Jurnal Sistim Informasi dan Teknologi
https://jsisfotek.org/index.php
2022 Vol. 4 No. 4 Hal: 198-203 e-ISSN: 2686-3154

Perbandingan Algoritma Simple Sorting antara Penggunaan Variabel Temporary dan Tanpa Variabel Temporary

Achmad Arrosyidi[1]✉, Didiet Anindita Arnandy[2]
[1,2]Universitas Dinamika
achmad@dinamika.ac.id

Abstract

Temporary variables used in swapping technique on sorting algorithm. There are alternative swapping techniques uses pairs of + and -, × and / operators in the sorting process. This study aims to determine the duration of the sorting process uses the + and - operator pairs, the × and / operator, and the uses of temporary variables. By knowing the duration, it can determine the performance of each swapping technique. Important of this research it can contribute for advanced research or applied research. Data generated input algorithms in descending and ascending sequence. The input was sorted by 18 types of Bubble, Selection, and Insertion sort algorithm. Output sequence contrary from the data input. Duration sorting process obtained from the end processed time subtract by the start processed time. Sorting duration recapitulated to get the average duration of the sorting process so can be compared between swapping techniques. The test method has run 18 types of Simple sorting algorithms from 10, 50, and 100 data variants. frequency of each variant 10 times tests. The total data used 28,800 of integer types. Result of the average duration test that swapping uses × and / operators 306,236 milliseconds uses + and - operators 294,998 milliseconds, and swapping technique with temporary variables 294,557 milliseconds. The best performance showed by swapping technique uses temporary variables instead of uses pair of + and – operators, also used pair of × and / operators.

Keywords: Algorithm, Sorting, Variable, Temporary, Comparation.

Abstrak

Teknik swapping dalam proses *sorting* menggunakan variabel *temporary*. Terdapat alternatif teknik *swapping* yaitu menggunakan pasangan operator + dan -, serta pasangan operator × dan / dalam proses *sorting*. Penelitian ini bertujuan untuk mengetahui durasi proses *sorting* dari penggunaan pasangan operator + dan -, pasangan operator × dan /, serta penggunaan variabel *temporary*. Dengan mengetahui durasi tersebut maka dapat mengetahui performa setiap teknik *swapping*, sehingga penelitian ini penting untuk dilakukan karena dapat berkontribusi dalam penelitian tingkat lanjut ataupun penelitian terapan. Data disusun oleh algoritma *input* secara *descending*, dan *ascending*. *Input* kemudian diproses oleh 18 jenis komposisi algoritma Bubble sort, Selection sort, dan Insertion sort. *Output* yang dihasilkan kebalikan dari *input*-nya. Durasi proses *sorting* didapat dari waktu akhir dikurangi waktu mulai. Durasi *sorting* direkapitulasi sehingga diperoleh rata-rata durasi proses *sorting* untuk dibandingkan antar teknik *swapping*. Metode pengujian yakni menjalankan 18 jenis algoritma Simple sorting dengan varian data 10, 50, dan 100, dengan frekuensi setiap pengujian sebanyak 10 kali. Total data yang digunakan sebanyak 28.800 bertipe bilangan bulat. Hasil rata-rata durasi *sorting* menggunakan teknik *swapping* dengan operator × dan / 306.236 mili detik, teknik *swapping* dengan operator + dan - 294.998 mili detik, dan teknik *swapping* dengan variabel *temporary* 294.557 mili detik. Performa terbaik proses sorting ditunjukkan oleh algoritma Simple sorting yaitu Bubble sort, Selection sort, dan Insertion sort menggunakan teknik *swapping* dengan variabel *temporary*.

Kata kunci: Algoritma, *Simple*, *Sorting*, *Temporary*, Perbandingan.

