Jurnal Ilmiah Wahana Pendidikan, Agustus 2022, 8 (13), 237-246
DOI: https://doi.org/10.5281/zenodo.6962346
p-ISSN: 2622-8327 e-ISSN: 2089-5364
Accredited by Directorate General of Strengthening for Research and Development
Available online at https://jurnal.peneliti.net/index.php/JIWP

Analisis Perbandingan Algoritma *Bubble Sort, Shell Sort,* dan *Quick Sort* dalam Mengurutkan Baris Angka Acak menggunakan Bahasa Java

Muhammad Luthfi Zulfa[1], Mikhael[2], Betha Nurina Sari[3]

[1,2,3]Universitas Singaperbangsa Karawang

Received: 14 Juli 2022
Revised: 18 Juli 2022
Accepted: 22 Juli 2022

Abstract

Sorting data is very useful because sorted data will be easier to check and correct if there are errors. There are various kinds of data sorting algorithms including bubble sort, merge sort, shell sort, radix sort, quicksort, and so on. Sorting data is the best treatment as it will be easier to check and correct if there are errors occurring. In this paper, bubble sort, shell sort, and quick sort are tested to sorting 100 lines of random integer with a value range of 0-100. The result shows that quick sort is the most efficient algorithm because it has the fewest steps, consume less memory, and take a little time while sorting.

Keywords: *Sorting, Bubble Sort, Shell Sort, Quick Sort*

(*) Corresponding Author: luthfi.zulfa18086@student.unsika.ac.id

How to Cite: Zulfa, M., Mikhael, M., & Sari, B. (2022). Analisis Perbandingan Algoritma Bubble Sort, Shell Sort, dan Quick Sort dalam Mengurutkan Baris Angka Acak menggunakan Bahasa Java. *Jurnal Ilmiah Wahana Pendidikan*, *8*(13), 237-246. https://doi.org/10.5281/zenodo.6962346

PENDAHULUAN

Lahirnya komputer telah membawa banyak perubahan dalam kehidupan sehari-hari, salah satu contohnya adalah kegiatan menyimpan data yang mana pada awalnya data disimpan di dalam indeks kini data secara masif disimpan ke dalam komputer. Seiring berjalannya waktu maka akan semakin banyak pula data yang akan disimpan ke dalam komputer. Contoh data yang umumnya akan disimpan ke dalam komputer adalah biodata, kuesioner, sensus penduduk, data internal perusahaan, dan lain sebagainya. Dengan demikian, masalah umum akibat masifnya jumlah data adalah sulitnya mengakses data akibat data tidak tersusun secara rapi sehingga perlu adanya pengelolaan data.

Jenis pengelolaan data yang paling umum adalah pengurutan data. Pengurutan data memegang peranan yang penting pada permasalahan pengelolaan data (Sonita & Nurtaneo, 2015). Hal ini dikarenakan data yang terurut akan mudah untuk dicari, diperiksa, serta dilakukan perbaikan jika terjadi kesalahan (Sari et al., 2022).

