Journal of Computer Engineering, Electronics and Information Technology (COELITE)
Volume 1, Issue 1, April 2022, pp. 09~13
e-ISSN: 2829-4149 | p-ISSN: 2829-4157

Comparison Analysis of Bubble Sort Algorithm with Tim Sort Algorithm Sorting Against the Amount of Data

Mohammad Rizal Hanafi[1], Muhammad Azfa Faadhilah[1], Deden Pradeka[1], Muhammad Taufik Dwi Putra[1]
[1]Department of Computer Engineering, Universitas Pendidikan Indonesia, Indonesia

ARTICLE INFO

Article history's:

Received : 30 Jan 2022
Revised : 28 Feb 2022
Accepted : 16 Mar 2022

Keywords:

Time Complexity
Bubble Short
Tim Sort
Algorithm Hybrid

ABSTRACT

The role of algorithms in software or programming is so important that you need to understand the basic concepts behind them. In most of our daily life, we often face problems that need to be solved by entering the data sorting process. A lot of programming logic has been created, both in the general case and in the special case. In this study, the authors carried out the sorting process with two methods, namely bubble sort and tim sort. Sorting application is built using C++ program. Algorithms are needed in order to be able to solve a problem more effectively and efficiently in a shorter time using only a few resources. The preparation of this research uses data collection methods that aim to obtain the necessary data so that it can be extracted to be used as information. It is hoped that from this information it can be seen that the comparison of the bubble sort algorithm with the tim sort is getting better in the data sorting process, if the number of inputted data is greater than n.

Muhammad Rizal Hanafi,
Department of Computer Engineering, Universitas Pendidikan Indonesia
Email: hanafi@upi.edu

1. INTRODUCTION

Information is one of the crucial things in this era. A piece of information can be sent from one corner of the earth to another in seconds because of the rapid technology in this era. The information sent contains a set of data, and these data can be used as benchmarks for making decisions and solving problems. In order to make decisions/problem solving quickly and adequately, the data must be processed first so that effective results are obtained.

A compilation process is needed in the formation or processing of data into information, which requires the data to be sorted first. This process also often appears when making software. For example, in the software for sales data, to find the highest and lowest sales data, the software must sort the existing data so that decisions can be made on the highest and lowest sales information. There are many different sorting algorithms out there. The algorithm itself is the same as the solution, where each solution step is clear and unambiguous [1]. Some problems have various algorithms (solutions). From the various existing algorithms, each has its advantages. For example, fast algorithms are not necessarily suitable for use for a long time because generally the amount of data stored in the software increases over time. For that reason, algorithm analysis is needed. Algorithm analysis aims to compare these algorithms, especially in running time, memory, and the effort required to build the algorithm [1].

Execution time or CPU time (central processing unit) is the time taken when a program or function is executed, depending on where the measurement starts and stops [2]. However, if algorithms such as bubble sort and team sort are executed on a system and then the execution time is used for comparison, it cannot be concluded directly that the team sort algorithm is better than bubble sort or vice versa because the specifications of a system are different from other systems. Therefore, an ideal calculation is needed, namely through asymptotic time complexity. Time complexity is denoted by t(n), which is an algorithm operation calculated as a function of input n, where the asymptotic time, (O), of the algorithm is called O-Big notation, the larger the value, the less effective the algorithm is [3]. Table 1 shows the categorization of big-o notation from the smallest to the largest [4].

Table 1. Categorization of algorithms based on O-Big notes.

No	Category	Description
1	$O(1)$	Constant
2	$O(log\ log\ n\ )$	Logarithmic
3	$O(n)$	Linear
4	$O(n\ log\ log\ n)$	*Log-Linear*
5	$O(n^2)$	Quadratic
6	$O(2^n)$	Exponential
7	$O(n!)$	Factorial

From research [5], compared to bubble sort, merge sort and quick sort algorithms. The results of the analysis show that the quick sort algorithm has the fastest time while bubble sort shows the longest time. For research [6], the following two algorithms were analyzed, Merge Sort and Insertion Sort. The results show that for a small amount of data, insertion sort is superior to merge sort, while for a large amount of data, merger sort is superior.

