World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:8, No:7, 2014

Enhanced Bidirectional Selection Sort

Jyoti Dua

Abstract**—**An algorithm is a well-defined procedure that takes some input in the form of some values, processes them and gives the desired output. It forms the basis of many other algorithms such as searching, pattern matching, digital filters etc., and other applications have been found in database systems, data statistics and processing, data communications and pattern matching. This paper introduces algorithmic "Enhanced Bidirectional Selection" sort which is bidirectional, stable. It is said to be bidirectional as it selects two values smallest from the front and largest from the rear and assigns them to their appropriate locations thus reducing the number of passes by half the total number of elements as compared to selection sort.

Keywords**—**Bubble sort, cocktail sort, selection sort, heap sort.

I. Introduction

An algorithm plays a vital and key role in solving the computational problems, it is a well defined computational procedure that takes input and produces output [9]. It is said to a tool or a sequence of well defined steps to solve the computational problems [3]. Sorting is an important data structure which finds its place in many real life applications. It has various applications in computer systems, memory management, and file management**.** There are various sorting algorithms that are in existence till date and ample of research is still going on to reduce their time complexity up to the extent possible. Research is also going on for finding the algorithms which are fast enough than the existing ones.

Sort is a significant operation in computer programming. If the data are sorted according to some criteria of order, the efficiency of data handling can be substantially increased. The sorted data is beneficial for record searching, insertion and deletion thus enhancing the efficiency of these operations.

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 [8]. Sort algorithm is one of the elementary techniques in computer science because of the following reasons. First, it is the origin of many other algorithms such as searching, pattern matching, digital filters etc., and various other applications have been found in database systems, data statistics and processing, data communications and pattern matching. Different environment requires different sorting methods.

The formal definition of the sorting problem is as follows:
Input: A sequence having n numbers in any random order (a1, a2, a3… an)

Output: A permutation (a'1, a'2, a'3… a'n) of the input sequence such that a'1 ≤ a'2 ≤ a'3 ≤ ….. a'n

For instance, if the given input of numbers is (59, 41, 31, 41, 26, 58), then the output sequence is returned by a sorting algorithm will be (26, 31, 41, 41, 58, 59) [3].

The common sorting algorithms can be divided into two classes by the complexity of their algorithms-O(n^2) which includes bubble sort, insertion sort, selection sort [6] etc., and O(n log n) which include heap sort [2], quick sort [7], merge sort. Algorithmic complexity is generally written in terms of Big-O notation, where the 'O' represents the complexity of the algorithm and a value n represents the size of the set the algorithm is run against [5], [10].

Two categories of sort algorithms were classified according to the records, whether stored in the main memory or auxiliary memory. One category is the internal sort which stores the records in the main memory. Another is the external sort which stores the records in the hard disk because of the records' large space occupation [1], [8].

II. Problem Statement

The Sorting problem is to arrange a sequence of records so that the values of their key fields form a non decreasing sequence. That is, given records $r_1,r_2,r_3,\ldots,r_n$ with key values $k_1,k_2,k_3,\ldots,k_n$ respectively, we must produce the same record in an order $r_1',r_2',r_3',\ldots,r_n'$ such that $k_1' \le k_2' \le k_3', \ldots, k_n'$.

III. Methodology

The three criteria of sorting techniques are:

Stability: Stable sort keeps records with the same key in the same qualified order that they were in before the sort.

Time: The time related to the swaps and comparisons of elements in the algorithm.

Space: The space may be dependent or independent of the input sequence size. If the additional space needed in the algorithm is independent to the input, its space complexity is O (1), otherwise O (n).

IV. Related Work

Enhanced Bidirectional Selection sort is an improvement and enhancement over Selection sort in terms of stability and efficiency. Enhanced Bidirectional Selection sort sort the elements bidirectional by finding the minimum value from left and maximum from right. At first pass the first minimum value is copied to 1^{st} position and maximum value is copied to n^{th} position of another array say B. Both minimum and maximum value is deleted from the original array say A thus reducing the comparison by the factor of 2.

In second pass second minimum value is copied to 2^{nd} position and second maximum element is copied to the $(n-1)^{th}$

Jyoti Dua is with the United Institute of Technology, Allahabad, India (e-mail: jyotidua_ecimt@yahoo.com).

position of array B. Again both values are deleted from array A. This is repeated until all the elements get copied. This bidirectional property makes the algorithm stable. In this algorithm only comparison takes place and requires no swapping.