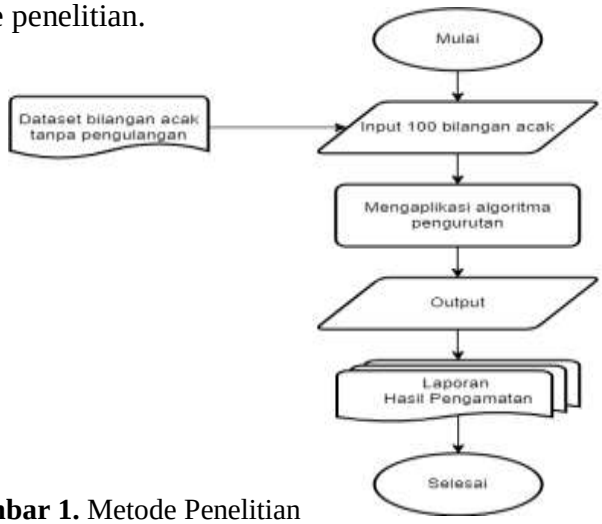
Pengurutan data atau sering disebut dengan *sorting* merupakan proses menyusun baris/deret angka atau karakter yang sebelumnya acak menjadi tersusun secara teratur dengan pola tertentu misalnya tersusun secara naik (*ascending*) atau menurun (*descending*) (Tambunan et al., 2018). Pengurutan data sangat berguna karena data yang terurut akan lebih mudah diperiksa dan diperbaiki jika terdapat kesalahan. Ada berbagai macam algoritma pengurutan data di antaranya adalah *bubble sort*, *merge sort*, *shell sort*, *radix sort*, *quicksort*, dan sebagainya. Namun demikian, pemilihan algoritma pengurutan biasanya didasarkan pada kesederhanaan kecepatan pemrosesan selama pengurutan (Sunandar & Indrianto, 2020).

Ada beberapa penelitian yang membandingkan performa algoritma dalam pengurutan data. Penelitian yang dilakukan oleh Sonita & Nurtaneo (2015) membandingkan algoritma *bubble sort*, *merge sort*, dan *quick sort*. Penelitian ini menyimpulkan algoritma *quick sort* merupakan algoritma dengan proses waktu tercepat. Penelitian lain oleh Sari et al. (2022) menganalisis algoritma pengurutan *bubble sort* secara *ascending* dan *descending*. Dalam penelitian ini menyimpulkan algoritma *bubble sort* akan semakin lambat prosesnya seiring dengan besarnya jumlah data. Kemudian penelitian oleh Tambunan et al. (2018) yang berjudul "Optimasi Algoritma Shell Sort dalam Pengurutan Data Huruf dan ANGKA" menerapkan algoritma *shell sort* untuk mengurutkan angka dan huruf dengan waktu pemrosesan. Kemudian penelitian yang dilakukan oleh Sunandar & Indrianto (2020) yang mengimplementasikan algoritma *bubble sort* terhadap dua buah model varian data menggunakan Bahasa Java. Kemudian penelitian oleh Sharma (2019) membandingkan performa Bahasa pemrograman Java dan C++ di mana diterapkan implementasi algoritma *quick sort* dan *insertion sort* untuk mengetahui waktu pemrosesan yang lebih cepat antara kedua Bahasa pemrograman tersebut.

Berdasarkan beberapa penelitian sebelumnya yang telah dipaparkan di atas, performa algoritma pengurutan hanya diukur berdasarkan waktu pemrosesan. Penelitian ini bertujuan untuk mengimplementasikan dan membandingkan performa algoritma *bubble sort*, *shell sort*, dan *quick sort* berdasarkan banyaknya langkah, waktu pemrosesan, dan konsumsi memori yang dilakukan dalam pengurutan angka secara *ascending* dan *descending*.

METODE PENELITIAN

Metode/tahapan penelitian dimulai dengan membuat dataset (file .csv) yang berisi 100 baris bilangan acak tanpa pengulangan (*repetition*). Adapun 100 baris bilangan acak ini didapatkan dengan memanfaatkan situs *online* https://www.calculatorsoup.com/. Kemudian algoritma pengurutan (*bubble sort*, *shell sort*, dan *quick sort*) diaplikasikan terhadap dataset untuk dilakukan pengurutan baik secara *ascending* (urutan naik) maupun secara *descending* (urutan menurun). Masing-masing pengaplikasian algoritma dilakukan sebanyak lima kali, kemudian akan menghasilkan *output* berupa banyak langkah pengurutan, konsumsi memori, dan waktu pemrosesan. Kemudian langkah terakhir adalah mencatat hasil pengamatan *output* menjadi sebuah laporan penelitian. Gambar 1 merupakan ilustrasi dari metode penelitian.



