LEFT-RIGHT SORT: A NOVEL SORTING ALGORITHM AND COMPARISON WITH INSERTION SORT, BUBBLE SORT AND SELECTION SORT

Shashank Singh*

Rajesh Shyam Singh**

Hardwari Lal Mandoria***

ABSTRACT

Sort is an algorithm that arranges all elements of an array in increasing or decreasing order. Sorting Technique is frequently used in so many important applications to arrange the data in ascending or descending order. Several Sorting Algorithms of different-different time and space complexity are exist and used. This paper provides a novel sorting algorithm Left-Right Sort in which element can be inserted from both side of array and the side will be decided by the algorithm. Comparison of Left-Right sort with Insertion Sort, Selection Sort and Bubble Sort also shown in this paper. Visual Studio 2008 Tool and C# language are used for implementation and analysis of CPU time taken, number of comparison and number of swaps. Analysis is based on random input sequence of length 100, 300, 500, 800, 1000, 1500, 2000, 3000, 4000, 5000, 8000, 10000, 15000, 20000, 30000, 40000, 50000. Result shows that Left-Right sort working well for all length of input values and CPU time and number of comparison also minimized.

Keywords: Bubble Sort, Insertion Sort, Selection Sort, Left-Right Sort, Sorting, Number of comparison, Number of swaps.

* Research Scholar, Department of Information Technology, G.B. Pant University of Agriculture and Technology, Pantnagar-263145 UK, INDIA

** Asst. Professor, Department of Information Technology, G.B. Pant University of Agriculture and Technology, Pantnagar-263145 UK, INDIA

*** Professor and Head, Department of Information Technology, G.B. Pant University of Agriculture and Technology, Pantnagar-263145 UK, INDIA

1. Introduction

An algorithm is a finite sequence and well-defined set of instructions that takes some value or set of values as input and produces some value as a output **Herbert Schild, 2005**. A good algorithm is that which provides satisfactory result for every range of data set. Sorting is a very basic concept and important for solving other problems like is prerequisite for Binary Search technique. Sorting is the fundamental problem of computer science and remained burning issue for re-search over the last several decades due to time complexity **Alfred V.et.al, 2002**. Sorting is often used in a large variety of critical applications and is a fundamental operation that is used by most computers. There are two general categories of sorting algorithms: algorithms that sort random-access objects, such as arrays or random-access disk files, and algorithms that sort sequential objects, such as disk or tape files, or linked lists. There are three general methods for sorting arrays: **Herbert Schild, 2005.**

- Exchange
- Selection
- Insertion

Sorting has been considered as a fundamental problem in the study of algorithms, that due to many reasons:

- The need to sort information is inherent in many important applications.
- Algorithms often use sorting as a key subroutine and efficient sorting is important to optimize the use of other algorithms that require sorted lists to work correctly.

The output should satisfy two major conditions:

- The output is a permutation, or reordering, of the input sequence.
- The output is in some order increasing or decreasing.

This paper provides a novel sorting algorithm Left-Right Sort and Left-Right sort compared with Bubble Sort, Selection Sort and Insertion sort.

The remainder of this paper is organized as follows. Section 2 introduces the overview of several existing sorting techniques; Section 3 describes our proposed Left-Right Sorting technique;

Section 4 describes comparative study of our proposed technique with other existing techniques; Conclusion and future scope in Section 5 and all the used references are given in section 6.

2. Overview Of Several Sorting Techniques

2.1 Bubble Sort

Bubble sort works in the following process: keep passing through the list, exchanging adjacent element, if the list is in unordered form; when the list is in sorted order, no exchange of data element are required. With the Bubble Sort, the number of comparisons is always the same because the two for loops repeat the specified number of times whether the list is initially ordered or not. This means that the bubble sort always performs

$$\frac{1}{2}(n^2 - n)$$

Comparisons, where n is the number of elements to be sorted **Trivedi.D et al, 2013**. Bubble sort average case and worst case complexity are both $O(n^2)$ but best case complexity is O(n).

