ISSN 2278-3091
Volume 10, No.2, March - April 2021
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse761022021.pdf
https://doi.org/10.30534/ijatcse/2021/761022021

# Comparison of Bubble and Insertion Sort in Rust and Python Language

**Fanila Ali Agha[1], Haque Nawaz[2]**
[1]BSCS Student, Sindh Madressatul Islam University, Karachi, Sindh, Pakistan, fanilaali99@gmail.com
[2]Sindh Madressatul Islam University, Karachi, Sindh, Pakistan, hnlashari@smiu.edu.pk

## ABSTRACT

Sorting is the basic activity in the field of computer science and it is commonly used in searching for information and data. The main goal of sorting is to make reports or records easier to edit, delete and search, etc. It organizes the given data in any sequence. There are many sorting algorithms like insertion sort, bubble sort, radix sort, heap sort, and so forth. Bubble sort and insertion sort are clearly described with algorithms and examples. In this paper, the bubble sort and insertion sort performance analysis is carried out by calculating the time complexity. These algorithm time complexities have been calculated by implementing in the rust and python languages and observed the best case, average case, and worst case. The flowchart shows the complete workflow of this study. The results have been shown graphically and time complexity has been shown in a tabular form. We have compared the efficiency of bubble sort and insertion sort algorithms in the rust and python platforms. The rust language is more efficient than python for both bubble and insertion sort algorithms. However, it is observed insertion sort is more efficient than the bubble sort algorithm.

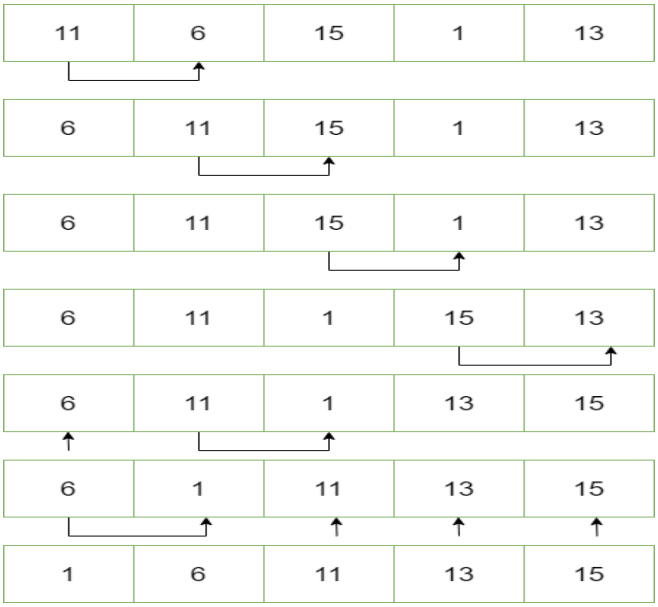
**Key words:** Sorting, Algorithm, Insertion Sort, Bubble Sort, Analysis, Best Case, Worst Case, Average Case, Time Complexity.

## 1. INTRODUCTION

Sorting algorithms are used to rearrange any data in some order [1][2][3]. There are many algorithms but none of them is considered the best solution for all the problems. When the series of activities are performed in a sequence that takes some input and after completing a number of the steps it returns some output that is known as an algorithm. A good algorithm is the one that outgrows appropriate output for a set of given inputs, it is also the one that gives us the desired output in the fewest number of steps, and a good algorithm should be written in a way that every person could understand it easily. Sorting is an elementary problem of computer science [4][5], and it remains a burning issue in the field of research for many years because of its time complexity. Sorting is a very basic element and it is most crucial for solving many problems like finding any component, database algorithm, image rendering, and a lot more [6]. Many algorithms are known for solving ordered lists like: bubble sort, merge sort, quick sort, and insertion sort [7].

### 1.1 Bubble Sort

Bubble sort is the ancient method of sorting, it is the simplest sorting method. Bubble sort compares one element of the array with the next element, if it needs a swap then it would swap both the elements with each other. The algorithm repeats the same process until the whole list of elements is in the proper order [8][9], bubble sort is also known as an adjacent comparison [10], bubble sort makes the list in a way that the smallest elements sink and the larger element of array bubbles at the end of the list. This process is also noted as the exchange method [1], When the bubble sort is applied on a sorted array or list its complexity is represented in O(n). When the bubble sort is applied on an array or list of the average case its complexity can be represented by O(n2). When the bubble sort is applied on the worst-case its complexity is represented by O(n2) [11].



