Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Issue 4, October 2025
DOI : https://doi.org/10.33395/sinkron.v9i4.15005
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Research and Analysis of Exchange Sort Algorithm in Data Structure

Rakhmat Purnomo[1], Tri Dharma Putra[2*]
[1,2]Department of Informatics, Faculty of Computer Science, Universitas Bhayangkara Jakarta Raya
rakhmat.purnomo@dsn.ubharajaya.ac.id, tri.dharma.putra@dsn.ubharajaya.ac.id*

Submitted : Jul 7, 2025 | **Accepted :** Aug 10, 2025 | **Publish :** Oct 2, 2025

Abstract: Exchange sort is different from bubble sort. Exchange sort compares an element with other elements in the array, and swaps elements if necessary. So there is an element that is always the center element (pivot). Here is its theoretical description: Comparison, the algorithm compares each element with its adjacent element. Then continue until all elements are compared. Swap: If the elements are in the wrong order (for example, in ascending order, if the left element is greater than the right), they are swapped. This swapping continues until all match numbers are swapped. Iteration, this process of comparing and swapping, is repeated for each pair of adjacent elements in the array. Looping, this process is repeated a number of times (traversing) the array until no more swapping is required, indicating that the array is sorted. It is concluded that for the six numbers in these three case studies, the iterations needed are 5 iterations each. The swaps counts needed are 7, for case study 1. The swap counts needed are 12 for case study 2 and the swap counts are 8 for case study 3. In this research and analysis, the order, all of them is descending, although it can be made ascending. In modern days, exchange sort plays a very important role in terms of sorting algorithms. This paper is only research and analysis. For novelty, the analysis is given with a clear step-by-step procedure of the algorithm.

Keywords: ascending, descending, exchange sort, pivot, swap

INTRODUCTION

The two most common problems in data processing and computer science in general are sorting and merging algorithms (Rabiu et al., 2022) (Chauhan & Duggal, 2020) (Alotaibi et al., 2020). Sorting is a vital data structure activity that facilitates finding, organizing, and locating information (Cherukuri Nischay Sai, 2021). Sorting algorithm plays a crucial role in information systems technology. A sorting algorithm is a step-by-step procedure in arranging items on a list in a particular order. Whether it is ascending or descending (Vilchez, 2020) (Zutshi & Goswami, 2021). Many things need to be sorted. Several sorting algorithms are available in the field. Bubble sort, heap sort, merge sort, quick sort, exchange sort, selection sort, etcetera.

One main issue in sorting algorithms is speed (Ekowati et al., 2022) (Mehdi Rizvi et al., 2024). Sorting is one of the urgent tasks in managing and processing the data, especially when the amount of data is very large (Gill et al., 2018). The faster the speed, the better. Optimization of the sorting algorithm to increase speed is conducted by many researchers. Comparison between candidate numbers may be reduced by some algorithms. To make the sorting has less time (Hanafi et al., 2022).

Many researchers have studied different algorithms of sorting in early 1950s and still continue till today. The new technology in IoT, Blockchain, booms the techniques both in parallelism and in processors. Hence, data processing, data curation, and data retrieval are also developed (Selvi et al., 2021) (Zutshi & Goswami, 2021) (Naz et al., 2021).

Exchange sort is different with bubble sort. Exchange sort compares an element with other elements in the array, and swaps elements if necessary (Purnomo & Putra, 2023). So there is an element that is always the center element (pivot). While Bubble sort will compare the first/last element with the previous/next element, then that element will become the center (pivot) to be compared with the previous/next element again, and so on (Zhu, 2020) (Zhu, 2020)(Mankowitz et al., 2023).

In this journal the discussion is about exchange sort. Analysis is given with three case studies. Each with different numbers to be sorted. Then discussion about this presented in the next chapter of it, to compare the result of this algorithm in the three case studies. Each random numbers are given with 6 of them. In this three case studies, explanation and analysis about exchange sort are given thoroughly. In modern days exchange sort plays a very important role in terms of sorting algorithm. This paper is only research and analysis, for novelty, the analysis given with clear step by step procedure of the algorithm

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Issue 4, October 2025
DOI : https://doi.org/10.33395/sinkron.v9i4.15005
e-ISSN : 2541-2019
p-ISSN : 2541-044X

This paper is organized as follows. Section 1 is introduction. Section 2 is method. It is about the algorithm of exchange sort. Section 3 is research and analysis with three case studies. Section 4 is discussion about the result, and the last one is conclusion.