2.2 Selection Sort

A Selection sort selects the element with the lowest value and exchanges it with the first element. Then, from the remaining n-1 elements, the element with the smallest key is found and exchanged with the second element, and so forth. The exchanges continue to the last two elements. The asymptotic complexity of basic selection sort in worst case, average case and best case is $O(n^2)$ which is due to comparisons of each data element with each other and its number of iterations. So these algorithms can be improved or enhanced by reducing the number of iterations or comparisons. As like Bubble sort, Selection sort also perform.

$$\frac{1}{2}(n^2 - n)$$

Number of comparison, where n is the number of elements to be sorted **Khairullah.M, 2013**.

2.3 Insertion Sort

Insertion Sort works as follows: We insert an element into its proper place in the previous sub-list. At each iteration, we identify two regions, sorted region and unsorted region. We take one element from the unsorted region and "insert" it in the sorted region. The element in the sorted region will increase by one after the iteration. We repeat this on the rest of unsorted region.

Insertion sort worst case and best case complexity is $O(n^2)$ and best case complexity is O(n) **Sodhi.T.S et al, 2013**.

Unlike selection sort and Bubble sort number of comparison and number of swaps in insertion sort cannot be explained in terms of number of element to be sorted. Same size of list may take different-different number of comparison and swaps. Because number of comparison and swaps are depend on the values which we are going to sort. But among Selection, Bubble and Insertion sort, Insertion sort is the fastest sorting algorithm. Like Selection sort and Bubble sort, Insertion sort is also an in-place sorting technique. Because Insertion sort, Bubble sort and Selection sort uses O(1) extra space to sort the list on n element.

3. Novel Proposed Left-Right Sort Algorithm

Left-Right sort uses approach unlike selection sort and bubble sort, But Left-Right sort is little bit similar to Insertion sort algorithm. Like Insertion sort in Left-Right sort algorithm element taken from the list one by one and inserted into its proper place in the previous sub-list. At each iteration, we find two regions, sorted region and unsorted region. We take again one element from the unsorted region and insert it in the sorted region. The element in the sorted region will increase by one after each iteration. We have to repeat this process on the rest of unsorted region till the end of unsorted region. But for the insertion of one element from the unsorted region, Left-Right sort use different approach than insertion sort. In Left-Right sort, when an element taken for the insertion in sorted region then two subtraction operation are calculated. Firstly Left-Right algorithm will subtract element (element to be inserted) with the lower bound of array (minimum value in the sorted region), after that Left-Right will subtract element (element to be inserted) with the last value in the sorted region (maximum value in the sorted region). Finally insert element from that side of sorted region where subtraction value is minimum. That means Left-Right sort is bidirectional. By using two subtraction operations in each iteration, The Number of comparison will be minimized and in turn CPU time will also minimized. Subtraction operation takes O(1) time that means subtraction does not affect the complexity of the algorithm. Left-Right sort uses the following algorithm:

Algorithm: Left-Right Sort (array[n]) /* array is set of total n input elements */

```
Int max;
Int min;
If (array[1] < array[0])
{
   Int temp=array[0];
   array[0]=array[1];
   array[1]=temp;
}
max=array[1];
min=array[0];
For( int a=2; a< n; a++)
{
  Int leftdiffrence= array[a]-min;
  Int rightdiffrence= array[a]-max;
  If ( leftdiffrence >= rightdiffrence)
   {
      Int j=a;
     While( j>0)
     {
         If ( array[j-1] > array[j])
           {
             Int temp= array[j-1];
             array[j-1] = array[j];
             array[j] = temp;
             J--;
          }
     }
     max = array[a];
     min = array[0];
  }
else
```

```
{
    Int k=0;
    While(k <=a)
     {
          If ( array[k] > array[a] )
            {
              Int temp = numarray[a];
              Int y = a;
              Int p = k;
 While (y > p)
{
  Array[y] = array[y-1];
  y--;
}
Array[p] = temp;
}
Else
{
K++;
}
}
Max = array[a];
Min = array[0];
}
}
```

