array Element→ 5 52 27 31 68

Iteration 2:
Reduce the size of list is n-2 i.e. List start from 1 to 3(1 to n-2)

Fix Min=2, Max=1

Max Min Curr
Array position → *0* *1* *2* *3* *4*
Array Element→ 5 52 27 31 68

Array [Min]>Array [curr]=?
No

Max Min Curr
Array position → *0* *1* *2* *3* *4*
Array Element→ 5 52 27 31 68

Array [Max]<Array [curr]=?
No
Min=2
Max=1
Swap Array [Max] & Array [3]
Swap Array [Min] & Array [1]
After iteration2
Array position → *0* *1* *2* *3* *4*
Array Element→ 5 52 27 31 68

2.4. Pseudo code for DESSA :

In simple Pseudo code for DESSA might be expressed as:

```
Function DESSA (array,size)
1.  var k,i,j,temp,min,max,n,a[300];
2.  k:=n;
3.  for i:=1 to n/2
4.      if array (i)<array(i+1)
5.      min:=i;
6.      max:=i+1;
7.      End if
8.      else
9.      min:=i+1;
10.     max:=i;
11.     End else
12.     for j=i+2 to <k
13.         if array(j)<array(min)
14.         min=j;
15.         else
16.         if array(j)>array(max)
17.         max=j;
18.         End if
19.   End for
```

```
20. if array(i)==array(max)&&array(k-1)==array(min)
21. temp:=array(i);
22. array(i):=array(k-1);
23. array(k-1)=temp;
24. End if
25. else if array(min)==array(k-1)
26. temp:=array(k-1);
27. array(k-1):=array(max);
28. array(max):=temp;
29. temp:=array(max);
30. array(max):=array(i);
31. array(i):=temp;
32. End else if
33. else if array(i)==array(min)&& array(k-1)==array(max)
34. array(i):=array(min);
35. array(k-1):=array(max);
36. End else if
37. else
38. temp:=array(k-1);
39. array(k-1):=array(max);
40. array(max):=temp;
41. temp:=array(min);
42. array(min):=array(i);
43. array(i):=temp;
44. End else
45. k--;
46. End for
```

3. Performance comparisons of algorithms

Correctness of the algorithm is most important. If the correct algorithm takes much time to execute or it takes more memory space, then there is no use to select particular algorithm. So the analysis of the algorithm is mandatory. Two methods are used to analyze an algorithm: [9]

i.) Analytical method
ii.) Experimental method

In analytical method, the factors the time and space requirements of a program depend on are identified and their contributions are determined. But since some of these factors are not known at the time the program is written, an accurate analysis of the time and space requirements cannot be made. Experimental method deals with actually performing experiment and measuring the space and time used by the program.[9]

3.1. Analytical method

Analysis of DESS algorithm:

T(1) = (n-2)
T(2) = (n-2)+(n-4)
T(3) = (n-2)+(n-4)+(n-6)
:
:
:
:
:
T(K-1) = (n-2)+(n-4)+(n-6)+……+(n-(n-2))

Where K=n/2
If n is odd number,
$\sum_{k=1}^{k<n/2} T(K) = \lceil (n/2) \rceil^2 - ((2\lceil (n/2) \rceil) - 1)$
Otherwise
$\sum_{k=1}^{k<n/2} T(K) = (n/2)^2 - (n/2)$
So,
T(K)=O (n^2)

The number of swaps in the proposed algorithm may elaborate as follows:

a) In the best-case scenario; if the input array is already sorted (ascending order), then there is **no** need to make swapping, since each element is in the correct place.

b) In the average-case scenario; if the input array is sorted in reverse (descending order), then the total number of swapping operations is: **n/2**. Since swapping the maximum value with the last element means that the maximum and the minimum values are in the correct places. Total number of swapping operations is: floor of **(n/2)**. Since swapping the maximum value with the last element means that the maximum and the minimum values are in the correct places.

c) In the worst case; if the input array is unsorted neither ascending nor descending, then the required swapping operations are approximately **n** operations.

Criteria	*DESS sort*	Selection Sort
Best Case	O (n^2)	O (n^2)
Average Case	O (n^2)	O (n^2)
Worst Case	O (n^2)	O (n^2)
Memory	*O(1)*	*O(1)*
Stability	Yes	Yes

Number of Swaps	Depends on data: 0, *n/2 or n*	Always *n*
Number of comparison	$(n/2)^2$-(n/2) or $(n/2)^2$	n/2(n-1)

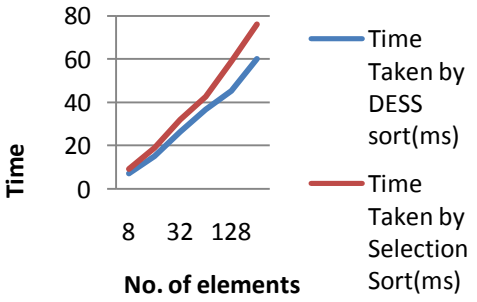
3.2. Experimental method

To prove that DESSA sorting algorithm is relatively faster than SS algorithm, I implement each of them using C, and measure the execution time of both programs with the same input data, and using the same computer. The built-in function (clock ()) in C is used to get the elapsed time of the two algorithms. [5]

```
#include<stdio.h>
#include<time.h>
void main()
{ . . . .
clock_t tStart = clock();
// the function call goes here
printf("Time taken: %.2f\n", (double)(clock() -
tStart)/CLOCKS_PER_SEC);
```