### 1.2 Insertion Sort

Insertion sort is a simple sorting technique that makes an algorithm in any language and then sorts the array. It is most efficient on a small number of elements [12][13]. Insertion sort is also known as the example of the incremental algorithm [14].

Insertion sort could also be compared with the person playing UNO cards, people usually place cards in order the same work is done by the insertion sort it is also the real-life exam of insertion sort. Insertion works with some rules its first rule, key is always the second digit of the list in comparison to any two numbers. Its second rule is to mark the unsorted list, let's assume that the unsorted list starts from the second element of the list till the last element of the list then compare it with the sorted part of the list, let's assume that the sorted part of the list is its first element, If the swap is required then swap else increase the sorted list's part by moving assumed sorted lists to the right. Repeat the same process until the whole array is sorted.


## 2. LITERATURE REVIEW

Sorting is a basic problem used in computer science [15]. In a research paper comparison of two sorting techniques is shown that is bubble sort and Insertion sort on two different programming languages, is java and C programming language [16]. Time required by insertion sort is O(n2) while the time required by one iteration of insertion sort is O(n) [17][18] Time complexity of bubble sort is O(n2) [18].

In a research paper, it is clearly stated that the complexity resources are time and space with these two things the complexity of sorting algorithms was measured by the researcher [19]. The authors discussed on the topic of the sorting algorithm. The researcher had highlighted that sorting is an important technique for data structure, it is used in many real-life functions. There are many sorting algorithms. In this paper, the researcher has tested all the sorting techniques. The researcher found some interesting results on the existing techniques of sorting on all the different cases. The researcher has attained results through the experimental data. The researcher had told us about all the sorting techniques, they had shown the algorithm of all the sorting techniques after that he had shown the number of passes of all the sorting techniques. The researcher had also shown the algorithm of all the different sorting techniques. Author had done all this work of sorting in the C++ programming language. Author had shown the time complexity of all the sorting techniques in different tables, after that, the researcher had made the graph of all the sorting techniques in which the comparison of all the techniques is shown, the graph clearly shows which sorting technique is very used fully in C++ programming language. One table shows the complex formulas of worst and best cases of all the sorting techniques. The researcher has taken data randomly, somewhere it is Best case, where the data gave is already sorted while somewhere it is the worst case where the whole data is in reverse order while somewhere he had taken average case data where the data is completely not in the correct order or maybe the data need only one swap to convert its array in sorted order. The researchers had taken data of the length of 10,000, 20,000, and 30,000 for all the sorting techniques [20].

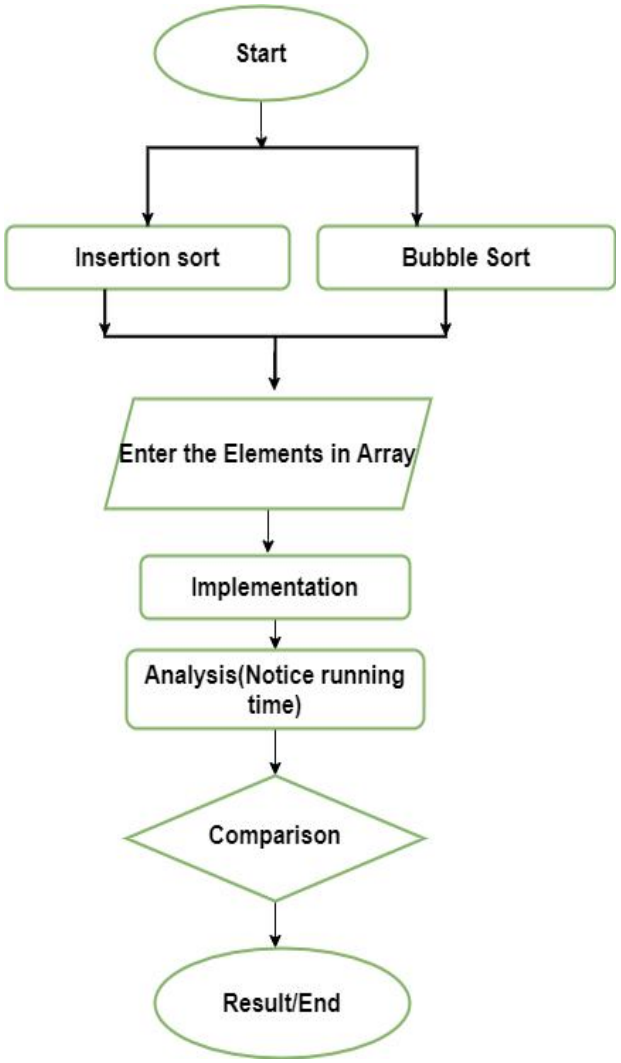
In another research, a paper researcher has shown the time complexity of improved algorithm Bubble sort in C programming language for that researcher has used data length of 1,000 to 10,000 and the software used by the researcher for running C language was turbo C++. The researcher has shown a graph in which he has compared the result for the bubble sort algorithm and its improved version in which algorithm works more efficiently than the traditional algorithm of Bubble sort [2].