Bubble sort is a solution to the simplest sorting problem because based on the principle of its work by comparing the first element of the array to the last element where in the comparison process the swap process occurs according to conditions from lowest to highest or vice versa, while merge sort is an algorithm that works by break down the array elements into the smallest size and then put them back together in the reunification process, a sorting process occurs where when all the elements have been combined, the elements have been sorted [1].

2. RESEARCH METHOD

The preparation of this research uses data collection methods in which the aim is to obtain the necessary data so that it can be drawn into information. It is hoped that from this information it can be seen that the comparison of the bubble sort algorithm with the tim sort is better in the process of sorting data if the number of data entered is greater. The following is the method used in this research.

A. Literatur Study
This is the stage of collecting data obtained from reading scientific journals, books or even the internet (certain reliable sources).

B. Experiment
This is the stage of applying the bubble sort and tim sort algorithms. The execution time of the algorithm will be measured with the size of the data entered will be even greater.

Tools and materials to be used in the comparative research of bubble sort algorithm and tim sort. Personal Computer with specifications using Windows 10 Pro 64-bit, Intel core i7 3770k 3.5GHz (8 CPU), 16GB DDR3 PC1280 RAM, GPU AMD Radeon Rx550/500 series DDR5 4GB, using SSD for the system, with C++ programming language and compiler MinGW-W64 version 8.1.0. Then in the implementation, an increasing number of n data will be simulated to see how the two algorithms do their job. The results of the implementation will be analyzed to be drawn into information. The pseudocode for the program is as follows.

```
1.  This Program Performs Execution Time Calculations on the Sort Function
2.  Fucntion timsort(argument) {
    //Sorting
3.  }
4.  Functiion bubblesort(argument){
    //Sorting
5.  }
6.  Function main() {
7.  Int n as array length
8.    Array data1[n], data2[n]
9.
10.   Input random() to data1 and data2
11.
12. auto start1 to mark the start time
```

```
13. timsort(data1,n) // calling the function
14. auto stop1 to mark stop time
15. auto dartaion1 difference between start and stop time
16.
17. auto start2 to mark the start time
18. bubblesort(data2, n) // calling the function
19. auto stop2 to mark stop time
20. auto dartaion2  difference between start and stop time
21. print(duration1, duration2);
22. }
23.
24.
```

3. RESULTS AND DISCUSSION

3.1. Bubble Sort Algorithm

Bubble sort is a sorting algorithm by compares the selected element with the next element [10]. This type of algorithm is included in the sort comparison algorithm. This is because a comparison is made between each array element provided by the process [11]. This algorithm works by parsing array elements from left to right. next compares each element whose location is not far from the value and swaps locations if the left element is greater than the right. This process is repeated until each item is in the right order [12]. For example, in the example of a data set in the form of values 7, 3, 5, and 2, the process of sorting data using the Bubble Sort algorithm is as follows.

Table 2. Buble Sort First Process

First Process				***Process***
7	**3**	4	2	change
3	**7**	**4**	2	change
3	4	**7**	**2**	change
3	5	2	**7**	result

Table 3. Bubble Sort Second Process

Second Process				***Process***
3	5	2	7	change
3	**5**	**2**	7	change
3	2	5	7	result

Table 4. Bubble Sort Third Process

Third Process				***Process***
3	**2**	5	7	change
2	3	5	7	result

From the sample data in the table above, In the first process, starting by checking the first element on the left and the first element on the right, we can see that the value of the first element is 7 higher than 3 on the left, each element is swapped based on the results this. Then the validation process runs until each element on the left has a value less than the element on the right. The final result (third process) shows that the first element 2 displays the lowest value, and the rightmost element 7 displays the highest value [13]. It is shown in table 5 that bubble sort has the best time complexity of O(n) which occurs when the data is sorted, while for the average and worst it has a value of O(n^2) which occurs when the data is not sorted.