Gambar 1. Metode Penelitian

KAJIAN PUSTAKA

Algoritma

Algoritma merupakan strategi atau serangkaian instruksi untuk menyelesaikan suatu permasalahan pada bidang komputasi dengan mengaplikasikan program komputer (Mushtofa, et al., 2021).

Bubble Sort

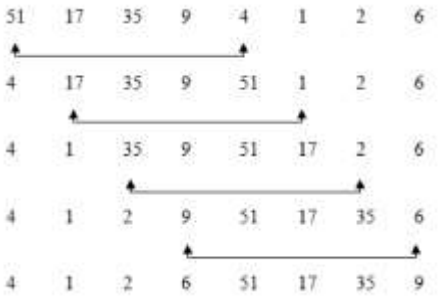
Bubble sort merupakan salah satu algoritma pengurutan yang tergolong dalam "*comparison sort*" karena langkah algoritma ini melakukan operasi perbandingan antar elemennya (Saptadi & Arief, 2013). Algoritma ini merupakan teknik paling dasar dan paling sederhana dibandingkan dengan algoritma pengurutan yang lain. Proses pengurutan algoritma ini dilakukan secara kasar (*brute force*) di mana semua elemen *array* akan dibandingkan satu per satu secara sekuensial hingga mendapatkan urutan secara *ascending* atau *descending*.

Di bawah ini merupakan *pseudocode bubble sort* dalam mengurutkan secara *ascending*:

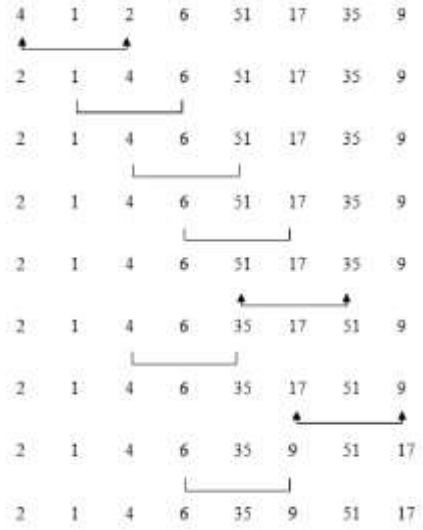
```
Define Array [0...n]
 For i = 0 to length(Array)
      For j = i to length(Array)
            If Array[i] > Array[j] then:
                  temp = Array[i]
                  Array[i] = Array[j]
                  Array[j] = temp
            End If
      End For
End For
```

Shell Sort

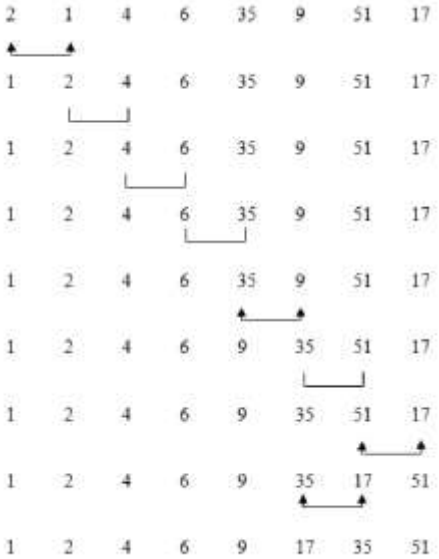
Algoritma *shell sort* adalah algoritma yang ditemukan oleh Donald L. Shell pada tahun 1959. Algoritma ini bekerja dengan cara membandingkan elemen *array* pada interval tertentu, kemudian dilakukan pertukaran (*swap*) jika diperlukan (Tambunan, et al., 2018). Interval atau sering disebut dengan *gap* pada algoritma *shell sort* biasanya dimulai sebesar setengah dari panjang *array* (tahap N/2), kemudian interval akan diperkecil sebesar setengah dari interval sebelumnya, begitu seterusnya. Gambar 2, Gambar 3, dan Gambar 4 merupakan tahapan algoritma *shell sort* dengan pengurutan secara *ascending* (Rachmat, 2018).



