International Journal of Recent Trends in Engineering & Research
ISSN(ONLINE) : 2455-1457
IMPACT FACTOR : 3.344

# Performance Visualization of Bubble Sort in Worst Case Using R Programming in Personal Computer

**Dipankar Das[1], Priyanka Das[2], Rishab Dey[3], Sreya Modak[4]**
[1]*Assistant Professor, The Heritage Academy, Kolkata, India*
[2,3,4]*Student, The Heritage Academy, Kolkata, India*

**Abstract—**The present work attempts to perform a visual analysis of the performance of Bubble sort in the worst case in a personal computer (laptop). The said algorithm is implemented using R programming language and the run time of the Bubble sort in the worst case for the data size under study have been recorded. For the purpose of the study, 100 observations for each data size are recorded. For each data size, the researchers have calculated four measures namely minimum time, maximum time, average time and median time. These four measures are visualized using scatter plots and line charts and they are also visually compared with quadratic curves. It has been observed that the performances of Bubble sort in all the four cases are approximately similar and approximately follows the quadratic curves.

**Keywords—**Bubble sort, Worst case, Scatter plot, Line charts, Algorithm visualization

## I. INTRODUCTION

In general, the performance analysis of any algorithm can be done either by calculating the space complexity of the algorithm or by calculating the time complexity of that algorithm. We know that the Bubble sort is one of the simplest sorting algorithms. In the present study, the researchers have done the performance visualization of Bubble sort in the worst case. Here, the researchers have taken the run time of the algorithm as the performance measure. In the present day, the field of data visualization has gained tremendous importance. In this work, the researchers have used data visualization techniques for the empirical analysis of Bubble sort algorithm in the worst case and the Bubble sort algorithm has been implemented using R programming language.

## II. RELATED WORK

While going through the literatures on the Bubble sort algorithm, we have seen that the recent works on the said algorithm can be classified into some broad categories, some of which are as follows: (a) History and origin of Bubble sort, (b) Enhancement of Bubble sort or new variants of Bubble sort and (c) Comparison of sorting algorithms.

(a) History and origin of Bubble sort: Astrachan (2003) had traced the history of Bubble sort in his work which includes origins and performance [1].

(b) Enhancement of Bubble sort or new variants of Bubble sort: Khairullah (2013) had done enhancement of Bubble sort, Selection sort and Insertion sort [2]. Rohil & Manisha (2014) had proposed Run Time Bubble Sort algorithm and had compared the proposed algorithm with Bubble sort algorithm [3]. Appiah and Martey (2015) had proposed an enhancement of Bubble sort algorithm called Magnetic Bubble sort algorithm which performs better in cases of redundancies [4]. Mundra and Pal (2015) in their work had shown that the execution time of Bubble sort algorithm can be improved by using a new algorithm [5]

(c) Comparison of sorting algorithms: Al-Kharabsheh, AlTurani, AlTurani and Zanoon (2013) had compared the run time for Selection sort, Insertion sort, Merge sort, Quick sort, Bubble sort and

 

Grouping comparison sort using C++ [6]. Sareen (2013) had compared Bubble sort, Selection sort, Insertion sort, Merge sort and Quick sort on the basis of their run time using C sharp language [7]. Kocher and Agrawal (2014) had analysed the performances of different comparison based sorting algorithm including Bubble sort [8]. Alam and Chugh (2014) had compared Bubble sort, Insertion sort, Selection sort, Quick sort, Shell sort and Heap sort on the basis of number of comparisons, number of swap and time taken using C++ [9].

## III. OBJECTIVES OF THE STUDY

To visualize the performance (Data size versus Run time in seconds) of the Bubble sort algorithm implemented using R programming language in the worst case in a personal computer (laptop).
To visually compare the performance (Data size versus Run time in seconds) of Bubble sort algorithm implemented using R programming language in the worst case in a personal computer (laptop) with quadratic curve.

## IV. METHODOLOGY

Step 1: Implementation of Bubble Sort algorithm in R programming language

Step 2: Observation of run time of Bubble sort in the worst case for data size one hundred (100) to two thousand (2000) with an interval of one hundred (100). One hundred observations will be recorded for each data size. The unit of time is second

Step 3: Calculation of the (i) minimum, (ii) maximum, (iii) average and (iv) median values of run time of Bubble sort in the worst case for each data size i.e. from one hundred (100) to two thousand (2000)

Step 4: Visualization of the performance of Bubble sort i.e. (i) Data size (x axis) versus Minimum value of run time (y axis), (ii) Data size (x axis) versus Maximum value of run time (y axis), (iii) Data size (x axis) versus Average value of run time (y axis) and (iv) Data size (x axis) versus Median value of run time (y axis) using simple scatter plots and line charts