1. Pendahuluan

Kecepatan adalah salah satu parameter performa algoritma *sorting* [1], [2]. Kecepatan didukung oleh variabel *temporary* ebagai penyimpanan sementara saat proses [3], [4]. Terdapat alternatif *sorting*, yaitu *swapping* menggunakan pasangan operator + dan -, serta × dan /. *Sorting* menggunakan variabel *temporary* akan menambah beban pada memori komputer [5], [6] dibandingkan *sorting* tanpa menggunakan variabel *temporary*. Sorting secara internal merupakan semua elemen yang dapat dibaca ke dalam memori pada saat yang sama sehingga proses pengurutan dilakukan dalam memori. Sorting secara eksternal memiliki dataset yang besar sehingga tidak dimuat dalam memori [7], [8], [9] dan pengurutan dilakukan dalam potongan.

Pengurutan juga dilakukan secara bubble, yaitu pertukaran data dengan data disebelahnya secara terus menerus sampai tidak ada lagi perubahan. Bubble sort menggunakan variabel *temporary* [10], [11]. Pengurutan menggunakan algoritma selection sort

Diterima: 15-12-2022 | Revisi: 21-12-2022 | Diterbitkan: 31-12-2022 | doi: 10.37034/jsisfotek.v4i2.185

adalah pengurutan dengan cara memilih atau mencari data terkecil atau terbesar dari setiap data [12]. Saat data ditemukan maka data akan diletakkan pada posisi pertama kemudian dilanjutkan dengan urutan kedua dan seterusnya sampai dengan data terakhir. Selection sort menggunakan variabel *temporary* sebagai variabel untuk menyimpan data sementara saat pertukaran dalam pengurutan data.

Pengurutkan data dengan cara menyisipkan (*insert*) data baru diantara data yang sudah ada sebelumnya dengan menggunakan kriteria tertentu (*ascending* / *descending*). Penyisipan data dilakukan secara terus menerus sampai tidak ada data yang tersisa lagi untuk disisipkan. Secara umum algoritma Insertion sort menggunakan variabel *temporary* sebagai variabel untuk menyimpan data sementara saat pertukaran dalam pengurutan data [13], [14].

Teknik pertukaran data yang menggunakan variabel *temporary* dalam menyimpan nilai x untuk sementara. Kemudian memberi nilai y di x dan kemudian *temporary* di y. Dengan cara ini, nilai ditukar. Algoritma 1 adalah contoh penerapan pertukaran data menggunakan variabel *temp*.

Algoritma 1. Variabel Temporary

```
X = 5
Y = 10
temp = x
x = y
y = temp
Print('The Value of x after swapping: ()', format(x))
Print('The Value of x after swapping: ()', format(x))
```

Teknik pertukaran data tTanpa menggunakan variabel *temporay* menggunakan operasi aritmatika dalam melakukan pertukaran. Pertukaran data tanpa variabel dapat dilakukan secara pasangan operator penjumlahan-pengurangan dengan menggantikan penggunaan variabel *temporary* dalam proses pertukaran data. Algoritma 2 adalah penerapan pertukaran data menggunakan operator penjumlahan dan pengurangan sebagai pengganti variabel *temporary*.

Algoritma 2. Proses Pertukaran Data Menggunakan Pasangan Operator Penambahan-Pengurangan

```
X = x + y
y = x – y
X = x - y
```

Penggunaan operator perkalian dan pembagian dapat menggantikan penggunaan variabel *temporary* dalam proses pertukaran data. Algoritma 3 adalah contoh penerapan pertukaran data menggunakan operator perkalian dan pembagian sebagai pengganti variabel *temporary*.

Algoritma 3. Proses Pertukaran Data Menggunakan Pasangan Operator Perkalian-Pembagian

```
X = x * y
y = x / y
X = x / y
```

Penguji kecepatan *simple sorting* Bubble *sort*, Insertion *sort*, dan Selection *sort* dengan menggunakan variabel *temporary* dapat mempengaruhi kecepatan [15], [16] serta tanpa variabel *temporary* [17], [18], [19], [20]. Durasi waktu dapat mempengaruhi kecepatan dalam menghasilkan informasi [21], [22]. Penelitian ini membantu dalam kontribusi *sorting* agar perfomanya dapat lebih optimal. Sehingga perlu untuk mengetahui performa kecepatan algoritma Simple Sorting antara menggunakan variabel *temporary* dan tanpa variabel *temporary*?