METHODS

Exchange sort is a simple comparison-based sorting algorithm. It works by repeatedly comparing adjacent elements and swapping them if they are not in order. This process is repeated until the entire array is sorted. The algorithm gets its name from the repeated swapping of elements . Here is its theoretical description (Singh et al., 2024):

Comparison
The algorithm compares each element with its adjacent element. Then continue until all elements are compared.

Swap
If the elements are in the wrong order (for example, in ascending order, if the left element is greater than the right), they are swapped. This swapping continues until all match numbers are swapped.

Iteration
This process of comparing and swapping is repeated for each pair of adjacent elements in the array. Repetition occurs here.

Looping
This process is repeated a number of times (traversing) the array until no more swapping is required, indicating that the array is sorted. The final result of this looping is the elements which is already sorted out.
The pseudo code of exchange sort algorithm:

```
n = length(A)
 //outer loop
 for i = 1 to n - 2 do
 //inner loop.
    for j = i + 1 to n-1 do
        if num[i] < num[j] do
          swap(num[i], num[j])
        end if
     end for
  end for
end procedure
```

Flowchart
This flowchart illustrates the conditional constructs, loops, and other elements of control flow that comprise an algorithm for sorting, from smallest to largest, an arbitrary list of numbers. In this type of diagram, arrows symbolize the flow of logic (control flow), rounded rectangles mark the start and end points, slanted parallelograms indicate I/O (e.g., a user-provided list), rectangles indicate specific subroutines or procedures (blocks of statements), and diamonds denote conditional constructs (branch points). Note that this sorting algorithm involves a pair of nested loops over the list size (blue and orange), meaning that the calculation cost will go as the square of the input size (here, an N-element list); this cost can be halved by adjusting the inner loop conditional to be “”, as the largest i elements will have already reached their final positions.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Issue 4, October 2025
DOI : https://doi.org/10.33395/sinkron.v9i4.15005
e-ISSN : 2541-2019
p-ISSN : 2541-044X

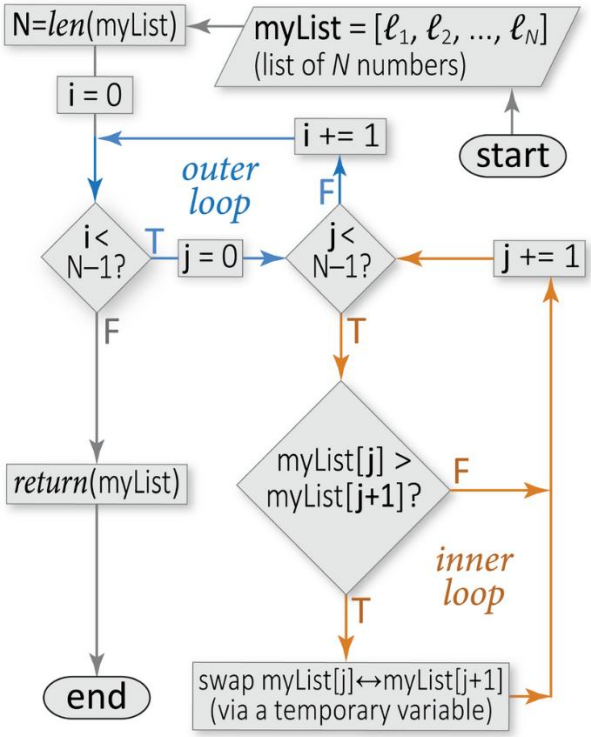

Figure 1. Flowchart of Exchange Sort (Ekmekci et al., 2016)

RESULTS

In this research and analysis, three case studies are given. Each with six numbers to be sorted. All of them is in descending order, from the biggest number to the smallest one. The numbers in the array that will be compared and matched, are given in blue. The read ones are the numbers that swapped. In this research, the step by step analysis are given. If compared with other exchange sort algorithm like bubble sort, there is no direct difference between the two. Since this research and analysis focus on exchange sort.

Case Study 1:
We have numbers below, which is not sorted yet. The pivot is in the first column, with number 90 in it. Then, this 90 is compared with other numbers from 29 , 30, and 45. Since this numbers is smaller then 90, there is no swapping. The swapping are done in the array[4], which is 99. Since 99 is bigger then 90, then swapped to the pivot. Please take a look at tabel 1. Below.

