Analysis of Sorting Algorithms Using Time Complexity

Shubham V, Ganmote,
2nd Semester, Dept. of MCA
R.V. College of Engineering,
Mysore Road, Bengaluru-59,
Affiliated to VTU, Belagavi

Vishwas G R S
2nd Semester, Dept. of MCA
R.V. College of Engineering,
Mysore Road, Bengaluru-59,
Affiliated to VTU, Belagavi

Anupama Kumar
Associate Prof., Dept. of MCA
R.V. College of Engineering,
Mysore Road, Bengaluru-59,
Affiliated to VTU, Belagavi

***Abstract:* Analysis of algorithms is one of the important phase in developing a project. The algorithms can be analyzed using two methods : Space and Time complexity. Sorting algorithms are used worldwide to arrange the data / files for efficient working. This paper discusses about the different sorting algorithms and their analysis using time complexity. The different sorting techniques like bubble sort, selection sort, insertion sort, quick sort and merge sort are implemented using C. The input values varying from 100 to 1000 are system generated. The time complexity of these algorithms are calculated and recorded. For the given data set, quick sort is found very efficient and has taken 168 ms for 1000 data inputs. The algorithm is in place and not stable since it takes extra memory space to divide and combine the solution.**

I. INTRODUCTION

Sorting algorithm in computer science, is an algorithm that puts elements of a list in a certain order. The most used orders are numerical ordering and lexicographical ordering. Efficiently sorting a list is important for optimizing the use of other algorithms (such as search and merge algorithms) which require input data to be stored in lists.

There are a lot of sorting algorithms available for sorting the given data or file. Some algorithms are efficient for some inputs and some are not. The efficiency or performance of an algorithm depends on the time and space complexity of the algorithm. The space complexity of an algorithm is the amount of memory it needs to run to completion. The time complexity of an algorithm is the amount of computer time it needs to run to completion. [1]. Any sorting algorithm should satisfy the following properties (i) The output must be
sorted, and (ii) It must still contain the same elements.[4]. In [8] the various sorting technique using the stability and the time efficiency. In this paper we discuss the efficiency of the algorithms using the number of inputs and the time taken to sort the elements. This paper deals with the following sorting techniques:

Bubble Sort: Bubble sort uses brute force method to sort the elements. Bubble sort is also called as sinking sort, is a simple sorting algorithm that repeatedly steps through the list to be sorted, it compares each pair of adjacent items and swaps them if they are in the wrong order. This algorithm is stable; since it only swaps two items if the latter one is strictly greater, so equal-valued items will stay in their original order. When the elements are already in sorted order, bubble sort takes minimum time. It can be applied to smaller set of data [1]

Selection sort: It is implemented using Brute force technique. The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning of the array. The algorithm maintains two sub arrays in a given array. The first sub array is the array which is already sorted. And the second is the remaining sub array which is unsorted. One of the application of selection sort is to find the unique element in a data set.[3]

Insertion Sort: Decrease and conquer method is used to sort the elements using insertion sort. Insertion sort is a simple sorting algorithm that builds the final sorted array (or list) one item at a time. The algorithm is more useful if the first few objects are already sorted, an unsorted object can be inserted in the sorted set in proper place[5].

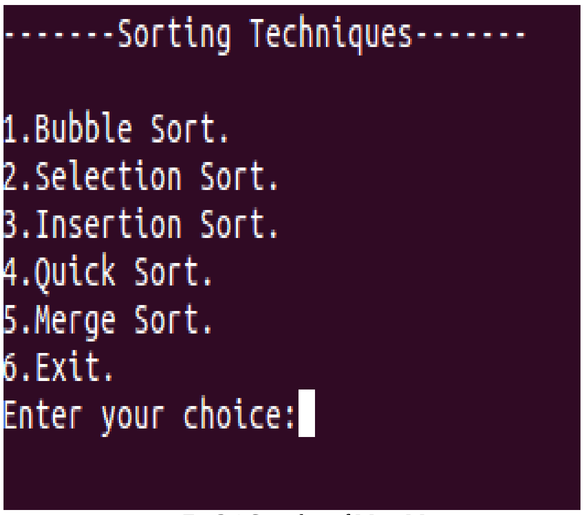
Quick Sort: Quick-sort uses Divide and Conquer technique. It picks an element as pivot element and partitions the given array around the picked pivot element. There are many different versions of quick-sort that picks the pivot element in different ways. Always pick first element as pivot element. Always pick last element as pivot elements. Pick a random element as pivot element. Pick median as pivot element[1]. The sorting technique is best suited to sort large number of data. This performs better than MergeSort and HeapSort which has same asymptotic time complexity O(n log n) on average case but the constant factors hidden in the asymptotic time complexity for quick sort are pretty small[6]. The algorithm is implemented in medical monitoring system, Google pages for fast retrieval, life support or control systems etc.