2. Metodologi Penelitian

Data uji yang telah disesiakan secara *ascending* dan *descending* dengan valid dan siap diuji. Setiap algoritma Bubble, Selection dan Insertion sort menghasilkan dua jenis urutan output yaitu *ascending*, dan *descending*. Setiap proses pengurutan memanfaatkan teknik menggunakan variabel *temporary*, dan tanpa variabel *temporary*. Penggunaan tanpa variabel *temporary* menggunakan pasangan operator + dan -, serta pasangan operator perkalian × dan /. Kombinasi ini akan menghasilkan 18 (delapan belas) jenis algoritma yaitu:

a. Bubble *sort ascending* dari *input descending*, menggunakan variabel *temporary*.
b. Bubble *sort descending* dari *input ascending*, menggunakan variabel *temporary*.
c. Bubble *sort ascending* dari *input descending*, menggunakan pasangan operator + dan –.
d. Bubble *sort descending* dari *input ascending*, menggunakan pasangan operator + dan –.
e. Bubble *sort ascending* dari *input descending*, menggunakan pasangan operator × dan /.
f. Bubble *sort descending* dari *input ascending*, menggunakan pasangan operator × dan /.
g. Selection *sort ascending* dari *input descending*, menggunakan variabel *temporary*.
h. Selection *sort descending* dari *input ascending*, menggunakan variabel *temporary*.
i. Selection *sort ascending* dari *input descending*, menggunakan pasangan operator + dan –.

j. Selection *sort descending* dari *input ascending*, menggunakan pasangan operator + dan –.

k. Selection *sort ascending* dari *input descending*, menggunakan pasangan operator × dan /.

l. Selection *sort descending* dari *input ascending*, menggunakan pasangan operator × dan /.

m. Insertion *sort ascending* dari *input descending*, menggunakan variabel *temporary*.

n. Insertion *sort descending* dari *input ascending*, menggunakan variabel *temporary*.

o. Insertion *sort ascending* dari - *input descending*, menggunakan pasangan operator + dan –.

p. Insertion *sort descending* dari *input ascending*, menggunakan pasangan operator + dan –.

q. Insertion *sort ascending* dari *input descending*, menggunakan pasangan operator × dan /.

r. Insertion *sort descending* dari *input ascending*, menggunakan pasangan operator × dan /.

Rinci percobaan dengan menggunakan 18 jenis algoritma *sorting* yang berbeda dalam setiap pemprosesan *input* berupa data *descending* menghasilkan *ouput* yang urut secara *ascending*. *Input* data *ascending* menghasilkan *ouput* yang urut secara *descending*. Proses menguji setiap algoritma dilakukan secara terbalik antara *input* dibandingkan *output*-nya. Setiap pengujian algoritma dilakukan rekapitulasi durasi pengurutan.

Frekuensi yang digunakan sebanyak 10 kali pengujian dengan menggunakan *sampling* jenuh, yaitu sebuah teknik penentuan sampel, karena semua anggota populasi digunakan sebagai sampel. Sampel jenuh merupakan sensus dari semua anggota populasi dijadikan sampel. Algoritma akan ditambahkan proses menghitung durasi proses setelah *sorting*. Proses perhitungan ini dilakukan diluar *sorting*.

Durasi yang didapat dari selisih waktu akhir dikurangi dengan waktu awal, juga melakukan proses perhitungan rata-rata dari setiap dari setiap data yang diuji, serta menghasilkan waktu rata-rata keseluruhan proses *sorting*. Setiap 18 algoritma mengurutkan data sebanyak 3 varian (10, 50, dan 100), sehingga terdapat 54 hasil pengujian. Hasil setiap pengujian yang dijalankan berturut mulai dari Bubble, Selection, dan Insertion sort.

3. Hasil dan Pembahasan

Pengujian untuk jumlah data 10 terhadap algoritma Bubble, Selection, dan Insertion sort dan penggabungan *input, output* data secara *ascending* dan *descending*. Durasi menggunakan variabel temporary sebesar 1.669 milidetik, pasangan operator + dan – sebesar 1.626,6667 milidetik, serta pasangan operator × dan / sebesar 1.650,333333 milidetik dengan pengujian dilakukan sebanyak 10 kali.