Table 1: Left-Right Sort for the input values 6, 5, 4, 3, 2, 1

Initial	6	5	4	3	2	1
Iteration1	5	6	4	3	2	1
Iteration2	4	5	6	3	2	1
Iteration3	3	4	5	6	2	1
Iteration4	2	3	4	5	6	1
Iteration5	1	2	3	4	5	6

4. Comparative Study and Discussion

All the four sorting algorithm (Insertion Sort, Selection Sort, Bubble Sort and Left-Right Sort) were implemented in Visual Studio 2008 tool and C# language used for coding. All four algorithms tested for the random sequence input of length 100, 300, 500, 800, 1000, 1500, 2000, 3000, 4000, 5000, 8000, 10000, 15000, 20000, 30000, 40000, 50000. All the four algorithm were executed on machine with 64-bit Operating System having Intel(R) Core(TM) i3-2330m CPU @ 2.20 GHz and installed memory (RAM) 2.00 GB.

The Plot of length of input and CPU time taken (sec) are shown in figure 1. Results shows that for the small input sequence length, the performance of all four algorithm is almost same, but for the large input sequence length Left-Right Sort is faster than Insertion sort, Selection Sort and Bubble Sort.

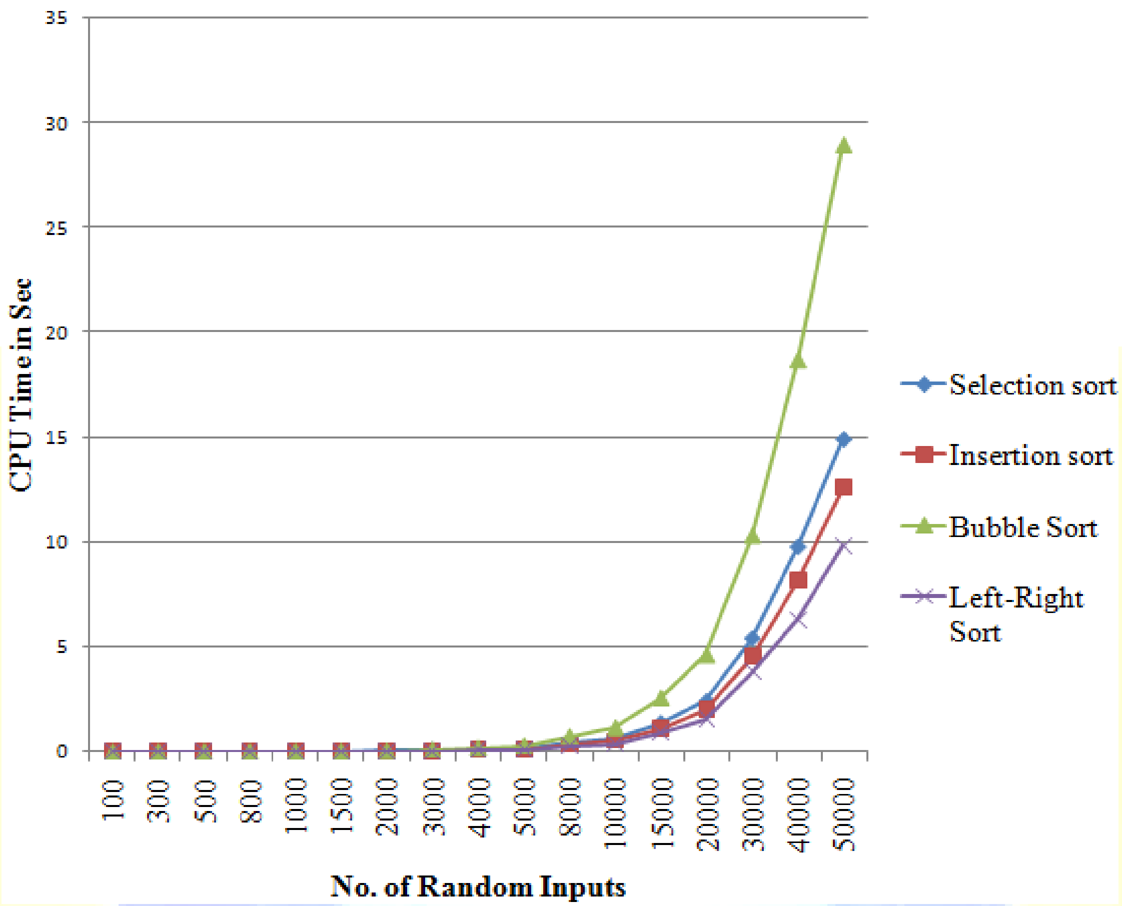

Figure 1: Plot of Number of Input vs. CPU time (sec)

CPU time (sec) for Insertion sort, Selection Sort, Bubble Sort and Left-Right Sort on same length of input sequence are represented in Table.2.

Table 2: CPU time taken in seconds