Step 5: Visual comparisons between the performance curves of Bubble sort and quadratic curves

**System Information:**
Processor: Intel(R) Core(TM) i7-4702MQ CPU @ 2.20 GHz
RAM: 8.00 GB
System type: 64-bit operating system, x64-based processor

## V. DATA ANALYSIS

The scatter plots of the performances of Bubble sort in the worst case (Data size versus run time in seconds) are given below (Figure 1):

 

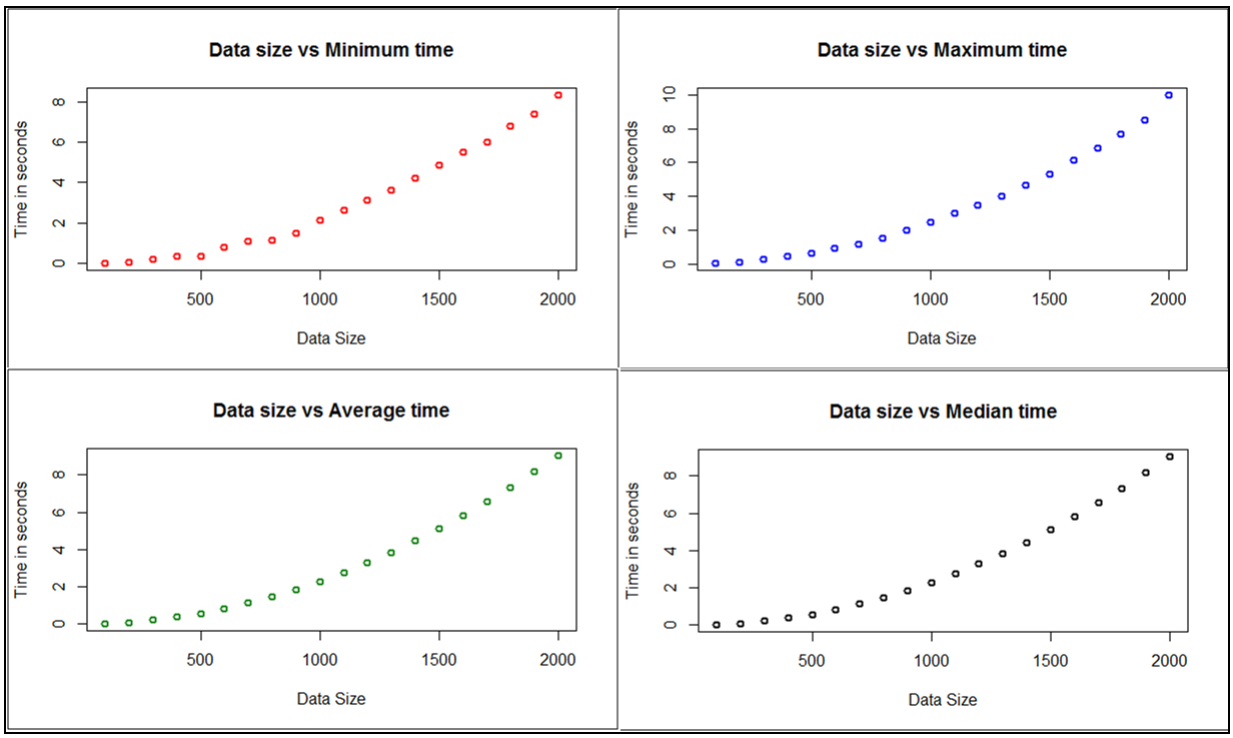


*Figure 1. Scatter plots of the performances of the Bubble sort in the worst case*

The line charts of the performances of Bubble sort in the worst case (Data size versus run time in seconds) are given below (Figure 2):