Pengujian dengan jumlah data 50 yaitu durasi menggunakan variabel *temporary* sebesar 241.995 milidetik, pasangan operator + dan – sebesar 243.456 milidetik, serta pasangan operator × dan / sebesar 101.869,3333 milidetik dengan pengujian dilakukan sebanyak 10 kali.

Pengujian dengan 100 yaitu durasi menggunakan variabel *temporary* sebesar 2.060.704 milidetik, pasangan operator + dan – sebesar 2.158.849 milidetik, serta pasangan operator × dan / sebesar 780.158,3333 milidetik dengan jumlah data 100. Pengujian dilakukan sebanyak 10 kali. Rekapitulasi hasil disajikan pada Tabel 1.

Tabel 1. Rekapitulasi Rata-rata Perbandingan Algoritma

Data	Urutan		Rata-Rata (milidetik)		
	Input	Output	Temporary	Operator	
				+ & -	× & /
10	Dsc	Asc	1.618,33	1.628	1.652,67
	Asc	Dsc	1.669	1.626,67	1.650,33
Rata-Rata			1.644	1.627,33	1.651,50
50	Dsc	Asc	101.676	101.986,30	102.684,67
	Asc	Dsc	102.062,67	101.632,30	101.973
Rata-Rata			101.869,33	101.809,30	102.328,83
100	Dsc	Asc	778.121,67	772.315,30	814.315
	Asc	Dsc	782.195	790.796,70	815.139,33
Rata-Rata			780.158,33	781.556	814.727,17

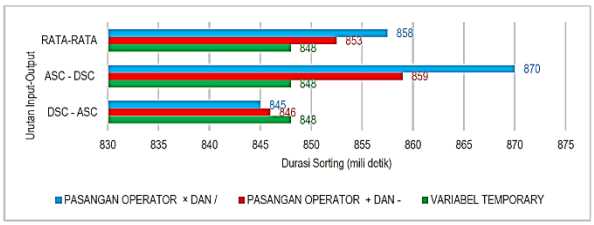
Dsc adalah *descending* dan Asc adalah *ascending*.

Dari rekapitulasi rata-rata Tabel 1 didapatkan, durasi menggunakan variabel *temporary* menggunakan varian data 10 yakni 1.643,666667 milidetik, menggunakan varian data 50 adalah 101.869,3333 milidetik, menggunakan varian data 100 selama 780.158,3333 milidetik. Durasi menggunakan operator + dan – menggunakan varian data 10 adalah 1.627,333 milidetik, menggunakan varian data 50 yaitu 101.809,3 milidetik, menggunakan varian data 100 adalah 781.556 milidetik. Durasi menggunakan operator × dan / menggunakan varian data 10 yakni 1.651,5 milidetik, menggunakan varian data 50 adalah 102.328,83 milidetik, menggunakan varian data 100 adalah 814.727,17 milidetik. Rekapitulasi perbandingan algoritma disajikan pada Tabel 2.

Tabel 2. Rekapitulasi Perbandingan Algoritma

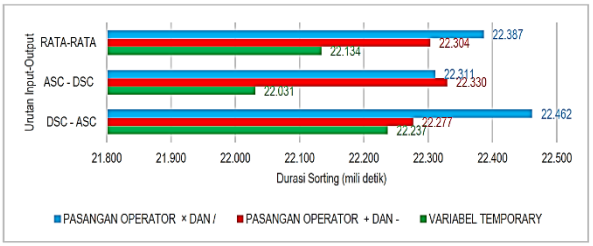
Data	Temporary	Pasangan Operator	
		+ & -	× & /
10	1.643,67	1.627,33	1.651,50
50	101.869,33	101.809,33	102.328,83
100	780.158,33	781.556	814.727,17

Analisis perbandingan Bubble Sort dengan data 10. dalam bentuk grafik pada Gambar 1.