Gambar 2. Tahap [N/2]



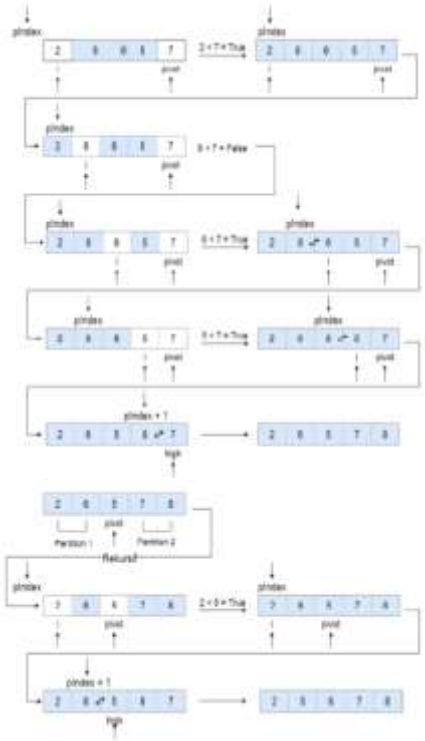
Gambar 3. Tahap [N/4]



Gambar 4. Tahap [N/8]

Quick Sort

Algoritma *Quick Sort* merupakan algoritma yang dikembangkan oleh ilmuwan komputer berkebangsaan Inggris, Tony Hoare pada tahun 1959 dan dipublikasikan pada tahun 1961 (Hoare, 1961). *Quick sort* merupakan algoritma pengurutan secara rekursif yang menggunakan metode "*divide and conquer"* untuk proses pengurutan sebagian kecil dari dataset kemudian menggabungkan kembali di proses akhir. Algoritma ini sangat efektif pada *array* yang berukuran sangat besar (Sharma, 2019). Gambar 5 merupakan ilustrasi pengurutan dengan *quick sort* (Poetra & Hayati, 2022)



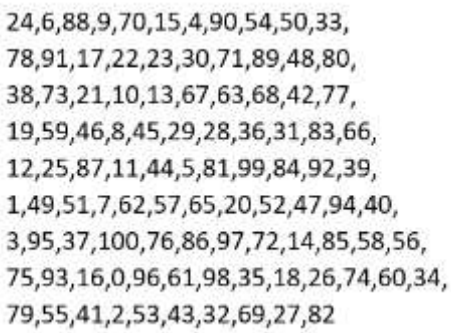
Gambar 5. Pengurutan dengan *quick sort*

HASIL DAN PEMBAHASAN

Penelitian ini dilakukan menggunakan Bahasa Java versi 17.0.1 dengan memanfaatkan perangkat *open source* NetBeans IDE 13. Adapun perangkat yang digunakan selama pengujian adalah komputer PC Windows 10 64-bit dengan spesifikasi:

1. Processor: AMD Ryzen 3 3250U with Radeon Graphics 4 CPU @2.60GHz
2. Memory: 8GB RAM
3. SSD 1TB

Adapun objek penelitian yang digunakan merupakan 100 baris angka acak dengan rentang nilai 0-100 yang dibuat dengan memanfaatkan *website* https://www.calculatorsoup.com/. Ilustrasi dataset dapat dilihat pada Gambar 6.



Gambar 6. Ilustrasi Dataset

Pada penelitian ini, setiap algoritma akan diuji sebanyak lima kali tes. Penilaian tiap algoritma dihitung berdasarkan banyaknya langkah pengurutan, konsumsi memori (Kilobytes (KB)), dan waktu pemrosesan (sekon (s)).

Hasil Pengujian *Bubble Sort*

Hasil pengujian *bubble sort* ditunjukkan oleh Tabel 1 dan Tabel 2 di mana Tabel 1 merupakan pengujian secara *ascending* dan Tabel 2 merupakan pengujian secara *descending*.