No of random inputs	CPU time for insertion sort	CPU time for selection sort	CPU time for bubble sort	CPU time for Left-right sort
100	.0041747	.0039815	.0036830	.0076935
300	.0053391	.0055373	.0074057	.0069060
500	.0075298	.0088705	.0100503	.0096976
800	.0110826	.0129221	.0149882	.0067316

1000	.0109343	.0215659	.0183913	.0075307
1500	.0161405	.0222059	.0257677	.0125857
2000	.0246163	.0511771	.0430301	.0185387
3000	.0476419	.0831204	.0969466	.0386441
4000	.0828451	.1186902	.1769451	.0672318
5000	.1271134	.1738065	.2779363	.1010415
8000	.3240337	.4144372	.7223528	.2542343
10000	.5042733	.6230625	01.1359573	.3948422
15000	01.1356560	01.3678546	02.5609767	.8822711
20000	02.0342127	02.4092226	04.5977310	01.5731188
30000	04.5379053	05.4154803	10.3245431	03.8025235
40000	08.1462073	09.8008539	18.6987054	06.3050336
50000	12.6300708	14.9390762	28.9814682	09.8261071

The Plot of length of input and Number of Comparison are shown in figure 2. Result shows that almost for all length of input sequence, the performance of Left-Right Sort is better than Insertion Sort, Selection Sort and Bubble Sort. To Sort a list, mostly Left-Right Sort requires less number of comparisons in comparison to Insertion, Selection and Bubble Sort. Number of comparison in bubble sort and selection sort are always equal, that’s why both are shown by a single series in graph.