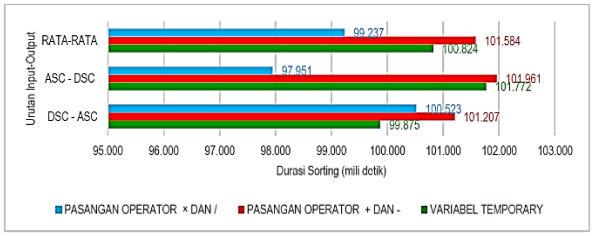
Gambar 1. Perbandingan Algoritma Bubble Sort dengan Data 10

Terdapat hasil konsisten pada *swapping* menggunakan variabel *temporary*, namun tidak pada *swapping* menggunakan operator + dan -, begitu juga dengan *swapping* menggunakan operator × dan /. Sedangkan rata-rata didapatkan durasi tercepat adalah *swapping* menggunakan variabel *temporary*. Analisis perbandingan Bubble Sort dengan Data 50 disajikan dalam bentuk grafik pada Gambar 2.



Gambar 2. Perbandingan Algoritma Bubble Sort dengan Data 50

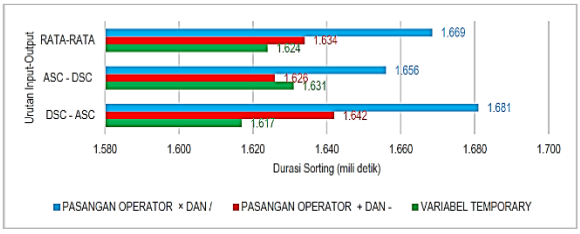
Terdapat hasil tidak konsisten pada semua teknik *swapping*. Sedangkan rata-rata didapatkan durasi tercepat adalah *swapping* menggunakan variabel *temporary*. Analisis Perbandingan Bubble Sort dengan Data 100 disajikan dalam bentuk grafik pada Gambar 3.



Gambar 3. Perbandingan Algoritma Bubble Sort dengan Data 100

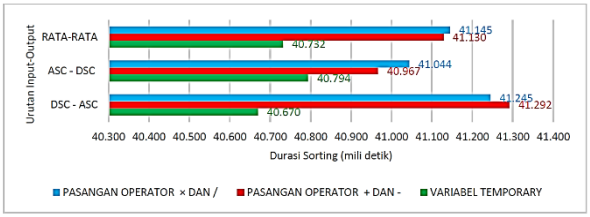
Terdapat hasil tidak konsisten pada semua teknik *swapping*. Sedangkan rata-rata didapatkan durasi tercepat adalah *swapping* menggunakan operator × dan /.

Perbandingan Selection Sort dengan data 10 disajikan dalam bentuk grafik pada Gambar 4.



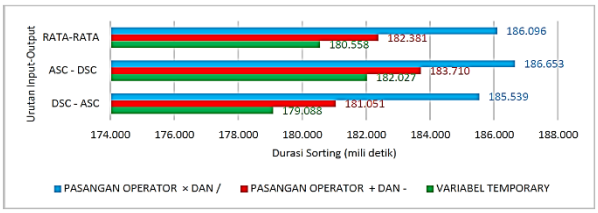
Gambar 4. Perbandingan Algoritma Selection Sort dengan Data 100

Terdapat hasil tidak konsisten pada semua teknik *swapping*. Sedangkan rata-rata didapatkan durasi tercepat adalah *swapping* menggunakan variabel *temporary*. Analisis perbandingan Selection Sort dengan data 50 disajikan dalam bentuk grafik pada Gambar 5.



Gambar 5. Perbandingan Algoritma Selection Sort dengan Data 50

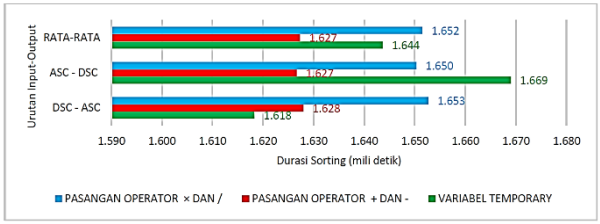
Terdapat hasil tidak konsisten pada semua teknik *swapping*. Sedangkan rata-rata didapatkan durasi tercepat adalah *swapping* menggunakan variabel *temporary*. Analisis perbandingan Selection Sort dengan data 100 disajikan dalam bentuk grafik pada Gambar 6.



