Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 2, April 2023
DOI : https://doi.org/10.33395/sinkron.v8i2.12153
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Theoretical Analysis of Standard Selection Sort Algorithm

Rakhmat Purnomo[1], Tri Dharma Putra[2*]
Universitas Bhayangkara Jakarta Raya
[1)]rakhmat.purnomo@dsn.ubharajaya.ac.id, [2)]tri.dharma.putra@dsn.ubharajaya.ac.id

Submitted : Jan 23, 2023 | **Accepted** : Feb 22, 2023 | **Published** : Apr 1, 2023

Abstract: Sorting algorithms plays an important role in the computer science field. Many applications use sorting algorithm. There are several sorting algorithms proposed by experts, namely bubble sort, exchange short, insertion short, heap sort, quick short, merge sort, standard selection sort. One well-known algorithm of sorting is selection sort. In this journal, discussion about standard selection sort is given with thorough analysis. Sorting is very important data structure concepts that has an important role in memory management, file management, in computer science in general, and in many real-life applications. Different sorting algorithms have differences in terms of time complexity, memory use, efficiency, and other factors. There are many sorting algorithms exist right now in the computer science field. Each algorithm has its benefits and limitations where a trade-off exists between execution time and the nature of the complexity of the algorithm itself. The method is theoretical analysis. Three theoretical analyses are given with deep explanation and analysis. Each with six index arrays, namely with six data on it. The numbers are sorted in ascending order. Pseudo code is also given, to understand this algorithm more thoroughly. It is concluded that this theoretical analysis explained the algorithm more clearly, by using process iteration by hand.

Keywords: Algorithm, Ascending, Process Iteration, Selection Sort, Theoretical Analysis

INTRODUCTION

Sorting plays one crucial role in the computer science field. The sorting algorithms are used in a number of applications. Research on sorting algorithm had occurred since 1950 and continues till today (Selvi et al., 2021), (Ekowati et al., 2022), (Fagbola & Thakur, 2019). Sorting is one common problem in data processing. Sorting means, arranging data in ascending or descending order, from random order of data.

Sorting is very important in the concept of computer science that plays a very important role in memory management, file management, and other real-life applications (Vilchez, 2020). Different sorting algorithms have differences in terms of time complexity, memory use, efficiency, and other factors (Zutshi & Goswami, 2021). There are many sorting algorithms exist right now in the computer science field. There are bubble sort, heap sort, quick sort, insertion sort, exchange sort, merge sort, selection sort algorithms in the computer science field. In each algorithm, there are limitations and benefits, in which a trade-off exists in terms of time execution and the way of the complexity of the algorithm itself.

There are many sorting algorithms that have been developed to solve the problems of random data. Increasing the speed of sorting is also important to improve the efficiency and effectivity of sorting

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 2, April 2023 e-ISSN : 2541-2019
DOI : https://doi.org/10.33395/sinkron.v8i2.12153 p-ISSN : 2541-044X

algorithm. To lower the running time and and increase the speed to make it efficient are the main objective from the experts. The performance of the sorting algorithm is important, since at this moment, the issue of big data emerged. Data with thousands of records must be sorted, if the speed and efficiency are not enough then, the creation of new algorithm is needed to solve this problem.

Then, the standard selection algorithm is one major algorithm to be introduced. This algorithm is a well-known and an easy sorting algorithm to be understood. This algorithm when dealing with small amounts of data are very useful. But, a lot of data will slow its processing's time and increase its iterations. Standard selection sort, actually, has one advantage if we compare it with other sort algorithm techniques. Eventhough standard selection sort algorithm does a lot of comparisons, this algorithm does a very few amount of data moving (Akshay Zade, Vinod Mandloi, 2020). Thus, if the data has small keys but has large data area, then standard selection sorting may be the choice to be the quickest one.

This standard selection sort algorithm is a theoretical analysis. This theoretical analysis gives an easy example to understand the selection sort algorithm thoroughly. By doing it one by one by hand, each step at a time. We follow the procedure the algorithm does. This gives an in depth understanding of this standard selection sort algorithm.

LITERATURE REVIEW

Selection sort is a one known algorithm. Experts have done many researches on selection sort. Naz Arisha and friends did research in selection sort. He and his team presented the implementation of selection sort using several programming language, which are Rush, Python and C/C++, and finally, measured the time complexity (Naz et al., 2021). Katon and his team created Garuda League support system for esport, with the selection sort method that can be used to facilitate the process of sorting the position fo the competing team from the number of points obtained in several tables on league official website (Priambodo & Sasongko Wibowo, 2021). Paingan and his team, proposed an algorithm which is called selection sort hybrid that is expected to have better performance than the normal selection sort algorithm. Selection sort hybrid algorithm combines the maximum and minimum searching techniques. This hybrid algorithm finds maximum and minimum values in the same time to sort from both side of data (Hardika et al., 2020). Vilchez in his research, proposed and found a remedy on the identified problems of the selection sort such as unstable sorting and run time complexity by way of modifying this well-known algorithm.

