Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.017

IJCSMC, Vol. 5, Issue. 11, November 2016, pg.158 – 166

Review on Execution Time of Sorting Algorithms- A Comparative Study

Jyoti Totla
Assistant Professor, Department of Computer Science Engineering, Mewar University, Chittorgarh, India
totlajyoti@gmail.com

ABSTRACT: Sorting is an important data structure in many real life applications. A number of sorting algorithms are in existence till date. In this paper the author have tried to improve upon execution time of the Bubble Sort algorithm by implementing the algorithm using a new algorithm. An extensive analysis has been done by author on the new algorithm and the algorithm has been compared with the traditional methods of —Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort. Observations have been obtained on comparing this new approach with the existing approaches of All Sorts. All algorithms were tested on random data of various ranges from small to large. It has been observed that the new approach has given efficient results in terms of execution time. Hence we have reached to the conclusion through the experimental observations that the new algorithm given in this paper is better than Selection Sort, Insertion Sort, Merge Sort, and Bubble Sort except Quick Sort for larger inputs.
Keywords: Bubble Sort; Insertion sort; Quick Sort; Merge Sort; Optimized Bubble Sorting; CPU Time.

I. INTRODUCTION

There are two basic categories of sorting methods [1]:

A. INTERNAL SORTING: If all the data that is to be sorted can be adjusted at a time in main memory, then internal sorting methods are used. The various internal sorting methods are: Bubble sort, Insertion sort, Quick Sort, Merge Sort, Heap Sort, and Radix Sort [1].

B. EXTERNAL SORTING: When the data to be sorted can't be accommodated in the memory at the time and some has to be kept in auxiliary memory (hard disk, floppy, tape etc), then external sorting method are used. The most common used external sorting method is: Merge Sort [1]

As stated in [2], 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 applications.

• Algorithms often use sorting as a key subroutine.

• In algorithm design there are many essential techniques represented in the body of sorting algorithms.

• Many engineering issues come to the fore when implementing sorting algorithms. Efficient sorting is important to optimize the use of other algorithms that require sorted lists to work correctly; it is also often in producing human-readable output. Formally, the output should satisfy two major conditions[2]:

• The output is in non-decreasing order.

• The output is a permutation, or reordering, of the input.

Since the early beginning of computing, the sorting problem has attracted many researchers, perhaps due to the complexity of solving it efficiently. Bubble sort was analyzed as early as 1956 [5]. Many researchers considered sorting as a solved problem. Even so, useful new sorting algorithms are still being invented, for example, library sort was first published in 2004. Sorting algorithms are prevalent in introductory computer science classes, where the abundance of algorithms for the problem provides a gentle introduction to a variety of core algorithm concepts [1, 13].

In [4], they classified sorting algorithms by:

• Execution time: Some algorithms take less CPU time and some takes More CPU time.

• Number of swaps (for in-place algorithms).

• Stability: stable sorting algorithms maintain the relative order of records with equal keys (values). Ex. when there are two records R and S with the same key and with R appearing before S in the original list, R will appear before S in the sorted list.

• Usage of memory and other computer resources. Some sorting algorithms are "in place", such that only O(1) or O(log n) memory is needed beyond the items being sorted, while others need to create auxiliary locations for data to be temporarily stored.

• Whether or not they are a comparison sort. A comparison sort examines the data only by comparing two elements with a comparison operator. In this paper we have proposed a slight variation to Bubble sort by introducing a new approach for implementing the bubble sort. We

wanted to design a stable sorting algorithm which could sort Maximum Number of elements in every pass.

II. PERFORMANCE IN AVERAGE CASE BETWEEN SORTING ALGORITHMS

A. Insertion Sort

It is an efficient algorithm for sorting a small number of elements. The insertion sort works just like its name suggests, inserts each item into its proper place in the final list. Sorting a hand of playing card is one of the real time examples of insertion sort. Insertion sort can take different amount of time to sort two input sequences of the same size depending upon how nearly they already sorted. It sort small array fast but big array very slow[4].

TABLE 1: Execution Time for Insertion Sort.

Number of elements	Running time (ms)
10000	1605
20000	3678
30000	6125

B. Selection Sort

It is among the most intuitive of all sorts. The basic rule of selection sort is to find out the smallest elements in each pass and placed it in proper location. These steps are repeated until the list is sorted. This is the simplest method of sorting. In this method, to sort the data in ascending order, the 0th element is compared with all the elements. If 0 th element is greater than smallest element than interchanged. So after the first pass, the smallest element is placed at the 0th position. The same procedure is repeated for 1th element and so on until the list is sorted[4].

TABLE 2: Execution Time for Selection Sort.

Number of elements	Running time (ms)
10000	2227
20000	5058
30000	8254

C. Merge Sort

Merging means combining two sorted list into one sorted list. The unsorted list is first divide in two half. Each half is again divided into two. This is continued until we get individual numbers. Then pairs of number are combined (merged) into sort list of two numbers. Pairs of these lists of four numbers are merged into sorted list of eight numbers. This is continued until the one fully sorted list is obtained [4].