Gambar 6. Perbandingan Algoritma Selection Sort dengan Data 100

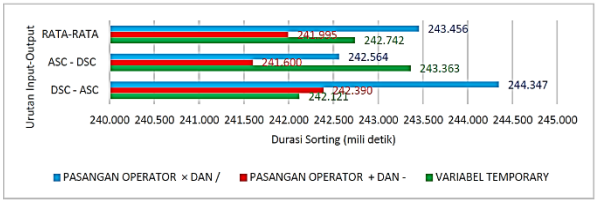
Terdapat hasil konsisten pada semua teknik *swapping*. Konsistensi tersebut pada durasi *sorting descending* ke *ascending* lebih cepat daripada durasi *sorting ascending* ke *descending*. Sedangkan rata-rata didapatkan durasi tercepat adalah *swapping* menggunakan variabel *temporary*.

Analisis perbandingan Insertion Sort dengan varian data 10 disajikan dalam bentuk grafik pada Gambar 70.



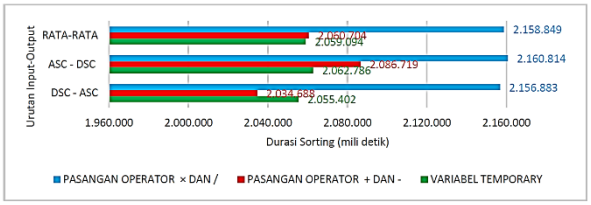
Gambar 7 Perbandingan Algoritma Insertion Sort dengan Data 100

Terdapat hasil tidak konsisten pada semua teknik *swapping*. Sedangkan rata-rata didapatkan durasi tercepat adalah *swapping* menggunakan operator + dan -. Analisis perbandingan Insertion Sort dengan data 50 disajikan dalam bentuk grafik pada Gambar 8.



Gambar 8. Perbandingan Algoritma Insertion Sort dengan Data 50

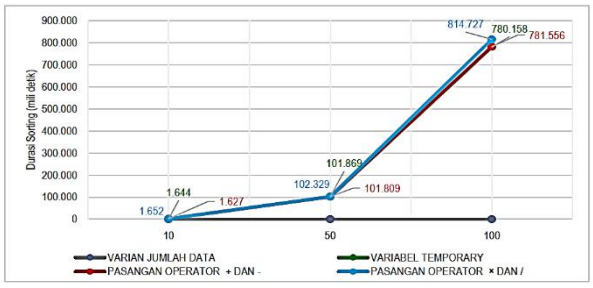
Terdapat hasil tidak konsisten pada semua teknik *swapping*. Sedangkan rata-rata didapatkan durasi tercepat adalah *swapping* menggunakan operator + dan -. Analisis perbandingan Insertion Sort dengan data 100 disajikan dalam bentuk grafik pada Gambar 92.



Gambar 9. Perbandingan Algoritma Insertion Sort dengan Data 100

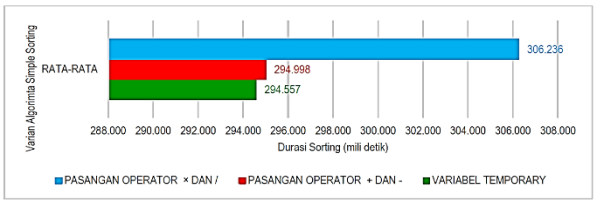
Terdapat hasil konsisten pada semua teknik *swapping*. Konsistensi tersebut pada durasi *sorting descending* ke *ascending* lebih cepat daripada durasi *sorting ascending* ke *descending*. Sedangkan rata-rata didapatkan durasi tercepat adalah *swapping* menggunakan variabel *temporary*.

Grafik perbandingan hasil *sorting* antara algoritma Bubble, Selection, dan Insertion sort menggunakan variabel *temporary*, pasangan + dan -, serta pasangan × dan / yang terdapat pada Gambar 10.