A. Algorithm

```
EnhSel(A,B,max,min,loc,loc1,N)
   // A and B are array, max is maximum value, min is minimum
   value
   // loc is location of minimum value, loc1 is location of
   maximum value, N is the number of elements
front←1
rear←N
while front<rear
      loc←1, loc1←N
      min←A[1] and max←A[N]
      for i←2 to N
           if A[i]<min
              min=A[1]
              loc←i
         B[front] ←min
         front←front+1
        for i←loc to N
            A[i] ←A[i+1]
      N=N-1
      for i←N-1 to 1
          if A[i]>max
              max←A[i]
              loc←i
       B[rear] ←max
       rear←rear-1
      for i←loc1 to N
          A[i] ←A[i+1]
       N=N-1
B[front]←A[1]
```

B. Example

Apply Enhanced Bidirectional Selection sort on 10 unsorted elements.

Array ‘A’= 66 17 31 22 56 29 33 13 9 11

Min= A[1] and Max= A[10]

Pass I: front=1 and rear=10

Min=9 Copy Min at positions front=1; increment front=front+1; delete Min from array ‘A’

Max=66 Copy Max at position rear=n; decrement rear=rear-1; delete Max from array ‘A’.

Array ‘B’ contains:

TABLE I
ARRAY ‘B’ AFTER PASS I

Index	Elements
1	9
2	
3	
4	
5	
6	
7	
8	
9	
10	66

Original array ‘A’ remains with following elements:

TABLE II
ORIGINAL ARRAY ‘A’ AFTER PASS I

Index	Elements
1	17
2	31
3	22
4	56
5	29
6	33
7	13
8	11

Pass II: front=2 and rear=9

Min=11 Copy Min at positions front=2; increment front=front+1; delete Min from array ‘A’

Max=56 Copy Max at positions rear=9; decrement rear=rear-1; delete Max from array ‘A’

Array ‘B’ contains:

TABLE III
ARRAY ‘B’ AFTER PASS II

Index	Elements
1	9
2	11
3	
4	
5	
6	
7	
8	
9	56
10	66

Original array ‘A’ remains with following elements:

TABLE IV
ORIGINAL ARRAY ‘A’ AFTER PASS II

Index	Elements
1	17
2	31
3	22
4	29
5	33
6	13

Pass III: front=3 and rear=8

Min=13 Copy Min at positions front=2; increment front=front+1; delete Min from array ‘A’

Max=33 Copy Max at positions rear=9; decrement rear=rear-1; delete Max from array ‘A’

Array ‘B’ contains:

TABLE V
ARRAY ‘B’ AFTER PASS III

Index	Elements
1	9
2	11
3	13
4	
5	
6	
7	
8	33
9	56
10	66

Original array 'A' remains with the following elements:

TABLE VI
ORIGINAL ARRAY 'A' AFTER PASS III

Index	Elements
1	17
2	31
3	22
4	29

Pass IV: front=4 and rear=7

Min=17 Copy Min at positions front=2; increment front=front+1; delete Min from array 'A'

Max=31 Copy Max at positions rear=9; decrement rear=rear-1; delete Max from array 'A'

Array 'B' contains:

TABLE VII
ARRAY 'B' AFTER PASS IV

Index	Elements
1	9
2	11
3	13
4	17
5	
6	
7	31
8	33
9	56
10	66

Original array 'A' remains with the following elements:

TABLE VIII
ORIGINAL ARRAY 'A' AFTER PASS IV

Index	Elements
1	17
2	31

Pass V: front=5 and rear=6

Min=22 Copy Min at positions front=2; increment front=front+1; delete Min from array 'A'

Since front=rear. The one element left with an array 'A' gets copied to the front position in Array 'B'

Therefore, the final result is stored in Array 'B':

TABLE IX
FINAL ARRAY 'B'

Index	Elements
1	9
2	11
3	13
4	17
5	22
6	29
7	31
8	33
9	56
10	66

C. Performance Analysis

For Analysis of Enhanced Bidirectional Selection sort, let time complexity be T (n). Therefore, the time taken by while loop in line 3 of the algorithm is n/2.The time taken to find minimum and maximum element is 2n-2. The time to delete minimum and maximum element is 2n.

Hence, $T(n)=n/2*(2n-2+2n)=O(n^2)$

TABLE X
SORTING COMPLEXITY

Sort	Best Case	Average Case	Worst Case	Space	Stable
Selection	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(1)	No
Heap	O(nlogn)	O(nlogn)	O(nlogn)	O(n)	No
Enhanced Selection	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(n)	Yes

As compared to selection sort Enhanced Bidirectional Selection sort avoids swapping and involves comparisons and assigns the element its correct position in another array.