Tabel 1. Pengujian *bubble sort* secara *ascending*

Test ke-	Banyak Langkah	Konsumsi Memori (KB)	Waktu Pemrosesan (s)
1	2276	4.930,904	13,159
2	2276	4.926,792	14,182
3	2276	4.935,000	11,339
4	2276	4.982,848	12,188
5	2276	4.924,456	12,015
Rata-rata		4.940,000	12,577

Tabel 2. Pengujian *bubble sort* secara *descending*

Test ke-	Banyak Langkah	Konsumsi Memori (KB)	Waktu Pemrosesan (s)
1	2674	7.519,512	14,401
2	2674	7.527,024	15,572
3	2674	7.549,448	13,127
4	2674	7.539,528	15,392
5	2674	7.519,512	13,520
Rata-rata		7.531,005	14,402

Hasil Pengujian *Shell Sort*

Hasil pengujian *shell sort* ditunjukkan oleh Tabel 3 dan Tabel 4 di mana Tabel 3 merupakan pengujian secara *ascending* dan Tabel 4 merupakan pengujian secara *descending*.

Tabel 3. Pengujian *shell sort* secara *ascending*

Test ke-	Banyak Langkah	Konsumsi Memori (KB)	Waktu Pemrosesan (s)
1	444	5.242,880	3,943
2	444	5.242,880	2,564
3	444	5.242,880	2,345
4	444	5.242,880	2,685
5	444	5.242,880	2,726
Rata-rata		5.242,880	2,853

Tabel 4. Pengujian *shell sort* secara *descending*

Test ke-	Banyak Langkah	Konsumsi Memori (KB)	Waktu Pemrosesan (s)
1	406	5.015,312	2,750
2	406	5.015,312	2,077
3	406	5.015,312	2,127
4	406	5.015,312	2,320
5	406	5.015,312	2,636
Rata-rata		5.015,312	2,382

Hasil Pengujian *Quick Sort*

Hasil pengujian *quick sort* ditunjukkan oleh Tabel 5 dan Tabel 6 di mana Tabel 5 merupakan pengujian secara *ascending* dan Tabel 6 merupakan pengujian secara *descending*.

Tabel 5. Pengujian *quick sort* secara *ascending*

Test ke-	Banyak Langkah	Konsumsi Memori (KB)	Waktu Pemrosesan (s)
1	298	4.194,304	1,655
2	298	4.194,304	1,650
3	298	4.194,304	1,607
4	298	4.194,304	1,683
5	298	4.194,304	1,580
Rata-rata		4.194,304	1,635

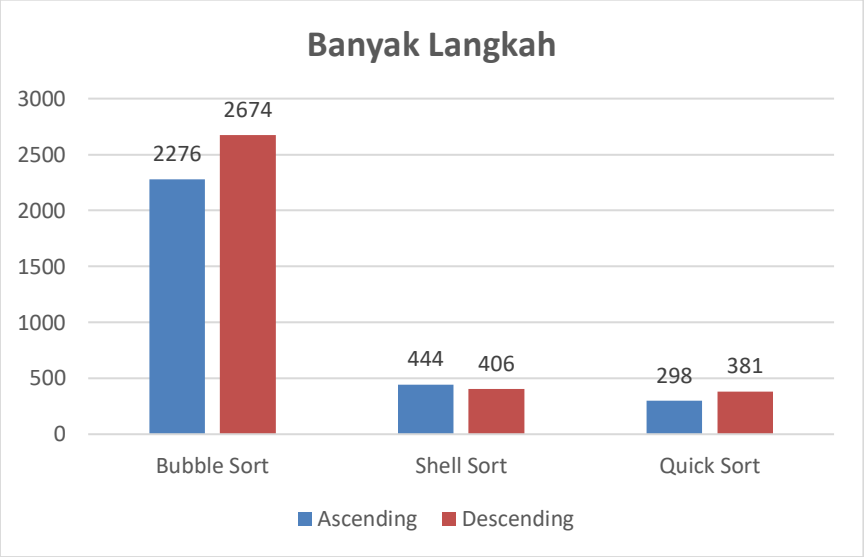
Tabel 6. Pengujian *quick sort* secara *descending*