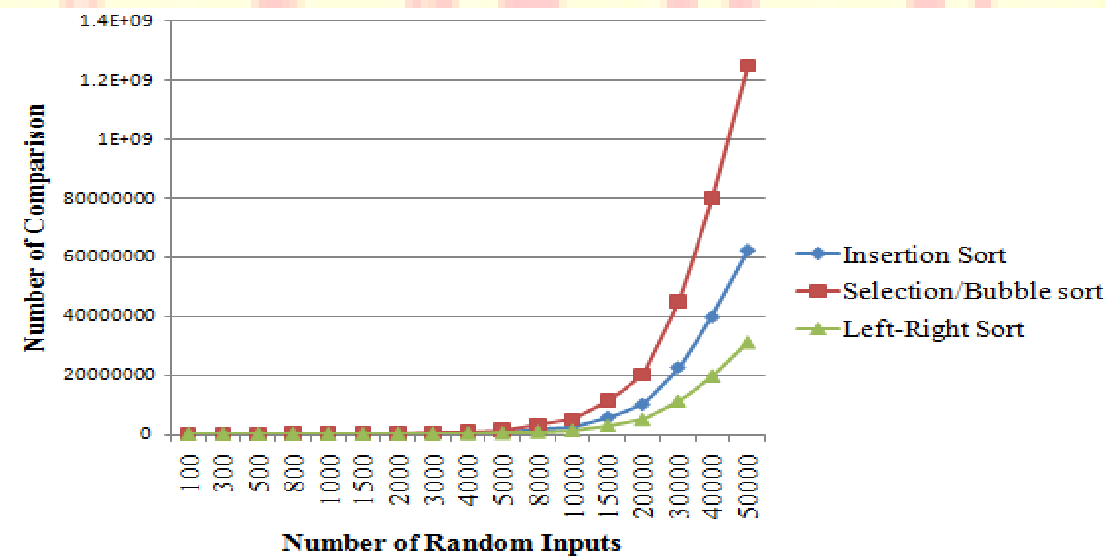

Figure 2: Plot of Number of Input vs. Number of Comparison

Number of comparison's in Insertion sort, Selection Sort, Bubble Sort and Left-Right Sort on same length of input sequence are represented in Table.3.

Table 3: Number of Comparison's

No of random inputs	No of comparison in Insertion sort	No of comparison in selection sort/bubble sort	No of comparison in left-right sort
100	2588	4950	1440
300	22138	44850	11584
500	61846	124750	30306
800	164973	319600	81560
1000	261430	499500	125196
1500	570231	1124250	276423
2000	970542	1999000	500121
3000	2223368	4498500	1151499
4000	4048557	7998000	1991607
5000	6224849	12497500	3095366
8000	16005189	31996000	8002482
10000	25101828	49995000	12430375
15000	56399836	112492500	28312913
20000	100073315	199990000	49657890
30000	226171380	449985000	113199891
40000	399627257	799980000	198576174
50000	625939083	1249975000	313443017

Numbers of swaps for N input values are always same in Left-Right Sort, Insertion sort and Bubble Sort but for Selection Sort they may be less or more than other three algorithms, because Selection sort always perform (N-1) swaps where n is the number of input to be sort. Number of swaps for input length 5000 and 2000 are shown in figure 3 and figure 4.