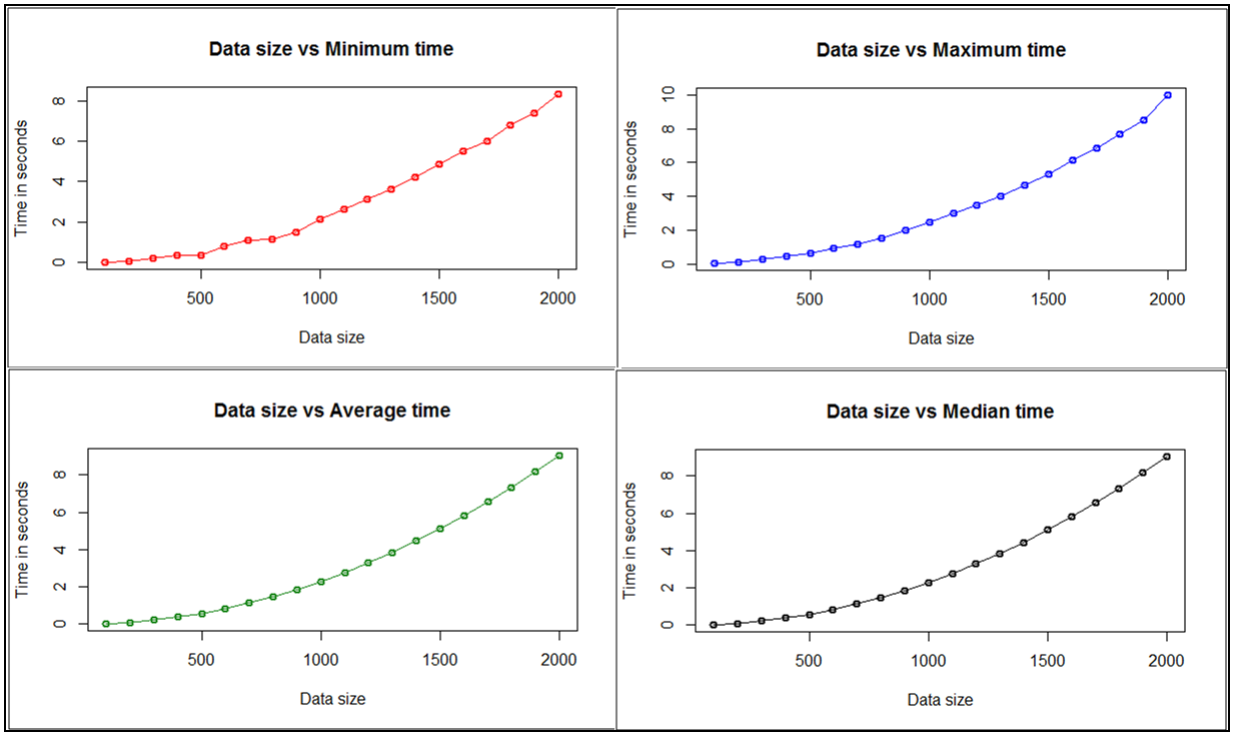


*Figure 2. Line charts of the performances of the Bubble sort in the worst case*

The red, blue, green and black circles in the above two figures (Figure 1 and Figure 2) represent the minimum, maximum, average and median measures of the one hundred (*100*) observations of the run

time of Bubble sort in the worst case for each data size starting from one hundred (*100*) to two thousand (*2000*) with an interval of one hundred (*100*).

The scatter plots of the performances of Bubble sort in the worst case (Data size versus run time in seconds) along with quadratic curves are given below (Figure 3):