3.2. Tim Sort Algorithm

As obtained from [7-9] TIM Sort is a stable sorting algorithm, created based on the needs in the real environment (not created in an academic environment). This algorithm is a combination of insertion and merge. The following is the algorithm of the Tim Sort.

1. Split the array into blocks, which is called run.
2. Consider the run size, either 32 or 64.
3. Sort each element on each run using insertion sort.
4. Then combine each run as well as sort them using merge sort.

Table 5 shows the time complexity of the Tim Sort. For the best case this algorithm produces O(n) which occurs when all the data is already sorted, while for the average and worst-case it produces O(n log(n)), which occurs when the data is unsorted.

Table 5. Time complexity of bubble sort, and tim sort

Algorithm	Time Complexity		
	Best	**Average**	**Worst**
Bubble sort	$O(n)$	$O(n^2)$	$O(n^2)$
Tim sort	$O(n)$	$O(n \log(n))$	$O(n \log(n))$

A. Execution time

Fig. 1 shows the results of the comparison of Tim Sort with Bubble Sort where n data from 10 to 100 does not show a significant difference in the two algorithms. Likewise, when n is 1000, bubble sort is only 6 milliseconds longer. A significant difference begins to be seen when the size n has a length of 10,000, where the Tim sort can be quite far ahead because it completes the sort in 9 milliseconds while the bubble short takes 919 milliseconds. Meanwhile, when n is 100,000, it is very clear that the comparison of the sort succeeded in completing the sorting within 140 milliseconds, for bubble sort the time taken to complete the sorting was 67,216 milliseconds or 1 minute 7.2 seconds.

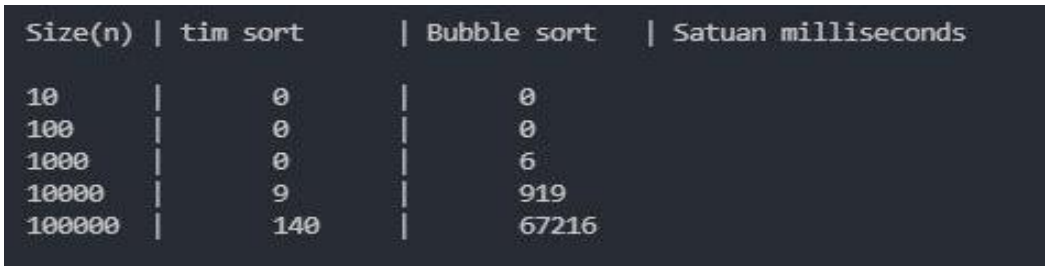

Fig. 1. Comparison of execution time of Tim sort, and bubble sort in milliseconds

While shown in **Fig. 2** in units of microseconds, when the number of n data is 10 to 100, the same sorting result is 0 microseconds. A significant difference appears when the amount of data reaches 1000 where the Tim Sort can still complete it in 0 microseconds while bubble sort takes 2997 microseconds.