The new created algorithm was tested using various data and he and his team, verify its performance. The result of their research was compared with the other available sorting algorithms. They could validate running time complexity. Their research results showed that the newly created selection sort algorithm by using distinct and boolean function in the technique of a bidirectional enhance selection has a significant improvement in terms of run time complexity compared with other sorting algorithms (Vilchez, 2020). Alotaibi proposed a new in-place sorting algorithm called OneByOne (OBO) sort. When compared to selection sort and bubble sort algorithm, OBO demonstrated faster run time at different array sizes. OBO also showed nearly equal performance when compared to insertion sort algorithm (Alotaibi et al., 2020). Again, Vilchez proposed the modified selection sort algorithm that utilizes a bidirectional enhanced selection sort algorithm technique.

In their research, Vilchez tried to reduce the number of comparisons and iterations that usually causes the delay in terms of sorting. The results showed that the modified algorithm has a significant run time complexity improvement compared with other $O(n^2)$ algorithms (Vilchez, 2020). The sorting algorithm must be effective, less complex, stable, and more efficient. Shabaz proposed SA sorting algorithm. Shabaz proposed a new approach to operate on both sorted and unsorted lists or records that can have better execution time on the sorted list. In their research, Shabaz and team, proposed to sort thousands of records either sorted or unsorted. Here, if the record numbers are small, traditional sorting approaches can be used. However, in large data, the situation becomes complex. Therefore, an optimized sorting approach is required. These SA sorting is an approach that developed to check sorted big numbers as it works better on sorted numbers then quick sort and many others (Shabaz & Kumar, 2019). Nishant proposed a new algorithm named modified selection sort algorithm, which work on the basis of selecting two elements at a time, that means selecting two elements simultaneously (Mishra, 2018)

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 2, April 2023 e-ISSN : 2541-2019
DOI : https://doi.org/10.33395/sinkron.v8i2.12153 p-ISSN : 2541-044X

METHOD

Here, the structure of this journal are comprises of six sections. The first section is introduction. In this introduction, the basic idea and the background concept of sorting algorithm is given. The second section is literature review. Previous works by other experts are explained here. The third section is research method. In this section, the concept of method is presented thoroughly, especially the pseudo code of selection algorithm. The fourth section is result. In this section, three theoretical analyses are presented. With each ascending order of numbers, analyse from random order into ascending order. The next section is discussion. This section explains the analysis in the previous section. The last section is conclusion. References are also given in the last part of this journal.

In this research the main method is theoretical analysis on standard selection sort algorithm. Three analyses are given. Each with the basic idea of ascending the data. Selection sort is a well-known algorithm to sort numbers, characters, et cetera to be in ascending order or descending order.

Standard selection sort algorithm is an algorithm that based on comparison to sort data. This algorithm will check an array of elements and will try to find the smallest element in the array. Then this algorithm exchanges the smallest element with the element in the first position. Then, after finishing it, it tries to select the smallest element from the unsorted section of the array after carried out the iteration each. It then exchanges the selected smallest element with the element in the unsorted part of the array. This process will continue until all elements in the array is sorted fully (Rabiu et al., 2022). Standard selection sort is the most simple algorithm, but this algorithm is inefficient in large data sets (Chauhan & Duggal, 2020), (Vilchez, 2019).

Please below find the pseudo code of standard selection sort algorithm, for ascending order of numbers.

```
Void selectionsort(int arr[], int n)
{
int i, min, temp;
        for(int i=0;i<n-1;i++)
        {
        min=i;
                for(int j=i+1;j<n;j++)
                {
                        if (arr[j]<arr[min])
                                min=j;
                }
        temp=arr[min];
        arr[min]=arr[j];
        arr[j]=temp;
        }
}
```

Sorting algorithm can be differentiated by the following parameters: a)Stability b)Adaptivity c)Time Complexity d)Space Complexity. Stability means after sorting those similar elements retain their relativistic positions. Adaptivity means, if it sorts the sequences that are close to sorted faster than random sequences. Time Complexity means, that the total time required by the program to run to completion. Space complexity means the number of memory cells which an algorithm needs (Chauhan & Duggal, 2020).