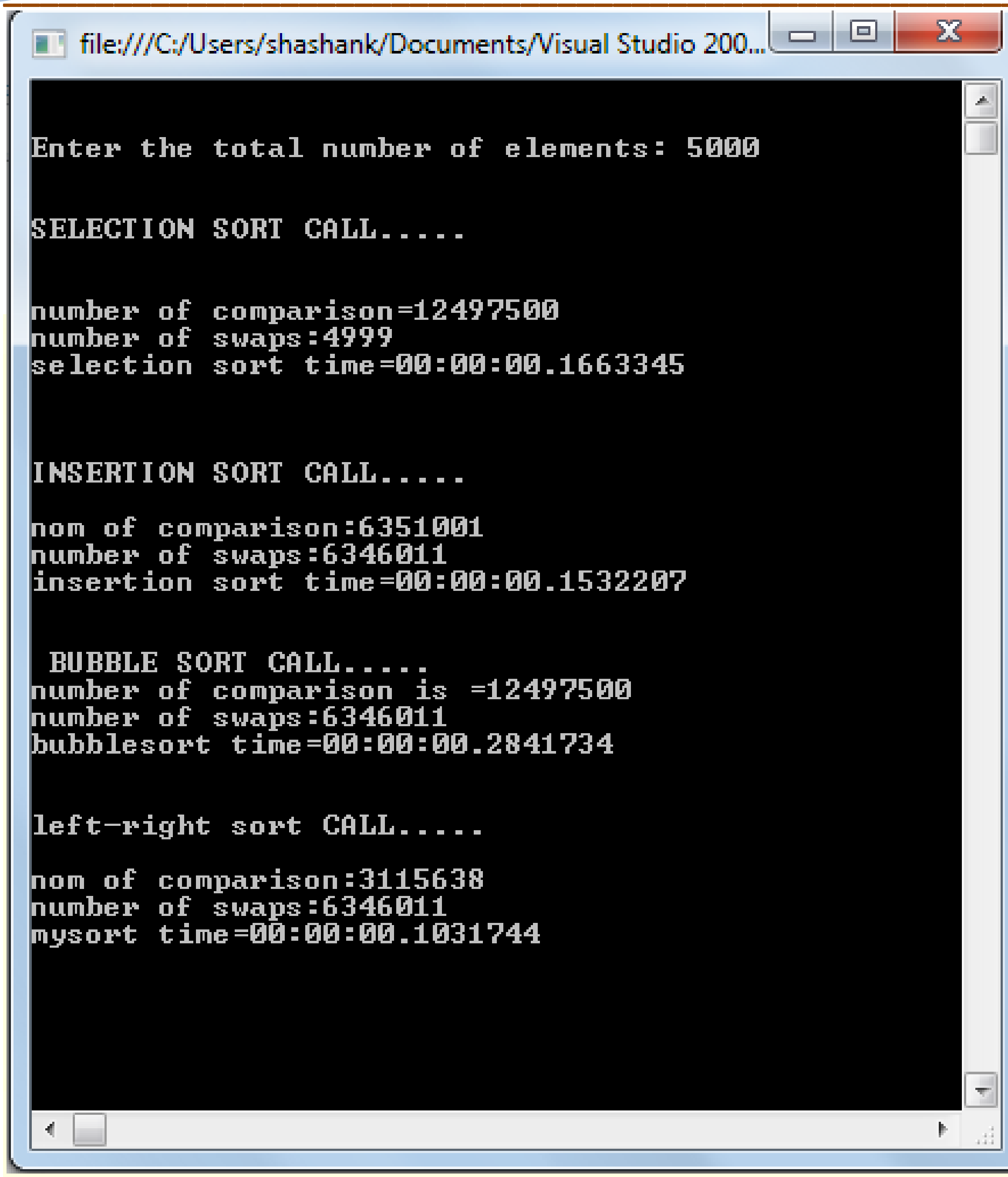

Figure 3: Number of Swaps for 5000 Random Values

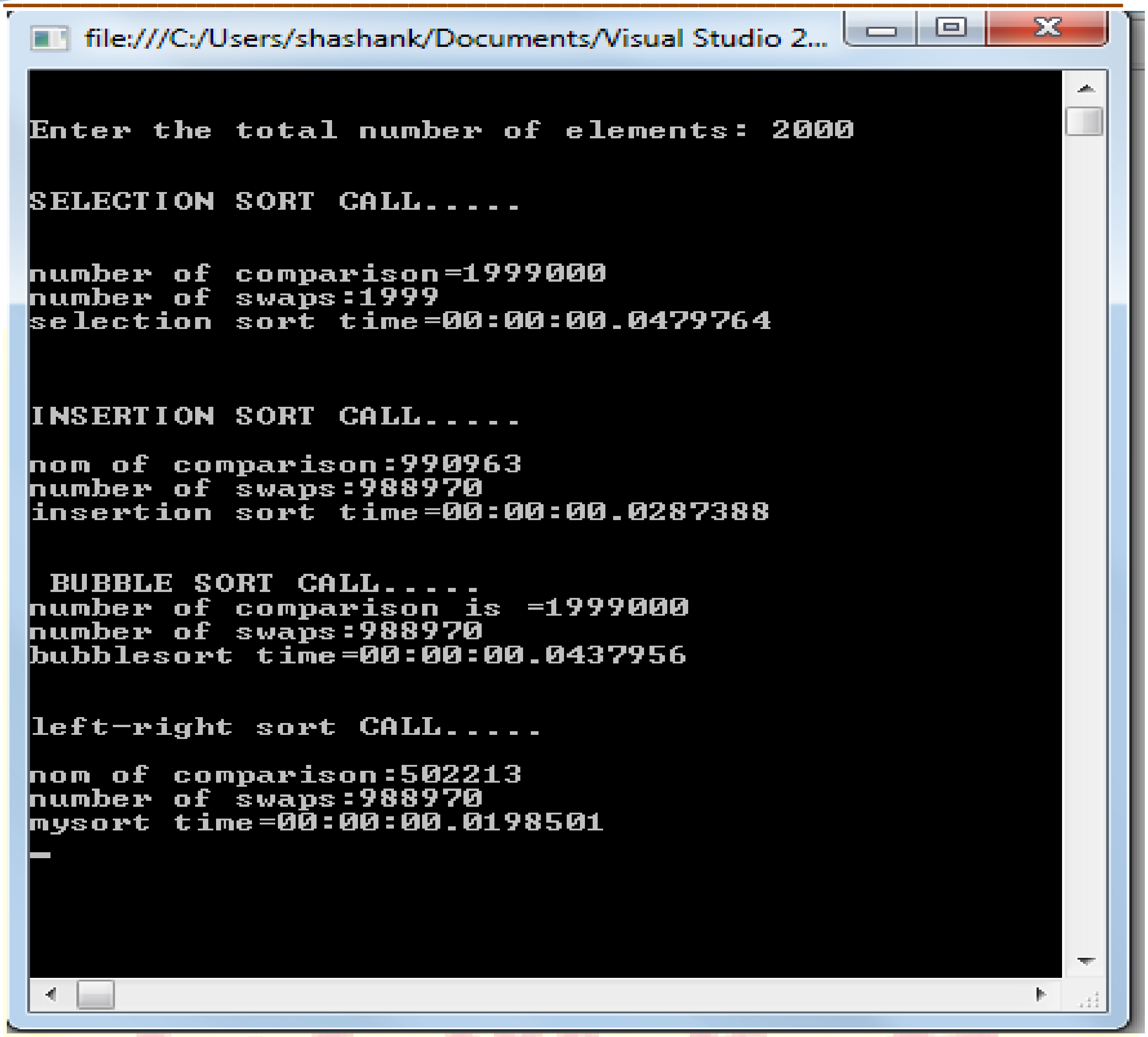

Figure 4: Number of Swaps for 2000 Random Values

5. Conclusion and Future Work

Sorting is a technique that arranges all element of an array in ascending or descending order. Result shows that new Left-Right sorting algorithm is working well for all length of input values. And mostly it takes smaller C.P.U time and less number of comparisons than Insertion sort, Selection sort and Bubble sort. And number of swaps is always same as Bubble Sort and Insertion Sort.

In future work we can make it more efficient in terms of C.P.U time and number of comparisons and can compare with other existing algorithm and can also calculate its time complexity.

References

[1] Srivastava, R., Tiwari, T., & Singh, S. (1899, December). Bidirectional Expansion-Insertion Algorithm for Sorting. In *icetet* (pp. 59-62). IEEE.

[2] Alfred V., Aho J., Horroroft, Jeffrey D.U. (2002) Data Structures and Algorithms.

[3] Herbert Schildt Tata McGraw-Hill [2005], "The Complete Reference C fourth Edition".