Test ke-	Banyak Langkah	Konsumsi Memori (KB)	Waktu Pemrosesan (s)
1	381	4.741,648	2,038
2	381	4.741,648	1,992
3	381	4.741,648	2,094

4	381	4.741,648	2,039
5	381	4.741,648	2,029
Rata-rata		4.741,648	2,038

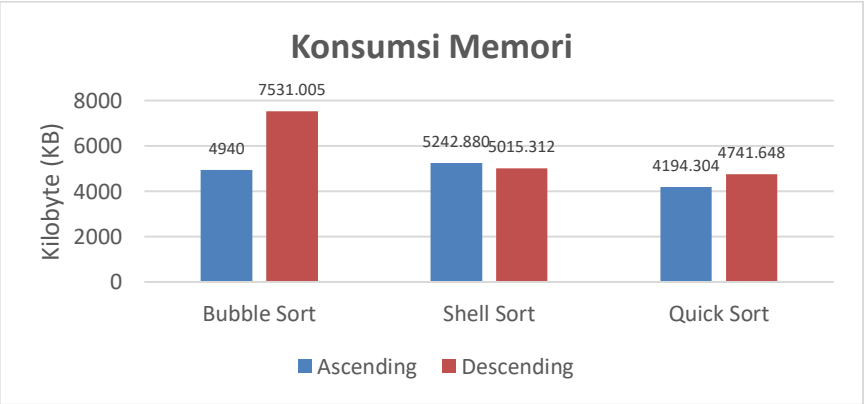
Diagram Perbandingan

Gambar 7 merupakan perbandingan banyaknya langkah yang dilakukan selama pengurutan. Langkah dihitung jika terjadi pertukaran pada elemen *array* hingga proses pengurutan selesai. Berdasarkan Gambar 7, algoritma *quick sort* paling efisien dalam pengurutan baik secara *ascending* maupun *descending* karena memiliki langkah yang lebih singkat