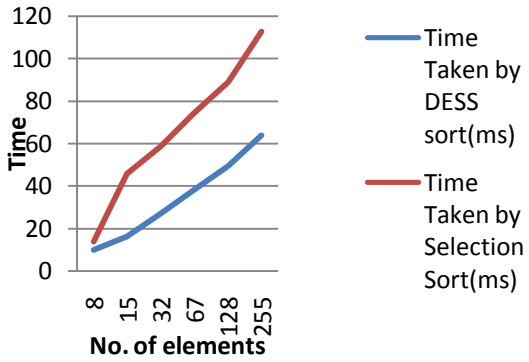
Comparison of already sorted list (ascending) in DESSA &SS:

No. of elements	Time Taken by DESSA (ms)	Time Taken by Selection Sort(ms)
8	*9.97*	*13.9*
15	*16.37*	*45.77*
32	*27.25*	*58.82*
67	*38.36*	*74.56*
128	*49.31*	*89.09*
255	*63.89*	*112.87*



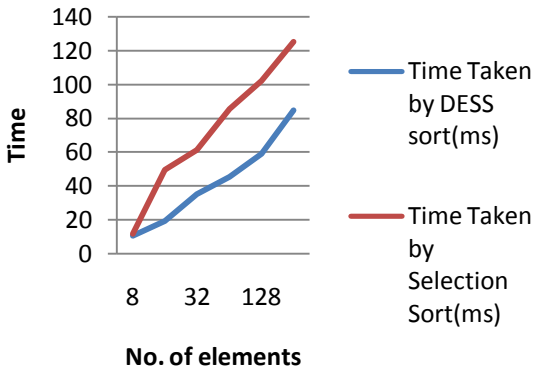
Comparison of unsorted list in DESSA &SS:

No. of elements	Time Taken by DESS sort(ms)	Time Taken by Selection Sort(ms)
8	*7.24*	*9.24*
15	*15.16*	*19.01*
32	*26.54*	*31.88*
67	*36.78*	*42.47*
128	*45.25*	*58.93*
255	*60.14*	*75.96*



Comparison of sorted list in reverse (Descending) in DESSA &SS:

No. of elements	Time Taken by DESS sort(ms)	Time Taken by Selection Sort(ms)
8	*10.84*	*11.85*
15	*19.36*	*49.78*
32	*35.33*	*61.26*
67	*45.47*	*85.32*
128	*59.21*	*102.02*
255	*84.65*	*125.09*



4. Conclusion

In this paper, a DESS sorting algorithm is presented. It has O (n^2) complexity. The number of swapping is lesser than the Selection sort algorithm. It is stable algorithm also. It needs only O (1) space complexity. So the performance of DESS is faster than the existing sorting algorithm. This is proved by analytical and experimental point of view.

References

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

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

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

[4] Deitel H. and Deitel P., "*C++ How to Program"*, Prentice Hall, 2001.

[5] Jehad Alnihoud and Rami Mansi.,"An Enhancement of Major Sorting Algorithms", *The International Arab Journal of Information Technology*, Vol. 7, No. 1, January 2010,pp.55-62.

[6] Knuth D.,"*The Art of Computer Programming"*, Addison Wesley, 1998.

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

[8] Levitin A., "*Introduction to the Design and Analysis of Algorithms"*, Addison Wesley, 2007.

[9] 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, August, 2009

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

[10]Tarundeep Singh Sodhi, Surmeet Kaur, Snehdeep Kaur," Enhanced Insertion Sort Algorithm", nternational Journal of Computer Applications (0975-8887) ,Volume 64,No.21, February 2013.

[11] Yingxu Wang.,"A New Sort Algorithm: Self-Indexed Sort", *Communications of ACM SIGPALN*, Vol.31, No.3, March, 1996, ACM, pp.28-36

[12] Weiss M., "*Data Structures and Problem Solving Using C"*, Addison Wesley, 2002.

Authors:

[1]P.Sumathi is currently working as Assistant Professor in the Department of Information Technology at SNS College of Engineering, Coimbatore. She obtained her B.E. in information technology from Periyar University, 2003. She also obtained her Post Graduate in Computer Science and Engineering from Anna University of Technology, Coimbatore, 2010. Her research interests include computer networks, algorithms design and analysis, image processing. She is a life member of ISTE.

[2]V.V.Karthikeyan is currently working as Associate Professor in the Department of Electrical and Electronics Engineering at SNS College of Engineering, Coimbatore. He obtained Diploma in Computer Technology at Kongu Polytechnic, Perundurai in the year May1997 with 80.4% and B.E. Degree in Electrical and Electronics Engineering at Maharaja Engineering College, Avinashi in the year of May 2000 with 66.7%. He got M.E. Degree at K.S.Rangasamy College of Technology, Tiruchengode in Power Electronics and Drives in the year July2007 with 82.45%.He has 11 years of experience in teaching and 5 years of experience in research. He has published 5 International Journal papers and 2 papers in International Conferences and 8 papers in National Conferences. He has attended 15 workshops/seminars/SDP/winter schools. He is a life member of ISTE, New Delhi. He has guided 5 M.E. research projects and 20 B.E. projects. In his 11 years of teaching experience, he has handled the subjects in the areas of Electrical Machines, Electric Circuits, Power Electronics, Power Quality, Digital signal processing for UG courses and Advanced Power Semi converter devices for PG courses. His area of research interest is implementation of active power filters in industries.