TABLE 3: Execution Time for Merge Sort

Number of elements	**Running time (ms)**
10000	728
20000	1509
30000	2272

D. Quick Sort

In general, quick sort can sort a list of data elements significantly faster than any of the common sorting algorithms. This algorithm is based on the fact that it is faster and easier to sort two small arrays than one larger one. Quick sort is based on divide and conquer method. Quick sort is also known as partition-exchange sort[4]. One of the elements is selected as the partition element known as pivot element. The remaining items are compared to it and a series of exchanges is performed. When the series of exchanges is done, the original sequence has been partitioned into three sub sequences.

1. All items less than the pivot element
2. The pivot element in its final place
3. All items greater than the pivot element

At this stage, step 2 is completed and quick sort will be applied recursively to steps 1 and 3 The sequence is sorted when the recursion terminates.

TABLE 4: Execution Time for Quick Sort

Number of elements	Running time (ms)
10000	489
20000	1084
30000	1648

E. Bubble sort

Because of bubble sort's simplicity, it is one of the oldest sorts known to man. It based on the property of a sorted list that any two adjacent elements are in sorted order. In a typical iteration of bubble sort each adjacent pair of elements is compared, starting with the first two elements, then the second and the third elements, and all the way to the final two elements. Each time two elements are compared, if they are already in sorted order, nothing is done to them and the next pair of elements is compared. In the case where the two elements are not in sorted order, the two elements are swapped, putting them in order [5].

TABLE 5: Execution Time for Bubble Sort.

Number of elements	Running time (ms)
10000	1133
20000	3103
30000	5730

F. PROPOSED OPTIMIZED BUBBLE SORT ALGORITHM

In this sorting algorithm, the sorting is done in two phases:

1) Phase 1: initially Leftmost bound is set at 0th position and Rightmost bound is set at last position i.e. Length of array - 1. Leftmost and rightmost elements are compared if leftmost is larger than rightmost then swapped .then leftmost bound get increased and right most bound get decreased .The main loop gets repeated “n/2” times provided “n” is the Length of array. After every iteration the distance between Leftmost bound and Rightmost bound goes on decreasing. In this array get optimized[14].

2) Phase 2: In this phase bubble sort algorithm is applied from left to right up to middle element and from right to left up to middle element. After these arrays got sorted[14].

3) Algorithm

```
proposed_Bubble_Sort ( A [ ] , N )
 Step1: Set Mid=N/2, S=0
Step 2 : Repeat For J= 0 to Mid-1 Begin
Step 3: If ( A [ J ] >A [ N- J -1] )
          Swap ( A [ J ] , A [ N-J – 1 ] )
          End For
Step 4: Repeat For P = 0 to Mid-1
 Begin
Step 5 : Repeat For J = 0 to Mid
 Begin
 Step 6: If ( A [ J ] > A [ J +1 ] )
          Swap ( A [ J ] , A [ J +1 ] ) S=1
          If ( A [ N-J -1] < A [ N-J -2 ] )
         Swap ( A [ N-J -1] , A [ N-J -2] )
          S=1
         End For
         IF (S=0 )
         Break
         Set S=0
          End For
 Step 7: Exit
```


TABLE 6: Execution Time for Proposed Optimized Bubble Sort

Number of elements	Running time (ms)
10000	240.0
20000	960
30000	2150

III. COMPARATIVE STUDY AND DISCUSSION

All the six sorting algorithms (Selection Sort, Insertion sort, Merge sort, Quick sort, Bubble Sort and proposed optimized bubble sort) were implemented it in C language on GCC complier on Linux operating system implemented in C++ programming languages and tested for the random sequence input of length 10000, 20000, 30000. To calculate the execution time of both algorithms, clock () function is used. The Plot of length of input and CPU time taken (ms) is shown in figure1. Result shows that for small input the performance for the six techniques is all most nearest, but for the large input Quick sort is the fastest and the selection sort the slowest. Proposed optimized bubble sort is faster than other sorting algorithms except Quick Sort for larger input length (more than 30000).