## 3. METHODOLOGY

The performance of sorting techniques, that is insertion sort and bubble sort is shown in the paper. For checking the performance of insertion sort and bubble sort the factor of time complexity is used. In this paper, performance is checked by the time required by the data length of 5, 10, and 15 inputs to sort the unsorted array. Performance analysis is done on three different cases, the best case: where the data gave is already sorted, the average case: where the data is not completely sorted nor the data is completely in reverse order and the worst case: where the data is completely in reverse order. All the cases are applied on 5, 10, and 15number of inputs, for the analysis rust programming language and Python language are used to check insertion sort and bubble sort technique. For compiling rust language, microsoft visual studio code was used. For compiling python code, anaconda software was used.

The given below flow chart, in which the flow of data is shown for insertion sort and bubble sort. The algorithm helps non-programmers to understand the sorting behavior of bubble sort and insertion sort. The time required by bubble

sort and insertion sort are shown in tables, one table shows the time required by rust programming language and the other table shows the time required by python language. After that, the comparison of bubble sort and insertion sort are shown in the table where one section represents the time required by bubble sort and the other section of the table represents the time required by insertion sort, this comparison is done on python programming language and rust programming language. Comparison is represented in a graphical format. The result is shown which tells that which technique works best for bubble sort and which technique is best for insertion sort.




Flowchart

## 4. IMPLEMENTATION

Bubble sort and insertion sort are implemented in best case: where the only comparison is needed but it does not require sorting because the data is already sorted, average case: where comparison and swap both are needed to sort the unsorted data and worst case: where comparison and swap both are needed because the data list is completely in reverse order. The array is checked that either the given array needs swap or not, if the swap is required, numbers would be swap else the index would be increased and this process should be done repeatedly until the whole list is sorted.

### 4.1 Algorithm of Bubble Sort

The algorithm of bubble sort is shown below

- Given a list of n elements with values or records A0, A1… An-1, bubble sort is applied to sort an unsorted list of arrays.
- Compare the first two elements of array A0, A1 in the list.
- If A1 < A0, swap those elements and continue with the next 2 elements of the list.
- Repeat the same process until whole the list of the array is sorted.
- Return the final sorted list of arrays.

### 4.2 Algorithm of Insertion Sort

The algorithm of insertion sort is as followed

- To arrange perfectly any bunch of numbers whose contents are of any size in a climb up the order.
- Repeat from array to array[n] on the given array.
- Analyze the ongoing array ingredient with its previous ingredient.
- If the previous ingredient is lower then, move the greater elements one position up.
- Repeat the same process until the whole list is sorted.

### 4.3 Results Analysis

The python language code of bubble sort is compiled to analyze three different inputs; the data length used was5, 10, and 15 numbers of inputs. The 5 number of inputs were used for analyzing the best case, 10 numbers of data lengths were used for analyzing the average case, and 15 number of input was used for analyzing worst-case. The time required to complete the run of bubble sort got different readings these readings are demonstrated below in the table 1.