The total number of comparisons required for finding the minimum element in Selection sort = the total number of comparisons required for finding minimum and maximum value in the Enhanced Bidirectional Selection sort.

While the outer while loop in line number 3 of Enhanced Bidirectional Selection sort algorithm reduces the number of passes by N/2 where N is the number of elements.

For sorting N=10 unsorted elements using Enhanced Bidirectional Selection sort the iteration will be as follows:

TABLE XI
ITERATION FOR FINDING MINIMUM AND MAXIMUM VALUES

Front	Rear	Comparison for finding Min value	Comparison for finding Maximum value
1	10	(10-1)	(9-1)
2	9	(8-1)	(7-1)
3	8	(6-1)	(5-1)
4	7	(4-1)	(3-1)
5	6	(2-1)	-

Using Table II data we get number of comparison as:

$$\{(10-1)+(8-1)+(6-1)+(4-1)+(2-1)\} + \{(9-1)+(7-1)+(5-1)+(3-1)\} = 45$$

TABLE XII
NUMBER OF COMPARISON AND PASSES IN SELECTION SORT

No. of Element	Comparison	No. of Pass
N=10	45	9
N=50	1225	49
N=100	4950	99
N=500	124750	499
N=1000	499500	999

TABLE XIII
NUMBER OF COMPARISON AND PASSES IN ENHANCED BIDIRECTIONAL SELECTION SORT

No. of Element	Comparison	No. of Pass
N=10	45	5
N=50	1225	25
N=100	4950	50
N=500	124750	250
N=1000	499500	500

V. CONCLUSION

Every Sorting problem has its pros and cons. The cocktail [8] is a bidirectional bubble sort which reduces the number of passes as compared to bubble sort [4]. Similarly, Enhanced Bidirectional Selection sort reduces the number of passes and is stable as compared to selection sort. There is no swapping between the two elements, but uses assignment operation to place the element in its right position so it takes memory O (n) compared to selection sort. Enhanced Bidirectional Selection sort takes extra space, but this issue is in less consideration.

REFERENCES

[1] D. Knuth, "The Art of Computer Programming *(*Sorting and Searching)," 3rd ed. vol. 3, Addison-Wesley, 1997, ISBN 0-201-89685-0. pp. 138–14, 1997.
[2] J.W.J Williams, *Algorithm 232: Heap sort*. Comm. *ACM* 7, 6 June 1964, pp.347-348.
[3] Thomas H. Coreman, Charles E. Leiserson and Ronald L. Rivest, *Introduction to Algorithms*. McGraw-Hill, New York.
[4] Aho, A.V.,Hopcroft,J.E. Ullman,J.D.,"Algorithms and Data Structure", Pearson India, Reprint 2000.
[5] Z. Iqbal, H. Gull and A.W. Muzaffar "A New Friends Sort Algorithm", *2nd IEEE Int. Conf. Software Engineering and Information Technology,* ISBN 978-1-4244-4520-2, pp 326-329,2009.
[6] Seymour Lipschutz. Schaum's "Selection Sort (Outline Series Theory and Problems of Data Structures*)"*, Int. ed. McGraw-Hill, 1986. ISBN 0-07-099130-8, pp. 324–325, ch. 9.
[7] Nidhi Chhajed, Simarjeet Singh Bhatia,"A Comparison based Analysis of different types of Sorting Algorithm with their performance",Indian Journal of Research PARIPEX,Vol 2,Issue 3,March 2013.
[8] D.T.V Dharmajee Rao, B.Ramesh, "Experimental Based Selection of Best Sorting", International Journal of Modern Engineering Research (IJMER),vol. 2, no. 4, July-Aug 2012 ISSN 2249-6645 pp-2908-2912.
[9] Eshan Kapur, Parveen Kumar and Sahil Gupta," *Proposal of a Two Way Sorting Algorithm and Performance Comparison with Existing Algorithms*", International Journal of Computer Science, Engineering and Applications (IJCSEA) vol.2, no.3, June 2012.
[10] Jehad Alnihoud and Rami Mansi, "*An Enhancement of Major Sorting Algorithms*", The International Arab Journal of Information Technology, Vol. 7, no. 1, January 2010.

Jyoti Dua completed her MCA from Ewing Christian Institute of Management and Technology and perusing her M.Tech from United Institute of Technology Allahabad, India. Her areas of interests are Algorithms, Data Mining, and Computer Networks.