Gambar 10. Perbandingan Algoritma Bubble, Selection, dan Selection Sort dengan Data 10, 50, dan 100

Gambar 10 menunjukkan, terdapat kondisi semakin banyak data yang diurutkan, baik secara *descending* ke *ascending* ataupun dari *ascending* ke *descending* membutuhkan waktu semakin lama. Terdapat perbedaan durasi *sorting* hasil dari setiap varian jumlah data. Dari hasil analisis perbandingan setiap algoritma dapat disajikan dalam bentuk grafik pada Gambar 11.



Gambar 11. Perbandingan Rata-rata Durasi Sorting Menggunakan Algoritma Bubble, Selection, dan Insertion Sort

Gambar 11 menunjukkan, hasil pengujian rata-rata *sorting* baik menggunakan algoritma Bubble, Selection, dan Insertion sort dengan varian data yang diproses 10, 50, dan 100, serta frekuensi pengujian sebanyak 10 kali pada setiap proses *sorting*. Rata-rata durasi terlama hingga ke tercepat tersusun yaitu, durasi menggunakan teknik *swapping* menggunakan operator × dan / yaitu 306.236 mili detik, sedangkan durasi menggunakan teknik *swapping* menggunakan operator + dan - yaitu 294.998 mili detik, dan durasi menggunakan teknik *swapping* menggunakan variabel *temporary* yaitu 294.557 mili detik.

4. Kesimpulan

Sorting tercepat didapatkan oleh teknik *swapping* menggunakan variabel *temporary* dibandingkan dengan tanpa variabel *temporary*. Sehingga penelitian ini dapat membantu dalam menhasilkan informasi yang cepat dalam pengurutan data dengan menggunakan variabel *temporary*.

Ucapan Terimakasih

Terima kasih kepada Pusat Penelitian dan Pengabdian Kepada Masyarakat Universitas Dinamika, yang telah memfasilitasi dalam bentuk dana, dan persetujuannya agar penelitian ini dapat terlaksana.

Daftar Rujukan

[1] Poetra, D. R. (2022). Performa Algoritma Bubble Sort dan Quick Sort pada Framework Flutter dan Dart SDK(Studi Kasus Aplikasi E-Commerce). *JATISI (Jurnal Teknik Informatika Dan Sistem Informasi)*, *9*(2), 806–816. https://doi.org/10.35957/jatisi.v9i2.1886

[2] Maulana, M. R. (2017). Komparasi Algoritma Sorting pada Bahasa Pemrograman Java. *IC-Tech*, *12*(2).

[3] Raghuvanshi, D. (2018). Data Structure: Theoretical Approach. *International Journal of Trend in Scientific Research and Development*, *Volume-3*(Issue-1), 268–273. https://doi.org/10.31142/ijtsrd18977

[4] Anggreani, D., Wibawa, A. P., Purnawansyah, P., & Herman, H. (2020). Perbandingan Efisiensi Algoritma Sorting dalam Penggunaan Bandwidth. *ILKOM Jurnal Ilmiah*, *12*(2), 96–103. https://doi.org/10.33096/ilkom.v12i2.538.96-103

[5] Zhao, L., Liu, X., & Shao, X. (2016). *Comparative Analysis of Numerical Sorting Algorithms in Java Language*. *Icadme*, 415–419. https://doi.org/10.2991/icadme-16.2016.68

[6] Abdel-Hafeez, S., & Gordon-Ross, A. (2017). An efficient O(N) comparison-free sorting algorithm. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *25*(6), 1930–1942. https://doi.org/10.1109/TVLSI.2017.2661746

[7] Rajagopal, D., & Thilakavalli, K. (2016). Different Sorting Algorithm's Comparison based Upon the Time Complexity. *International Journal of U- and e- Service, Science and Technology*, *9*(8), 287–296. https://doi.org/10.14257/ijunesst.2016.9.8.24

[8] Sari, N., Gunawan, W. A., Sari, P. K., Zikri, I., & Syahputra, A. (2022). Analisis Algoritma Bubble Sort Secara Ascending Dan Descending Serta Implementasinya Dengan Menggunakan Bahasa Pemrograman Java. *ADI Bisnis Digital Interdisiplin Jurnal*, *3*(1), 16–23. https://doi.org/10.34306/abdi.v3i1.625