**Table 1:** Bubble sort in Python

| Bubble sort | Time required | No. of inputs |
|---|---|---|
| Best cases | 1.44 seconds | 5 |
| Average case | 1.56 seconds | 10 |
| Worst case | 1.56 seconds | 15 |

The code of insertion sort is compiled to analyze 5, 10, and 15 inputs for the best case, the average case, and the worst case. The time complexity is noted, the process was done in python language. The readings obtained are presented in the below given table 2.

**Table 2:** Insertion sort in Python

| Insertion sort | Time required | No. of inputs |
|---|---|---|
| Best cases | 0.99 seconds | 5 |
| Average case | 1 second | 10 |
| Worst case | 1.01 seconds | 15 |

The time required by bubble sort to complete its sorting in rust language is noted for best case, the average case, and the worst case. The below table 3, shows clearly the time complexity.

**Table 3:** Bubble sort in Rust

| Bubble sort | Time required | No. of inputs |
|---|---|---|
| Best cases | 1.39 seconds | 5 |
| Average case | 0.79 seconds | 10 |
| Worst case | 0.78 seconds | 15 |

The time required by insertion sort to complete sorting was noted where the number of inputs was 5, 10, and 15 for best case, the average case, and the worst case. The time required by rust is clearly reflected in the table 4.

**Table 4:** Insertion sort in Rust

| Insertion sort | Time required | No. of inputs |
|---|---|---|
| Best cases | 0.82 seconds | 5 |
| Average case | 0.89 seconds | 10 |
| Worst case | 0.90 seconds | 15 |

**Comparison:**

The comparison of bubble sort was done by the time required by rust language and python language. The below table 5, show the time required by rust for bubble sort and the time required by python for bubble sort both are given below. The table below shows which language works better for bubble sort.

**Table 5:** Comparison of Bubble sort

| No. of inputs | Time required by Python | Time required by Rust |
|---|---|---|
| 5 | 1.44 sec | 1.39 sec |
| 10 | 1.56 sec | 0.79 sec |
| 15 | 1.57 sec | 0.78 sec |

The table 6, demonstrates the time required by rust language and python language to complete sorting in insertion sort, for 5, 10, and 15 numbers of inputs over best, average and worst case.

**Table 6:** Comparison of Insertion sort

| No. of inputs | Time required by Python | Time required by Rust |
|---|---|---|
| 5 | 0.99sec | 0.82 sec |
| 10 | 1 sec | 0.89 sec |
| 15 | 1.01 sec | 0.90 sec |

## 5. RESULTS DISCUSSION AND COMPARISON

By the experiment over bubble sort and insertion sort on different data lengths in rust language and python language. It is noticed that insertion sort works better in rust language for the data length of 5, 10, and 15 numbers. Insertion sort works efficiently in rust language as it requires the least amount of time to complete sorting. The time required by bubble sort in rust and python language is represented graphically; the graph clearly shows that rust language requires less amount of time to complete sorting. Insertion sort works better for data length of 5, 10, and 15 numbers. The graph of insertion sort is also shown below, which shows the time required by rust and python language. By the analysis, it is found that Rust language is efficient for insertion sort as it requires the least amount of time to complete sorting. While comparing Rust and python language, it is analyzed that rust language requires the least amount of time. Comparing bubble and insertion sort it is found that bubble sort required the least amount of time to complete sorting for the given data length. Rust is a good language for both bubble and insertion sort techniques.