Tabel 1. 1st iteration

pivot					
Array[0]	Array[1]	Array[2]	Array[3]	Array[4]	Array[5]
90	29	30	45	**99**	6
99	29	30	45	90	6

For 2nd iteration, the pivot is now in array[1]. The array[0] are done, since it is the biggest number already. Then comparison are done at this pivot with the next numbers. First swap is for 29 with 30. Then for 30 and 45, then 90 and 45. They are all arranged in descending order. Please take a look at tabel 2., below.

Tabel 2. 2nd iteration

	pivot				
Array[0]	Array[1]	Array[2]	Array[3]	Array[4]	Array[5]
99	**29**	30	45	90	6
99	**30**	29	45	90	6
99	**45**	29	30	90	6
99	**90**	29	30	45	6

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Issue 4, October 2025
DOI : https://doi.org/10.33395/sinkron.v9i4.15005
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Please take a look at tabel 3., below. The pivot is now in array[2] and two swaps are done. After the comparisons of numbers sorted, the first swap is 30 and 29, the second swap is 30 and 45.

Tabel 3. 3rd iteration

		pivot			
Array[0]	Array[1]	Array[2]	Array[3]	Array[4]	Array[5]
99	90	**29**	30	45	6
99	90	**30**	29	45	6
99	90	**45**	29	30	6

In the tabel 4., below, the pivot is in array[3]. The swap is only one, which is between 30 and 29.

Tabel 4. 4th iteration

			pivot		
Array[0]	Array[1]	Array[2]	Array[3]	Array[4]	Array[5]
99	90	45	**29**	30	6
99	90	45	**30**	29	6

For the last iterations, there is no swap of numbers to be sorted in this iteration. The pivot is in arry[4], but when compared between 29 and 6, it is already in descending order, there is no need to be sorted, since it is already in descending order. In this tabel 5., it is concluded that the sorting is finished, with 99 as the biggest number in the left, and 6 as the smallest number in the right, in array[5].

Tabel 5. 5th iteration

				pivot	
Array[0]	Array[1]	Array[2]	Array[3]	Array[4]	Array[5]
99	90	45	30	**29**	6

Case Study 2:

We have numbers below, which is not sorted yet. Please take a look at array[0]. The pivot is in the first column, with number 20 in it. Then, this 20 is swapped with other numbers from 21, 44, 99 and 93. Let's take a look at array[1]. Since this numbers is bigger than 20, it is swapped. The swapping are done in the array[1], which is 21. In array[2], since 44 is bigger than 21, then it is swapped to the pivot. And the last time 44 in the pivot is swapped with 99 in the array[4]. This is the first iteration. Please take a look at tabel 6., below.

Tabel 6. 1st iteration

pivot					
Array[0]	Array[1]	Array[2]	Array[3]	Array[4]	Array[5]
20	21	44	8	99	93
21	20	44	8	99	93
44	20	21	8	99	93
44	20	21	8	99	93
99	20	21	8	44	93

Please take a look at tabel 7., below. The pivot now is the second column. There are three swapped. First, number 21 and 20. Then 44 and 21 and then 44 swapped with 93.

Tabel 7. 2nd iteration

	pivot				
Array[0]	Array[1]	Array[2]	Array[3]	Array[4]	Array[5]
99	**20**	21	8	44	93
99	**21**	20	8	44	93
99	**44**	20	8	21	93
99	**93**	20	8	21	44

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Issue 4, October 2025
DOI : https://doi.org/10.33395/sinkron.v9i4.15005
e-ISSN : 2541-2019
p-ISSN : 2541-044X

For the 3rd iteration, please take a look at tabel 8., below. The pivot now is the third column, it is array[2]. There are two swapps here. First 21 and 20, and then 21 with 44.

Tabel 8. 3rd iteration

		pivot			
Array[0]	Array[1]	Array[2]	Array[3]	Array[4]	Array[5]
99	93	**20**	8	21	44
99	93	**21**	8	20	44
99	93	**44**	8	20	21

For the 4th iteration, please take a look at tabel 9., below. Two swapps are done. First 20 and 8 and the second one is 20 and 21. Since 21 is bigger than 20, it is swapped and 20 is bigger than 8.

Tabel 9. 4th iteration

			pivot		
Array[0]	Array[1]	Array[2]	Array[3]	Array[4]	Array[5]
99	93	44	**8**	20	21
99	93	44	**20**	8	21
99	93	44	**21**	8	20

For the last iteration, the 5th iteration, please take a look at tabel 10., below. Array[4] contains 8 and array[5] with 20 in it. Since 20 is bigger than 8, the it is swapped. This means in this iteration, there is only one swapp.