[4] Arora, N., Kumar, S., & Tamta, V. K. (2012). A Novel Sorting Algorithm and Comparison with Bubble sort and Insertion sort. International Journal of Computer Applications, 45(1), 31-32.

[5] Kapur, E., Kumar, P., & Gupta, S. (2012). Proposal of a two way sorting algorithm and performance comparison with existing algorithms. International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol, 2.

[6] Trivedi, D., Trivedi, P., & Singh, S. (2013) Min-Max Select Bubble Sorting Algorithm. International Journal of Applied Information Systems (IJAIS) – ISSN: 2249-0868.

[7] Beniwal, S., & Grover, D. (2013). Comparison of various sorting algorithms: A review. International Journal of Emerging Research in Management &Technology, ISSN: 2278-9359 (Volume-2, Issue-5).

[8] Khairullah, M. (2013). Enhancing Worst Sorting Algorithms. International Journal of Advanced Science and Technology Vol. 56.

[9] Sodhi, T. S., Kaur, S., & Kaur, S. (2013). Enhanced Insertion Sort Algorithm. International Journal of Computer Applications, 64(21), 35-39.

[10] Dutta, P. S.(2013) An Approach to Improve the Performance of Insertion Sort Algorithm. International Journal of Computer Science & Engineering Technology (IJCSET) Vol. 4 No. 05.