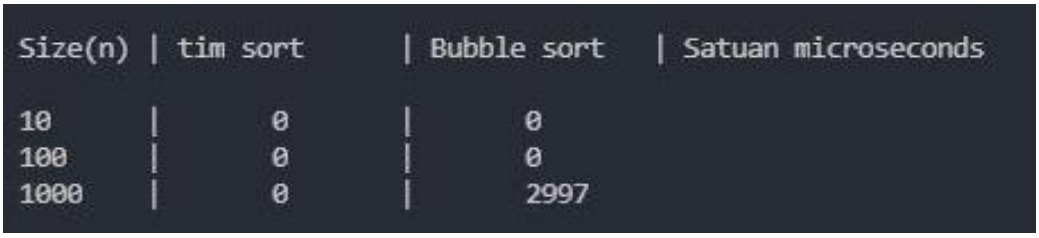

Fig 2. Comparison of execution time of Tim sort, and bubble sort in microseconds

4. CONCLUSION

Tim Sort algorithm is suitable for software with a long service life because it can be seen in the previous discussion that it shows that Tim Sort uses two algorithms, namely insertion and merge sort. This shows a powerful performance when used in software that has the goal of use over a long time. The drawback of this algorithm is the effort required to build it. Meanwhile, creating a bubble sort algorithm does not require much effort because the working principle is easy to understand. Still, this algorithm is not suitable for long-term applications where the amount of data increases over time because of the increasing amount of data, this algorithm gets slower in completing the sorting. Regarding the next research, the suggestion is to analyze further the Tim Sort algorithm with other sorting algorithms to find an algorithm that works for more varied conditions

REFERENCES

[1] N. Karumanchi, Data Structures and Algorithms Made Easy, India: Careermonk Publications, 2011.
[2] K. vivek, "Measure execution time with high precision in C/C++", geeksforgeeks, 07 Mar, 2019. Available : https://www.geeksforgeeks.org/measure-execution-time-with-high-precision-in-c-c. [Diakses : 13 Desember2021].
[3] B. Rizki Rizkyatul, "Analisis Kompleksitas Ruang Dan Waktu Terhadap Laju Pertumbuhan Algoritma Heap Sort, Insertion Sort Dan Merge Dengan Pemrograman Java", String (Satuan Tulisan Riset dan Inovasi Teknologi), Vol. 5, No. 2, pp. 109-118, july. 2020.
[4] Subandijo, "Efisiensi Algoritma Dan Notasi O-Besar", ComTech, Vol. 2, No. 2, pp. 850-858, Desember. 2011.
[5] S. Anisya, and F. Nurtaneo, "Analisis Perbandingan Algoritma Bubble Sort, Merge Sort, Dan Quick Sort Dalam Proses Pengurutan Kombinasi Angka Dan Huruf", Pseudocode, Vol. 2, No. 2 pp. 75-80, September 2015,
[6] S. H. Arief, and D. W. Sari, "Analisis algoritma insertion sort, merge sort dan implementasinya dalam bahasa pemrograman c++," vol. 8, pp. 1–8, 2012.
[7] Brandon, "Timsort — the fastest sorting algorithm you've never heard of", hackernoon, June 27, 2018. [online]. https://hackernoon.com/timsort-the-fastest-sorting-algorithm-youve-never-heard-of-36b28417f399. [Acessed 15 December 15, 2021].
[8] Admin, "TimSort", geeksforgeeks, June 26, 2021. [online]. https://www.geeksforgeeks.org/timsort. [Acessed 15 December 15, 2021].
[9] Admin, "Tim Sort Algorithm", Javatpoint, Unkown. [online]. https://www.javatpoint.com/tim-sort. [Acessed 15 December 15, 2021].
[10] S, Roma Rio dkk. 2017. "Implementasi Algoritma Bubble Sort Dan Selection Sort Menggunakan Arraylist Multidimensi Pada Pengurutan Data Multi Prioritas". Lampung : Ilmu Komputer Unila Publishing Network.
[11] S, Dwi Yulian R dkk. 2021. "Analisis Perbandingan Algoritma Bubble Sort Dan Selection Sort Pada Sistem Pendukung Keputusan Pemilihan Tempat Kost Berbasis Ios (Iphone Operating System)". Jakarta Selatan : JIPI.
[12] S. M. Cheema, N. Sarwar, and F. Yousaf, "Contrastive Analysis of Bubble & Merge Sort Proposing Hybrid Approach," 6 th INTECT, pp. 371-375, 2016
[13] Pane, Harmein dkk. 2020. "Rancang Bangun Aplikasi Kraepelin Test Berbasis Web Menggunakan Metode Bubble Sort" Jakarta Selatan : JOINTECS.