Tabel 10. 5th iteration

				pivot	
Array[0]	Array[1]	Array[2]	Array[3]	Array[4]	Array[5]
99	93	44	21	**8**	20
99	93	44	21	**20**	8

Case Study 3:

We have numbers below, which is not sorted yet. Take a look at tabel 11., below. Please take a look at array[0]. The pivot is in the first column, with number 33 in it. Then, this 33 is swapped with the rest of the numbers, since this 33, is the smallest. Let's take a look at array[1]. Since this number, 63 is bigger than 33, it is swapped. The swapping are done in the array[1], which is 63. In array[2], since 57 is smaller than 63, then it is not swapped to the pivot. But for array[3], the contain is 88, Since this 88 is bigger than 63, it is swapped. This is the first iteration.

Tabel 11. 1st iteration

pivot					
Array[0]	Array[1]	Array[2]	Array[3]	Array[4]	Array[5]
33	63	57	88	49	50
63	33	57	88	49	50
88	33	57	63	49	50

Please take a look at tabel 12. This is the 2nd iteration. The pivot is now in array[1]. There are two swaps here, which are between array[1] and array[2], 33 and 57. Then between 57 and 63, since 63 is bigger than 57.

Tabel 12. 2nd iteration

	pivot				
Array[0]	Array[1]	Array[2]	Array[3]	Array[4]	Array[5]
88	**33**	57	63	49	50
88	**57**	33	63	49	50
88	**63**	33	57	49	50

Please take a look at tabel 13., below. This is the 3rd iteration. The pivot is now in the 4th column, array[3]. Only two swaps occurred here. It is between and 33 and 49, then between 49 and 50.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Issue 4, October 2025
DOI : https://doi.org/10.33395/sinkron.v9i4.15005
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Tabel 13. 3rd iteration

		pivot			
Array[0]	Array[1]	Array[2]	Array[3]	Array[4]	Array[5]
88	63	**33**	57	49	50
88	63	**57**	33	49	50

Please take a look at tabel 14., below. This is the 4th iteration. The pivot now is in array[3]. Number 33 is compared with 49, then it is swapped. After that, 49 is swapped with 50.

Tabel 14. 4th iteration

			pivot		
Array[0]	Array[1]	Array[2]	Array[3]	Array[4]	Array[5]
88	63	57	**33**	49	50
88	63	57	**49**	33	50
88	63	57	**50**	33	49

Please take a look at the 5th iteration, tabel 15., below. The pivot is not in the 5th column. It is array[4]. Only once swap occurred. We swap 33 and 49. Then this is the last sorted numbers, because right now it is already in descending order, all of them.

Tabel 15. 5th iteration

				pivot	
Array[0]	Array[1]	Array[2]	Array[3]	Array[4]	Array[5]
88	63	57	50	**33**	49
88	63	57	50	**49**	33

DISCUSSION

Based on results we get from the research and analysis above, we have this tabel 16., to be discussed. Each case study has 5 iterations, since the numbers are only six. But the total swaps are different. For case study 1, the swaps are 7. For second case study the swaps are 12 and for the last case study, case study 3, the swaps are 8. The second case study has the biggest swaps, namely 12. The total iterations for each case study is 5.

Tabel 16. Discussion of the results

Case Study	Iterations	Total swapped
1	5	7
2	5	12
3	5	8

CONCLUSION

It is concluded that for the six numbers in these three case studies, the iterations needed are 5 iterations each. The biggest swaps needed are 12, for case study 2. The smallest swap needed is 7 for case study 1, and the swap counts are 8 for case study 3. In this research and analysis, the order, all of them, is descending. To make it ascending, it is just with reversing the analysis and comparisons. This paper is only research and analysis, for novelty, the analysis is given with a clear step-by-step procedure of the algorithm. A flowchart is also given. For future works, it is better to compare the exchange sort algorithm with other known algorithms in sorting numbers. Major known algorithms in terms of sorting are available and the comparison in terms of the time needed to execute these algorithms are crucial. It is also suggested that implementation with major programming language can be given, like C++ or other programming languages. So that the speed of each case study can be discussed.

REFERENCES

Alotaibi, A., Almutairi, A., & Kurdi, H. (2020). OneByone (OBO): A fast sorting algorithm. *Procedia Computer Science*, *175*, 270–277. https://doi.org/10.1016/j.procs.2020.07.040

Chauhan, Y., & Duggal, A. (2020). Different Sorting Algorithms comparison based upon the Time Complexity. *International Journal of Research and Analytical Reviews*, *7*(3), 114–121. www.ijrar.org

Cherukuri Nischay Sai. (2021). A Comprehensive Study of Various Sorting Algorithms. *International Journal of Science and Research (IJSR*, *10*(11), 1044–1052.