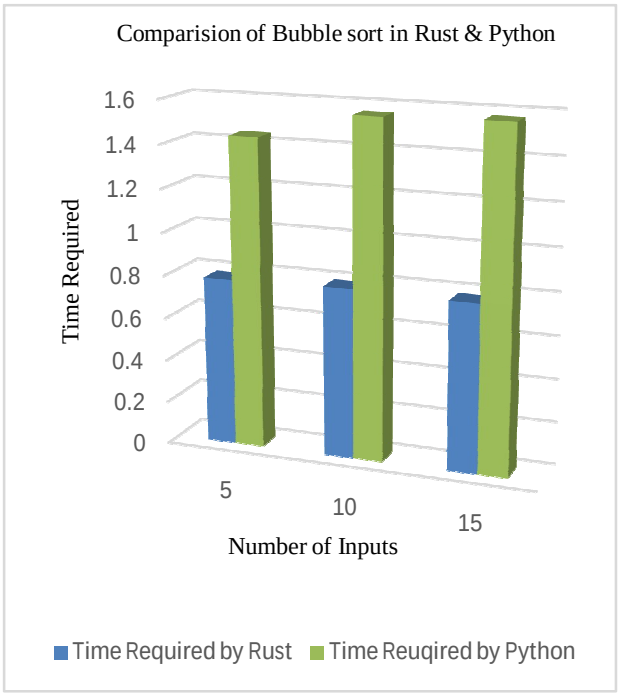



**Figure 1:** Comparison of Bubble sort

The above figure 1: shows the efficiency of bubble sort in both languages. The grey bar shows the time required by

python language and the blue bar shows the time required by rust language. By whole data, it is analyzed that rust language is more efficient for Bubble sort and works efficiently.

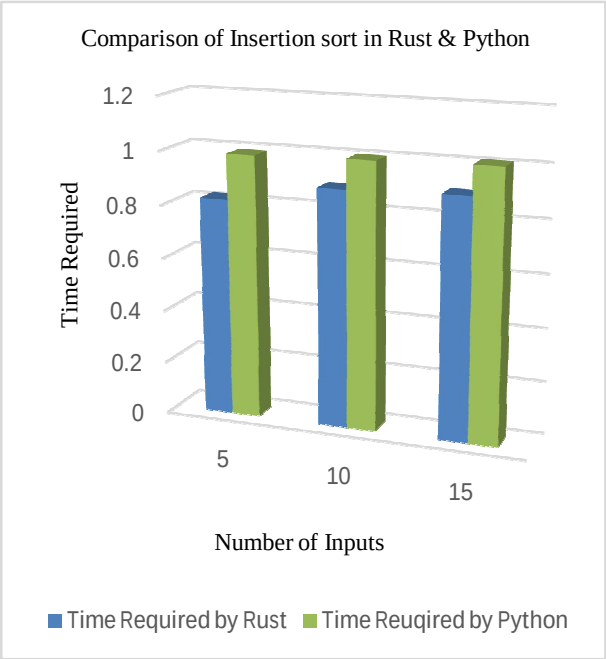



**Figure 2:** Comparison of Insertion sort

The above figure 2: show the efficiency of insertion sort in rust and python language. The grey bar shows the time required by python language and the blue bar shows the time required by rust language. By whole data, it is analyzed that rust language is more efficient for insertion sort as it requires less amount of time to complete sorting.

## 6. CONCLUSION

In this study, the authors have attempted to analyze the performance of bubble sort and insertion sort algorithms concerning time complexity. Firstly, the bubble sort and insertion sort algorithm was implemented in python language and observed the time complexity. Secondly, the bubble sort and insertion sort algorithm implemented in rust language and time complexity observed. The processing time of algorithms is observed in platforms, python, and rust. However, the results show that the rust language is more efficient in performance than python concerning time complexity for bubble sort and insertion sort algorithms. Besides this, it is observed that insertion sort is more efficient than bubble sort.

## REFERENCES

[1] M. Shahzad, M. Shakeel, A. U. Rehman, and M. U. Shoukat. **Review on Sorting Algorithms - A Comparative Study**, *International Journal of Innovative Science and Modern Engineering*, vol. 5, no. 1, p. 4, 2017.

[2] Ali, I., H, Nawaz, I.K., Ameen, M., Chhajro, M. and Maitlo, A. **Performance Comparison between Merge and Quick Sort Algorithms in Data Structure**, in *International Journal Of Advanced Computer Science And Applications*, 9(11), pp.192-195, 2018.

[3] P. Kumar, A. Gangal, S. Kumari, and S. Tiwari. **Recombinant Sort: N-Dimensional Cartesian Spaced Algorithm Designed from Synergetic Combination of Hashing, Bucket, Counting and Radix Sort**, in *Ingénierie Systèmes Inf.*, vol. 25, Dec. 2020. doi: 10.18280/isi.250513.