RESULT

Below is three theoretical analyses. The basic idea is to make the random order of numbers to be an ascending order. Analyses are given with comparison and indexing of the arrays. Three examples are given, with random numbers chosen randomly. The analyses comprise of six indexes, which means it is with six numbers on it. Swapping comments, which indexes are swap, are also given, which one is swap between indexes.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 2, April 2023 e-ISSN : 2541-2019
DOI : https://doi.org/10.33395/sinkron.v8i2.12153 p-ISSN : 2541-044X

First Theoretical Analysis

Let's say we have array with indexes from 0 until 5. There are six numbers. Let's say these numbers are 6, 45, 34, 20, 100, 38. We want to make this numbers ascending. Please take a look on the 1st process below to 5th process below. There are five process iterations:

1st Process Iteration

Index:	0	1	2	3	4	5
Number:	**6**	45	34	20	100	38

Comparison	Position
6<45	0
6<34	0
6<20	0
6<100	0
6<38	0

Comment: There is no changing position

Index:	0	1	2	3	4	5
Number:		6	45	34	20	100	38

2nd Process Iteration

Index:	0	1	2	3	4	5
Number:	6	**45**	34	20	100	38

Comparison	Position
45>34 (change index)	2
34>20	3
20<100	3
20<38	3

Comment: Swap data on index 1 with index 3

Index:	0	1	2	3	4	5
Number:		6	20	34	45	100	38

3rd Process Iteration

Index:	0	1	2	3	4	5
Number:	6	20	**34**	45	100	38

Comparison	Position
34<45	2
34<100	2
34<38	2

Comment: There is no changing position

Index:	0	1	2	3	4	5
Number:		6	20	34	45	100	38

4th Process Iteration

Index:	0	1	2	3	4	5
Number:	6	20	34	**45**	100	38

Comparison	Position
45<100	3
45>38	5

Comment: Swap data on index 3 with index 5

Index:	0	1	2	3	4	5
Number:		6	20	34	38	100	45

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 2, April 2023 e-ISSN : 2541-2019
DOI : https://doi.org/10.33395/sinkron.v8i2.12153 p-ISSN : 2541-044X

5th Process Iteration

Index:	0	1	2	3	4	5	
Number:	6	20	34	38	**100**	45	
Comparison		Position					
100>45			5				
Comment: Swap data on index 4 with index 5							
Index:	0	1	2	3	4	5	
Number:		6	20	34	38	45	100

So that the end result of the ascending data is 6, 20, 34, 38, 45, and 100.

Second Theoretical Analysis

In this second theoretical analysis, let's say we have array with indexes from 0 until 5. There are six numbers. Let's say these numbers are 89, 40, 33, 56, 99, 39. We want to make this numbers ascending. Please take a look on the 1st process to 5th process below. There are five process iterations:

1st Process Iteration

Index:	0	1	2	3	4	5	
Number:	**89**	40	33	56	99	39	
Comparison		Position					
89>40		1					
40>33		2					
33<56		2					
33<99		2					
33<39		2					
Comment: Swap data on index 0 with index 2							
Index:	0	1	2	3	4	5	
Number:		33	40	89	56	99	39

2nd Process Iteration

Index:	0	1	2	3	4	5	
Number:	33	**40**	89	56	99	39	
Comparison		Position					
40<89		1					
40<56		1					
40<99		1					
40>39		5					
Comment: Swap data on index 1 with index 5							
Index:	0	1	2	3	4	5	
Number:		33	39	89	56	99	40

3rd Process Iteration

Index:	0	1	2	3	4	5	
Number:	33	39	**89**	56	99	40	
Comparison		Position					
89>56		3					
56<99		3					
56>40		5					
Comment: Swap data on index 2 with index 5							
Index:	0	1	2	3	4	5	
Number:		33	39	40	56	99	89

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 2, April 2023
DOI : https://doi.org/10.33395/sinkron.v8i2.12153
e-ISSN : 2541-2019
p-ISSN : 2541-044X

4th Process Iteration

Index:	0	1	2	3	4	5
Number:	33	39	40	**56**	99	89

Comparison	Position
56<99	3
56<89	3

Comment: There is no changing position

Index:	0	1	2	3	4	5	
Number:		33	39	40	56	99	89

5th Process Iteration

Index:	0	1	2	3	4	5
Number:	33	39	40	56	**99**	89

Comparison	Position
99>89	5

Comment: Swap data on index 2 with index 5

Index:	0	1	2	3	4	5	
Number:		33	39	40	56	89	99

So that the end result of the ascending data is 33, 39, 40, 56, 89, and 99.