Ekmekci, B., McAnany, C. E., & Mura, C. (2016). An Introduction to Programming for Bioscientists: A Python-Based Primer. *PLoS Computational Biology*, *12*(6). https://doi.org/10.1371/journal.pcbi.1004867

*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

Ekowati, M. A. S., Nindyatama, Z. P., Widianto, W., & Dananti, K. (2022). Comparative Analysis of the Speed of the Sorting Method on Google Translate Indonesian-English Using Binary Search. *International Journal of Global Operations Research*, *3*(3), 108–115. https://doi.org/10.47194/ijgor.v3i3.167

Gill, S. K., Singh, V. P., Sharma, P., & Kumar, D. (2018). A Comparative Study of Various Sorting Algorithms. *International Journal of Advanced Research in Computer Science*, *4*, 366–372.

Hanafi, M. R., Faadhilah, M. A., Dwi Putra, M. T., & Pradeka, D. (2022). Comparison Analysis of Bubble Sort Algorithm with Tim Sort Algorithm Sorting Against the Amount of Data. *Journal of Computer Engineering, Electronics and Information Technology*, *1*(1), 29–38. https://doi.org/10.17509/coelite.v1i1.43794

Mankowitz, D. J., Michi, A., Zhernov, A., Gelmi, M., Selvi, M., Paduraru, C., Leurent, E., Iqbal, S., Lespiau, J. B., Ahern, A., Köppe, T., Millikin, K., Gaffney, S., Elster, S., Broshear, J., Gamble, C., Milan, K., Tung, R., Hwang, M., … Silver, D. (2023). Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, *618*(7964), 257–263. https://doi.org/10.1038/s41586-023-06004-9

Mehdi Rizvi, Q., Rai, H., & Jaiswal, R. (2024). Sorting Algorithms in Focus: A Critical Examination of Sorting Algorithm Performance. *Emerging Trends in IoT and Computing Technologies*, *March*, 103–106. https://doi.org/10.1201/9781003535423-19

Naz, A., Nawaz, H., Maitlo, A., & Hassan, S. M. (2021). Implementation of Selection Sort Algorithm in Various Programming Languages. *International Journal of Advanced Trends in Computer Science and Engineering*, *10*(3), 2371–2377. https://doi.org/10.30534/ijatcse/2021/1231032021

Purnomo, R., & Putra, T. D. (2023). Theoretical Analysis of Standard Selection Sort Algorithm. *SinkrOn*, *8*(2), 666–673. https://doi.org/10.33395/sinkron.v8i2.12153

Rabiu, A. M., Garba, E. J., Baha, B. Y., & Mukhtar, M. I. (2022). Comparative Analysis between Selection Sort and Merge Sort Algorithms. *Nigerian Journal of Basic and Applied Sciences*, *29*(1), 43–48. https://doi.org/10.4314/njbas.v29i1.5

Selvi, S., Evert, M. A. C., & Case, B. (2021). *Online Copy Available : www.ijmer.in IMPLEMENTATION OF EFFICIENT SORTING ALGORITHM IN C / C ++*. *514*(3), 34–40.

Singh, Y., Verma, M., Pandey, I., Saini, A., Chawla, P., & Niranjan, V. (2024). Analysis and Comparison of Various Sorting Algorithms. *Tuijin Jishu/Journal of Propulsion Technology*, *45*(2), 5885–5890.

Vilchez, R. N. (2020). Modified Selection Sort Algorithm Employing Boolean and Distinct Function in a Bidirectional Enhanced Selection Technique. *International Journal of Machine Learning and Computing*, *10*(1), 93–98. https://doi.org/10.18178/ijmlc.2020.10.1.904

Zhu, Z. G. (2020). Analysis and Research of Sorting Algorithm in Data Structure Based on C Language. *Journal of Physics: Conference Series*, *1544*(1). https://doi.org/10.1088/1742-6596/1544/1/012002

Zutshi, A., & Goswami, D. (2021). Systematic review and exploration of new avenues for sorting algorithm. *International Journal of Information Management Data Insights*, *1*(2), 100042. https://doi.org/10.1016/j.jjimei.2021.100042

*name of corresponding author