[4] J. P. Ocampo. **An empirical comparison of the runtime of five sorting algorithms**, p. 26.

[5] A. Alotaibi, A. Almutairi, and H. Kurdi. **OneByOne (OBO): A Fast Sorting Algorithm**, in *Procedia Comput. Sci.*, vol. 175, pp. 270–277, 2020. doi: 10.1016/j.procs.2020.07.040.

[6] B. Bramas. **Fast Sorting Algorithms using AVX-512 on Intel Knights Landing**, *arXiv preprint arXiv:1704.08579*, pp. 1-17, 2017.

[7] J. Alnihoud and R. Mansi. **An Enhancement of Major Sorting Algorithms**, *Int. Arab J. Inf. Technol.* vol. 7, no. 1, pp 55-62, 2010.

[8] E. Insanudin. **Implementation of python source code comparison results with Java using bubble sort method**, in *J. Phys. Conf. Ser.*, vol. 1280, p. 032027, Nov. 2019, doi: 10.1088/1742-6596/1280/3/032027.

[9] J. Tait, T. Ripke, L. Roger, and T. Matsuo. **Comparing Python and C++ Efficiency Through Sorting**, in *International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, NV, USA, Dec. 2018, pp. 864–871. doi: 10.1109/CSCI46756.2018.00172.

[10] W. Min. **Analysis on Bubble Sort Algorithm Optimization**, in *2010 International Forum on Information Technology and Applications*, Kunming, China, Jul. 2010, pp. 208–211. doi: 10.1109/IFITA.2010.9.

[11] S. Zafar, Hina, and A. Wahab, A new friends sort algorithm, in *2nd IEEE International Conference on Computer Science and Information Technology*, Aug. 2009, pp. 326–329. doi: 10.1109/ICCSIT.2009.5234550.

.[12]F. G. Furat. **A Comparative Study of Selection Sort and Insertion Sort Algorithms**, in *International Research Journal of Engineering and Technology (IRJET)*, vol. 03, no. 12, pp. 336–330, Dec. 2016.

[13] D. Rajagopal and K. Thilakavalli. **Different Sorting Algorithm's Comparison based Upon the Time Complexity**, in *Int. J. U- E- Serv. Sci. Technol.*, vol. 9, no. 8, pp. 287–296, Aug. 2016, doi: 10.14257/ijunesst.2016.9.8.24.

[14] N. Yadav and S. Kumari. **Sorting Algorithms,** in *International Research Journal of Engineering and Technology*, vol. 3, no. 2, pp. 528-531, 2016.

[15] You Yang, Ping Yu, and Yan Gan. **Experimental study on the five sort algorithms**, in *Second International Conference on Mechanic Automation and Control Engineering*, Inner Mongolia, China, Jul. 2011, pp. 1314–1317. doi: 10.1109/MACE.2011.5987184.

[16] R. dwivedi Dr. Dinesh C. Jain. **A Comparative Study on Different Types of Sorting Algorithms (On the Basis of C and Java)**, *Eng. Technol.*, vol. 5, no. 08, pp. 805–808, 2014.

[17] M. A. Bender, M. Farach-Colton, and M. Mosteiro. **Insertion Sort is O(n log n),** *arXiv:cs/0407003*, Jul. 2004, Accessed: Mar. 13, 2021. [Online]. Available: http://arxiv.org/abs/cs/0407003.

[18] N. Akhter and M. Idrees. **Sorting Algorithms – A Comparative Study**, vol. 14, no. 12, p. 8, 2016.

[19] M. Agenis-Nevers, N. D. Bokde, Z. M. Yaseen, and M. Shende. **An empirical estimation for time and memory algorithm complexities: newly developed R package**, *Multimed. Tools Appl.*, vol. 80, no. 2, pp. 2997–3015, Jan. 2021.
doi: 10.1007/s11042-020-09471-8.

[20] K. S. Al-Kharabsheh, I. M. AlTurani, A. M. I. AlTurani, and N. I. Zanoon. **Review on Sorting Algorithms A Comparative Study**, *International Journal of Computer Science and Security (IJCSS), 7*(3), pp.120-126, 2013.