Third Theoretical Analysis

In this third theoretical analysis, let's say we have array with indexes from 0 until 5. There are six numbers. Let's say these numbers are 28, 30, 37, 2, 78, 23. We want to make this numbers ascending. Please take a look on the 1st process to 5th process below. There are five process iterations:

1st Process Iteration

Index:	0	1	2	3	4	5
Number:	**28**	30	37	2	78	23

Comparison	Position
28<30	0
28<37	0
28>2	3
2<78	3
2<23	3

Comment: Swap data on index 0 with index 3

Index:	0	1	2	3	4	5	
Number:		2	30	37	28	78	23

2nd Process Iteration

Index:	0	1	2	3	4	5
Number:	2	**30**	37	28	78	23

Comparison	Position
30<37	1
30>28	3
28<78	3
28>23	5

Comment: Swap data on index 1 with index 5

Index:	0	1	2	3	4	5	
Number:		2	23	37	28	78	30

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 2, April 2023 e-ISSN : 2541-2019
DOI : https://doi.org/10.33395/sinkron.v8i2.12153 p-ISSN : 2541-044X

3rd Process Iteration

Index:	0	1	2	3	4	5
Number:	2	23	**37**	28	78	30

Comparison	Position
37>28	3
28<78	3
28<30	3

Comment: Swap data on index 2 with index 3

Index:	0	1	2	3	4	5
Number:		2	23	28	37	78	30

4th Process Iteration

Index:	0	1	2	3	4	5
Number:	2	23	28	**37**	78	30

Comparison	Position
37<78	3
37>30	5

Comment: Swap data on index 3 with index 5

Index:	0	1	2	3	4	5
Number:		2	23	28	30	78	37

5th Process Iteration

Index:	0	1	2	3	4	5
Number:	2	23	28	30	**78**	37

Comparison	Position
78>37	5

Comment: Swap data on index 4 with index 5

Index:	0	1	2	3	4	5
Number:		2	23	28	30	37	78

So that the end result of the ascending data is 2, 23, 28, 30, 37 and 78.

DISCUSSION

In the first theoretical analysis, we did the procedure step by step and we concluded that, it was conducted by hand and we got the numbers to be ascending from 6, 45, 34, 20, 100, 38 to be 6, 20, 34, 38, 45, 100. In this first theoretical analysis, we had two 'there is no changing position' of index. It was in the first process and the third process. But we had three swapping numbers. In the second theoretical analysis, again which conducted by hand, we got the numbers to be ascending from 89, 40, 33, 56, 99, 39 to be 33, 39, 40, 56, 89, 99. Here also, we had one 'there is no changing position' of index. This was in the fourth process. But we had four swapping numbers. In the third theoretical analysis, again which was conducted by hand, we got numbers to be ascending from 28, 30, 37, 2, 78, 23 to be 2, 23, 28, 30, 37, 78. But, in the third theoretical analysis, we had no 'there is no changing position'. All of them had swapping numbers. In all the analysis, the comparisons of numbers between index were decreasing from five times to be just one time.

CONCLUSION

In this journal, we have learned the standard selection sort algorithm in depth. Based on analysis above, we can conclude that selection sort is one well-known algorithm in term of sorting characters and numbers. Based on three theoretical analyses above, it needed five process iterations each to make the ascending order of numbers. By using indexes from zero to five, which means six numbers, each needs five process iterations to finish the algorithm. For future works, some comparisons between

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 2, April 2023 e-ISSN : 2541-2019
DOI : https://doi.org/10.33395/sinkron.v8i2.12153 p-ISSN : 2541-044X

selection sort algorithm and other sorting algorithm like bubble sort, heap sort, exchange sort, insertion sort, merge sort, and quick sort, can be analyzed and compared, which one is the most effective and efficient one. Other future works also can be done by implementing standard selection sort algorithm with major programming language like C++ and Python

REFERENCES

Akshay Zade, Vinod Mandloi, P. R. B. (2020). Bilateral Selection Sort. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, *8*(VII), 230–236.

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

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

Fagbola, T. M., & Thakur, S. C. (2019). Investigating the effect of implementation languages and large problem sizes on the tractability and efficiency of sorting algorithms. *International Journal of Engineering Research and Technology*, *12*(2), 196–203.

Hardika, E., Atmaja, S., & Pinaryanto, K. (2020). *Unjuk Kerja Selection Sort Hybrid*. 17–25.

Mishra, P. (2018). *A New Approach to Improve Selection Sort by the Modified Selection Sort (MSSA) and Performance Comparison*. *3*(March), 677–683.

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

Priambodo, K., & Sasongko Wibowo, J. (2021). *Implementasi Algoritma Selection Sort Untuk Perangkingan Poin Pada E-Sports Tournament Garuda League*. *2020*, 978–979. www.garudaleague.com

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.

Shabaz, M., & Kumar, A. (2019). SA sorting: A novel sorting technique for large-scale data. *Journal of Computer Networks and Communications*, *2019*. https://doi.org/10.1155/2019/3027578

Vilchez, R. N. (2019). Bidirectional Enhanced Selection Sort Algorithm Technique. *International Journal of Applied and Physical Sciences*, *5*(1), 28–35. https://doi.org/10.20469/ijaps.5.50004-1

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

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