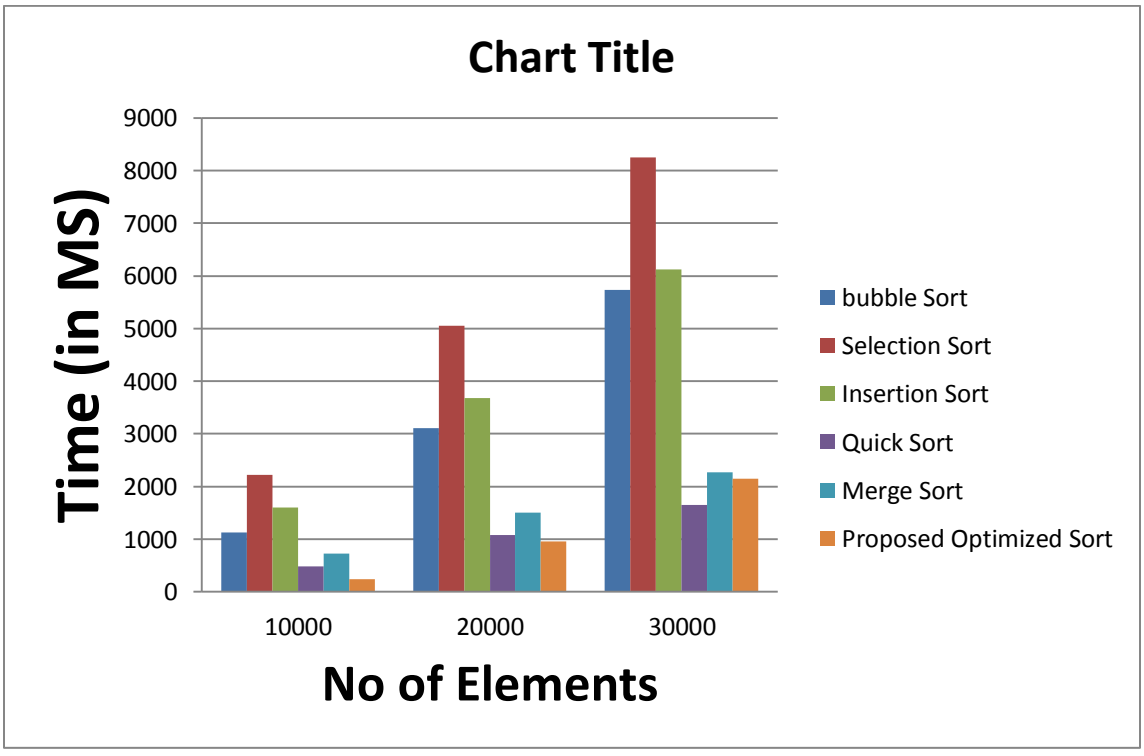

Fig.1 Graph of Number of Input vs CPU time(in sec)

IV. CONCLUSION

In this paper, efforts are made to point out some deficiencies in earlier work related to all five traditional sorting algorithms (Selection Sort, Insertion sort, Merge sort, Quick sort, Bubble Sort. By going through all the experimental results and their analysis it is concluded that the proposed new algorithm is very efficient than existing bubble sort, Selection Sort, Insertion sort, Merge sort, Quick sort, algorithms .It is reducing the Execution time. The benefits of this algorithm can be used in various areas like sorting of contact lists, Web Browsers, etc. One of the advantages is that it is a stable sort which can be used in all applications where similar valued keys are used in databases, same last name but different first name of people.

REFERENCES

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

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

[3] Knuth, D. The Art of Computer Programming, Vol. 3: Sorting and Searching, Third edition. Addison- Wesley, 1997. ISBN 0-201-89685-0. pp. 106-110 of section.

[4] Aho A., Hopcroft J., and Ullman J., The Designand Analysis of Computer Algorithms, Addison Wesley, 1974.

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

[6] D.Sharma,V.Thapar,R.A.Ammar,S.Rajasekaran,M.Ahme d, "Efficient sorting algorithms for the cell broadband engine,"Computers and Communications,2008.ISCC 2008.IEEE Symposium.

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

[8] 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 and computer sciences IJECS-IJENS VOL. 11, No. 01 pp. 16- 22.

[9] Dhwaneel Trivedi, Prathmesh Trivedi, Suraj Singh,"Min-Max Select Bubble Sorting Algorithm" ,International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868 Foundation of Computer Science FCS, New York, USA International Conference & workshop on Advanced Computing 2013 (ICWAC 2013) – www.ijais.org.

[10] Vipul Sharma "A New Approach to Improve Worst Case Efficiency of Bubble Sort" International Research Journal of Computer Science (IRJCS) ISSN: 2393-9842 Issue 6, Volume 2 (June 2015) www.irjcs.com.

[11] Ramesh M. Patelia, Shilpan D. Vyas, Parina S. Vyas "An Analysis and Design of Optimized Bubble Sort Algorithm".IJRIT International Journal of Research in Information Technology, Volume 3, Issue 1, January 2015, Pg. 65-68

[12] Data Structures by Seymour Lipschutz, Schaum's outlines, The MacGraw Hill Companies.

[13] Levitin A., "Introduction to the Design & Analysis of Algorithms", 2nd Ed. Pearson Educational, 2007.

[14] Arora Nitin, Kumar vivek and Kumar Suresh. "A Novel Sorting Algorithm and Comparison with Bubble Sort and Insertion Sort," International Journal of Computer Applications (0975-8887) vol. 45, No. 1, May 2012.

[15] Jyoti Mundra, Mr. B. L. Pal." Minimizing Execution Time of Bubble Sort Algorithm", International Journal of Computer Science and Mobile Computing, Vol.4 Issue.9, September-2015, pg. 173-181