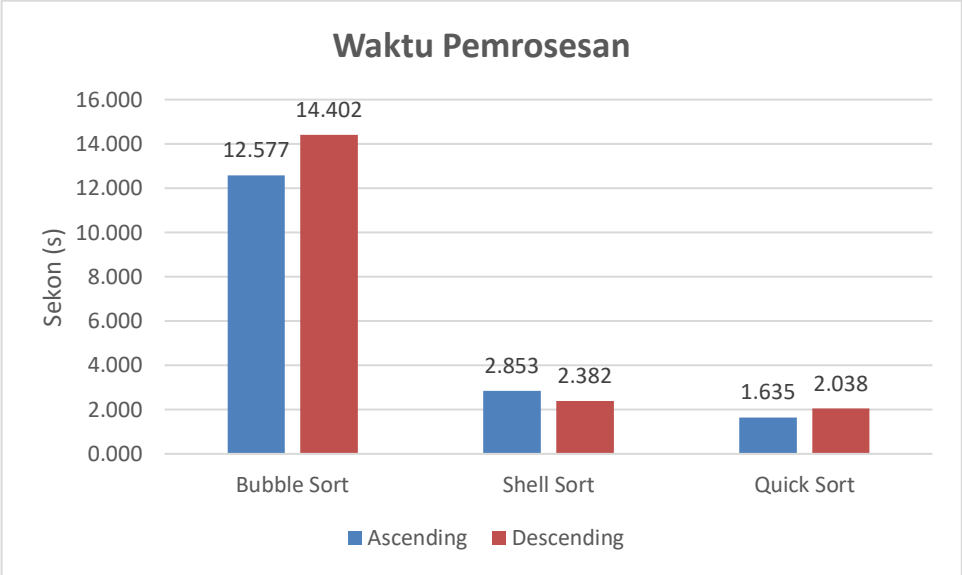
Gambar 7. Langkah Pengurutan

Gambar 8 merupakan perbandingan konsumsi memori selama proses pengurutan. Konsumsi memori dihitung dengan satuan Kilobyte (KB). Pada Gambar 8, disimpulkan bahwa *quick sort* merupakan algoritma paling efisien dalam mengkonsumsi memori. Hal ini dikarenakan algoritma *quick sort* menggunakan memori yang paling kecil dibandingkan algoritma lain.



Gambar 8. Konsumsi Memori Selama Pengurutan

Gambar 9 merupakan grafik perbandingan waktu pemrosesan ketiga algoritma selama pengurutan. Algoritma *quick sort* merupakan algoritma paling cepat dalam pengurutan baik secara *ascending* maupun *descending* karena memiliki waktu pemrosesan yang paling kecil dibandingkan algoritma lain.



Gambar 9. Waktu Pemrosesan Selama Pengurutan

KESIMPULAN

Berdasarkan hasil pengujian pada penelitian ini, maka dapat disimpulkan algoritma *quick sort* merupakan algoritma dengan performa terbaik dalam mengurutkan data acak baik secara *ascending* maupun *descending* dibandingkan dengan algoritma *bubble sort* dan *shell sort*. Hal ini ditunjukkan algoritma *quick sort* menggunakan lebih sedikit pertukaran elemen *array*, mengkonsumsi sedikit memori dan memiliki waktu pemrosesan yang lebih cepat dibandingkan algoritma lain.

REFERENCES

Hoare, C. A. R. (1961). Algorithm 64: Quicksort. *Communications of the ACM*, *4*(7), 321.

Mushthofa, Wahyono, Asfarian, A., Ramadhan, D. A., Putro, H. P., Wisnubhadra, I.,…Saputra, B. (2021). Informatika untuk SMA Kelas X. *Jakarta: Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi*.

Rachmat, N. (2018). Perbandingan Bubble Sort, Shell Sort dan Kombinasi Bubble Sort Dengan Shell Sort. *Jusikom : Jurnal Sistem Komputer Musirawas*, *3*(1), 64–71.

Rizka Poetra, D., & Hayati, N. (2022). Performa Algoritma Bubble Sort Dan Quick Sort Pada Framework Flutter Dan Dart SDK (Studi Kasus Aplikasi E-Commerce). *Jurnal Teknik Informatika Dan Sistem Informasi*, *9*(2), 806–816.

Saptadi, H., Arief. (2012). Analisis Algoritma Insertion sort, Merge sort dan Implementasinya dalam Bahasa Pemrograman C++. *Jurnal Infotel*, *4*(2), 10-17.

Sari, N., Gunawan, W. A., Sari, P. K., Zikri, I., & Syahputra, A. (2022). Analisis Algoritma Bubble Sort Secara Ascending Dan Descending Serta Implementasinya Menggunakan Bahasa Pemrograman Java. *Abdi Jurnal*, *3*(1), 15–23.

Satria Tambunan, H., Gunawan, I., Irawan, E., & Tunas Bangsa, S. (2018). Optimasi Algoritma Shell Sort dalam Pengurutan Data Huruf dan Angka. *Junal Sistem Informasi Ilmu Komputer Prima*, *2*(1), 23–27.

Sharma, S. (2019). Performance *comparison of Java and C++ when sorting integers and writing/reading files*.

Sonita, A., & Nurtaneo, F. (2015). Analisis Perbandingan Algoritma Bubble Sort, Merge Sort, dan Quick Sort dalam Proses Pengurutan Kombinasi Angka dan Huruf. *Jurnal Pseudocode*, *2*(2), 75–80.

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.