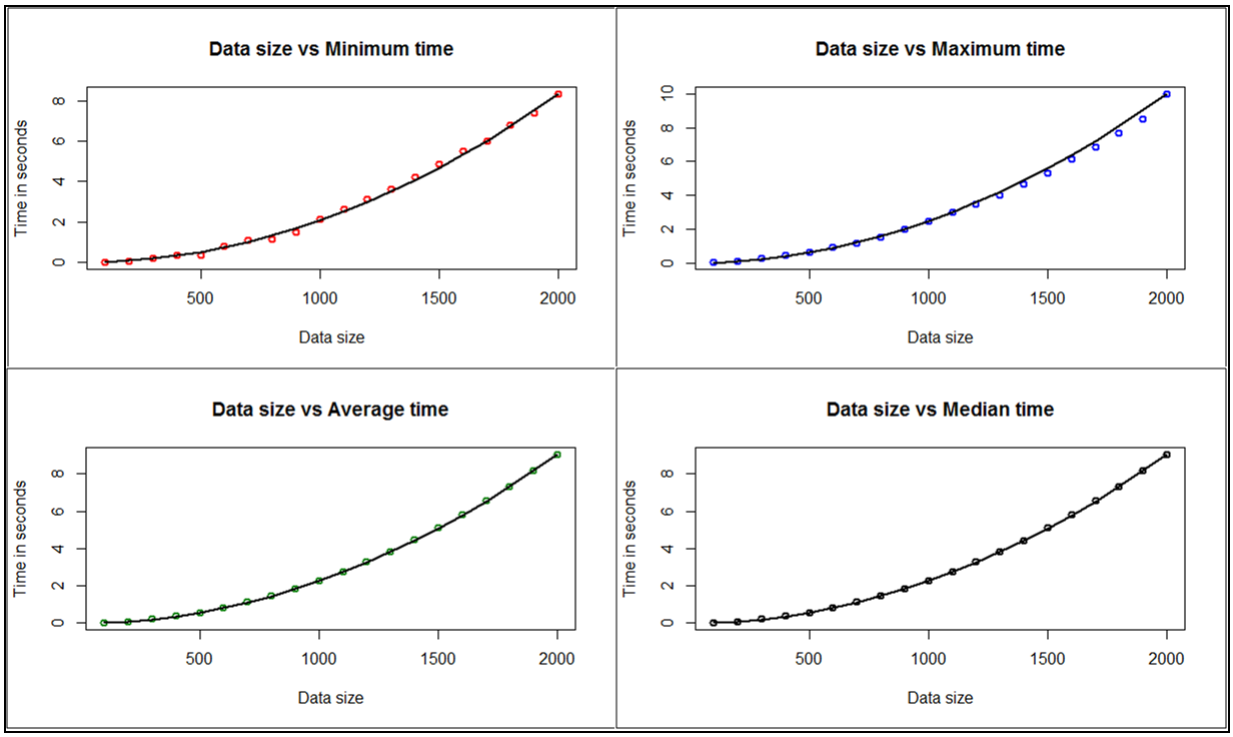


***Figure 5. Scatter plots of the performances of the Bubble sort in the worst case along with quadratic curves***

The dark black curves in all the four cases i.e. Data size versus minimum time, Data size versus maximum time, Data size versus average time and Data size versus median time represent four (*4*) different quadratic curves. Each of these quadratic curves is obtained by taking the top most point of each graph as the starting point.

## VI. CONCLUSION

In the present study, the main objectives of the researchers are (i) to visualize the performance of Bubble sort in the worst case scenario in a personal computer (laptop) where the said algorithm will be implemented using R programming language and (ii) to perform the visual comparative analysis of the performance of Bubble sort with the quadratic curve to understand and/or interpret the result. From the above two figures (Figure 1 and Figure 2) we observe that all the four types of measures (i.e. minimum, maximum, average and median) show almost identical shapes. When all these measures are plotted against quadratic curves, we observe that in all the four (*4*) cases the performance of Bubble sort in the worst case (implemented using R programming language in the personal computer under study) approximately follows the quadratic curve which is clearly evident from the Figure 3. In this study, we have not tried to find the best possible curve which can be fitted to the data points. The study is carried out only on one personal computer (laptop) and with a relatively small set of data. Therefore, at this point we cannot predict what will happen if the same experiment is carried on either large set of data or on different computers and getting suitable answers of these possibilities will be our future scope of work.

 

## REFERENCES

1. Astrachan, O. (2003, February). Bubble sort: an archaeological algorithmic analysis. In *ACM SIGCSE Bulletin* (Vol. 35, No. 1, pp. 1-5). ACM. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.8.9358&rep=rep1&type=pdf
2. Khairullah, M. (2013, July). Enhancing Worst Sorting Algorithms. *International Journal of Advanced Science and Technology, 56*, 13-26. Retrieved from http://www.sersc.org/journals/IJAST/vol56/2.pdf
3. Rohil, H., & Manisha. (2014, August). Run Time Bubble Sort – An Enhancement of Bubble Sort. *International Journal of Computer Trends and Technology, 14*(1), 36-38. Retrieved from http://www.ijcttjournal.org/Volume14/number-1/IJCTT-V14P109.pdf
4. Appiah, O., & Martey, E. M. (2015). Magnetic Bubble Sort Algorithm. *International Journal of Computer Applications IJCA, 122*(21), 24-28. doi:10.5120/21850-5168
5. Mundra, J., & Pal, B. L. (2015, September). Minimizing Execution Time of Bubble Sort Algorithm. *International Journal of Computer Science and Mobile Computing, 4*(9), 173-181. Retrieved from http://ijcsmc.com/docs/papers/September2015/V4I9201531.pdf
6. Al-Kharabsheh, K. S., AlTurani, I. M., AlTurani, A. M. I., & Zanoon, N. I. (2013). Review on Sorting Algorithms A Comparative Study. *International Journal of Computer Science and Security, 7*(3), 120-126. Retrieved from http://www.cscjournals.org/manuscript/Journals/IJCSS/Volume7/Issue3/IJCSS-877.pdf
7. Sareen, P. (2013, March). Comparison of Sorting Algorithms (On the Basis of Average Case). *International Journal of Advanced Research in Computer Science and Software Engineering, 3*(3), 522-532. Retrieved from https://www.ijarcsse.com/docs/papers/Volume_3/3_March2013/V3I3-0319.pdf.
8. Kocher, G., & Agrawal, N. (2014, March). Analysis and Review of Sorting Algorithms. *International Journal of Scientific Engineering and Research, 2*(3), 81-84. Retrieved from http://www.ijser.in/archives/v2i3/SjIwMTMxODE=.pdf
9. Alam, M., & Chugh, A. (2014, March). Sorting Algorithm: An Empirical Analysis. *International Journal of Engineering Science and Innovative Technolog, 3*(2), 118-126. Retrieved from http://www.ijesit.com/Volume 3/Issue 2/IJESIT201402_16.pdf

 