Merge Sort: Merge Sort uses Divide and Conquer algorithm. It divides the input array into two halves, calls itself for the two sorted halves and then merges the two sorted halves. [1]. The author in [2] has discussed the optimum solution of merge sort and explained the sequence in detail.

The following sections discuss about the implementation of the algorithms.

II. IMPLEMENTATION

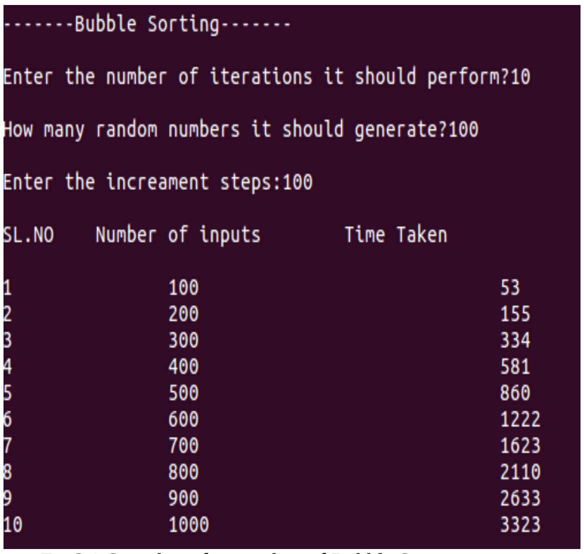
C programming is used in this work for demonstrating the sorting algorithm. The input is varied from 100 to 1000 and generated automatically in the program. The figure 3.1 shows the main menu of the implementation:



```
-------Sorting Techniques-------

1.Bubble Sort.
2.Selection Sort.
3.Insertion Sort.
4.Quick Sort.
5.Merge Sort.
6.Exit.
Enter your choice:
```

Fig 3.1 Snapshot of Main Menu

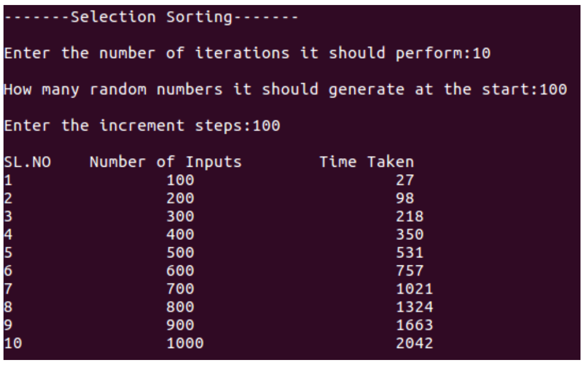
The figure 3.2 shows the implementation of bubble sort technique.The time taken for sorting 100 inputs is 53 Millie-seconds and the time taken for sorting 1000 inputs is 3323 Millie-seconds. As the number of inputs increases, the time taken to sort the data also increases.



```
-------Bubble Sorting-------

Enter the number of iterations it should perform?10

How many random numbers it should generate?100

Enter the increament steps:100

SL.NO    Number of inputs        Time Taken

1             100                            53
2             200                            155
3             300                            334
4             400                            581
5             500                            860
6             600                            1222
7             700                            1623
8             800                            2110
9             900                            2633
10            1000                           3323
```

Fig 3.2 Snapshot of time taken of Bubble Sorting program

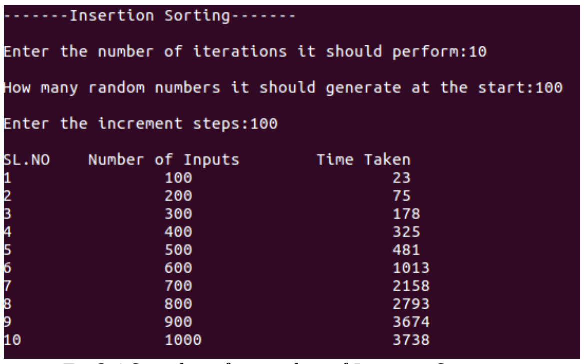
The figure 3.3 shows the implementation of selection sort technique.The time taken for sorting 100 inputs is 27 Millie-seconds and the time taken for sorting 1000 inputs is 2842 Millie-seconds. As the number of inputs increases, the time taken to sort the data also increases.



```
-------Selection Sorting-------

Enter the number of iterations it should perform:10

How many random numbers it should generate at the start:100

Enter the increment steps:100

SL.NO    Number of Inputs        Time Taken
1               100                  27
2               200                  98
3               300                  218
4               400                  350
5               500                  531
6               600                  757
7               700                  1021
8               800                  1324
9               900                  1663
10              1000                 2042
```