[9] Pasetto, D., & Akhriev, A. (2011). A comparative study of parallel sort algorithms. *SPLASH'11 Compilation - Proceedings of OOPSLA'11, Onward! 2011, GPCE'11, DLS'11, and SPLASH'11 Companion*, 203–204. https://doi.org/10.1145/2048147.2048207

[10] Kavdikar, P. (2021). *Comparative study of sorting algorithms*. *April*. https://doi.org/10.13140/RG.2.2.30552.62724

[11] Sunandar, E. (2019). Perbandingan Metode Selection Sort dan Insertion Sort Dalam Pengurutan Data Menggunakan Bahasa Program Java. *Petir*, *12*(2), 172–178. https://doi.org/10.33322/petir.v12i2.485

[12] Setiawan, R. (2017). Comparing sorting algorithm complexity based on control flow structure. *Proceedings of 2016 International Conference on Information Management and Technology, ICIMTech 2016*, *November*, 224–228. https://doi.org/10.1109/ICIMTech.2016.7930334

[13] Sunandar, E., & Indrianto, I. (2020). Implementasi Algoritma Bubble Sort Terhadap 2 Buah Model Varian Pengurutan Data Menggunakan Bahasa Program Java. *Petir*, *13*(2), 255–265. https://doi.org/10.33322/petir.v13i2.1008

[14] Betschart, R. O., Thiéry, A., Aguilera-Garcia, D., Zoche, M., Moch, H., Twerenbold, R., Zeller, T., Blankenberg, S., & Ziegler, A. (2022). Comparison of calling pipelines for whole genome sequencing: an empirical study demonstrating the importance of mapping and alignment. *BioRxiv*, 2022.09.18.508404. https://doi.org/10.1038/s41598-022-26181-3

[15] Bustami, B., Fadlisyah, F., & Alfyansyah, G. (2019). Comparison of Simple Algorithm Data (Sorting) Control Methods on Selection and Bubble Sort. *TECHSI - Jurnal Teknik Informatika*, *11*(2), 289. https://doi.org/10.29103/techsi.v11i2.1601

[16] 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

[17] Htwe Htwe Aung. (2019). Analysis and Comparative of Sorting Algorithms. *Published in International Journal of Trend in Scientific Research and Development*, *3*(5), 1049–1053. https://doi.org/https://doi.org/10.31142/ijtsrd26575

[18] Jmaa, Y. Ben, Atitallah, R. Ben, Duvivier, D., & Jemaa, M. Ben. (2019). A comparative study of sorting algorithms with FPGA acceleration by high level synthesis. *Computacion y Sistemas*, *23*(1), 213–230. https://doi.org/10.13053/CyS-23-1-2999

[19] Jones, T. B., & Ackley, D. H. (2014). Comparison criticality in sorting algorithms. *Proceedings of the International Conference on Dependable Systems and Networks*, 726–731. https://doi.org/10.1109/DSN.2014.74

[20] K, M. R., & J, M. R. (2018). A Comparative Study of Sorting and Searching Algorithms. *International Research Journal of Engineering and Technology (IRJET)*, *05*(01), 1412–1416. https://doi.org/10.5539/mas.v12n4p143

[21] M. Rabiu, A., J. Garba, E., Y. Baha, B., M. Malgw, Y., & Dauda, M. (2022). Performance Comparison of three Sorting Algorithms Using Shared Data and Concurrency Mechanisms in Java. *Arid-Zone Journal of Basic & Applied Research*, *1*(1), 55–64. https://doi.org/10.55639/607fox

[22] Sonita, A., & Nurtaneo, F. (2016). Analisis Perbandingan Algoritma Bubble Sort, Merge Sort, Dan Quick Sort Dalam Proses Pengurutan Kombinasi Angka Dan Huruf. *Pseudocode*, *2*(2), 75–80. https://doi.org/10.33369/pseudocode.2.2.75-80