Fig 3.3 Snapshot of time taken of Selection Sorting program

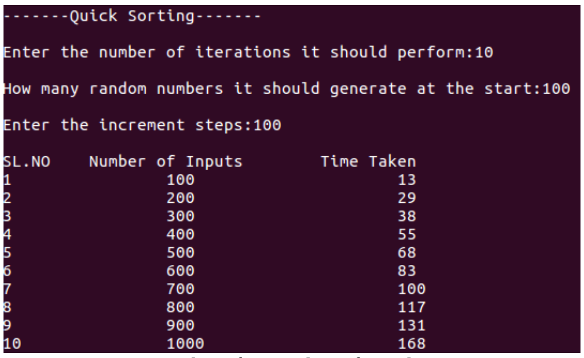
The figure 3.4 shows the implementation of insertion sort technique.The time taken for sorting 100 inputs is 23 Millie-seconds and the time taken for sorting 1000 inputs is 3738 Millie-seconds. As the number of inputs increases, the time taken to sort the data also increases.



```
------Insertion Sorting-------

Enter the number of iterations it should perform:10

How many random numbers it should generate at the start:100

Enter the increment steps:100

SL.NO    Number of Inputs        Time Taken
1               100                  23
2               200                  75
3               300                  178
4               400                  325
5               500                  481
6               600                  1013
7               700                  2158
8               800                  2793
9               900                  3674
10              1000                 3738
```

Fig 3.4 Snapshot of time taken of Insertion Sorting program

The figure 3.5 shows the implementation of Quick sort technique.The time taken for sorting 100 inputs is 13 Millie-seconds and the time taken for sorting 1000 inputs is 168 Millie-seconds. As the number of inputs increases, the time taken to sort the data also increases.

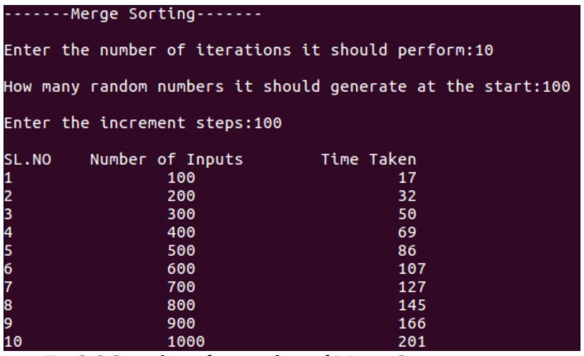


```
------Quick Sorting-------

Enter the number of iterations it should perform:10

How many random numbers it should generate at the start:100

Enter the increment steps:100

SL.NO    Number of Inputs        Time Taken
1               100                  13
2               200                  29
3               300                  38
4               400                  55
5               500                  68
6               600                  83
7               700                  100
8               800                  117
9               900                  131
10              1000                 168
```

Fig 3.5 Snapshot of time taken of Quick Sorting program

The figure 3.6 shows the implementation of merge sort technique.The time taken for sorting 100 inputs is 17 Millie-seconds and the time taken for sorting 1000 inputs is

201 Millie-seconds. As the number of inputs increases, the time taken to sort the data also increases.

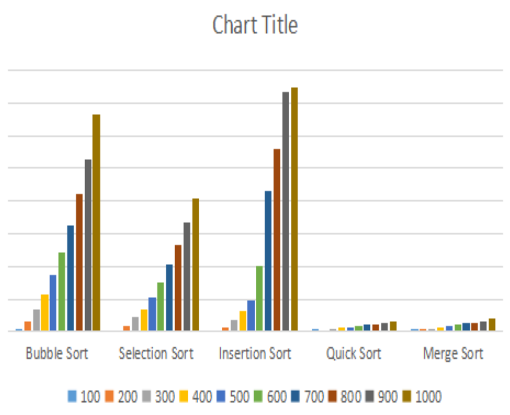


```
-------Merge Sorting-------

Enter the number of iterations it should perform:10

How many random numbers it should generate at the start:100

Enter the increment steps:100

SL.NO    Number of Inputs        Time Taken
1                100                     17
2                200                     32
3                300                     50
4                400                     69
5                500                     86
6                600                     107
7                700                     127
8                800                     145
9                900                     166
10               1000                    201
```

Fig 3.6 Snapshot of time taken of Merge Sorting program

III CONCLUSION

The below graph shows the time taken by the sorting algorithms. The graph shows that the quick sort and merge sort are more efficient compared to the bubble,selection and insertion sorting techniques, When there is more number of inputs the quick sort and merge sort take less time to sort compared to the other sorting techniques.



REFERENCES:

[1] Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran. Fundamentals of Computer Algorithms second edition, Introduction, 2007
[2] www2.latech.edu/~choi/Bens/Teaching/Development/Algorithm/PDF/CH04.pdf
[3] www8.cs.umu.se/kurser/TDBA77/VT06/algorithms/
[4] www- bcf.usc.edu/~dkempe/CS104/10-31.pdf
[5] http://www.personal.kent.edu/ rmuhamma
[6] SadaKurupathi, Quicksort – a practical and efficient sorting algorithm, https://sadakurapati.wordpress.com
[7] Marcin Sydow, Algorithms and Data structures
[8] https://www.saylor.org/site/wp-content/uploads/2011/06/Sorting-Algorithm.pdf