JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 3, No. 12, Tahun 2026
ISSN 3025-0919 (media online)
Hal 3049-3057

Pengaruh Karakteristik Data Terhadap Performa Algoritma Sorting

Nayla Anjani Nasution[1*], Natasha Patricia Nainggolan[2], Zevan Irfandi Surbakti[3], Adidtya Perdana[4]

[1]Fakultas Matematika dan Ilmu Pengetahuan Alam, Program Studi Ilmu Komputer, Universitas Negeri Medan, Medan, Indonesia
Email: [1*]naylanjaninst26@gmail.com , [2]nattadsah@gmail.com, [3]zevanirfandisurbakti@gmail.com
(* : coressponding author)

Abstrak− Algoritma sorting merupakan salah satu komponen fundamental dalam ilmu komputer yang banyak digunakan dalam berbagai proses pengolahan data. Pemilihan algoritma sorting yang tepat menjadi sangat penting karena setiap algoritma memiliki karakteristik performa yang berbeda tergantung pada kondisi data yang diproses. Penelitian ini bertujuan untuk menganalisis pengaruh karakteristik data terhadap performa beberapa algoritma sorting. Algoritma yang diuji dalam penelitian ini adalah Insertion Sort, Merge Sort, dan Quick Sort. Metode penelitian yang digunakan adalah eksperimen komputasional dengan mengimplementasikan ketiga algoritma tersebut menggunakan bahasa pemrograman Python. Pengujian dilakukan pada empat jenis karakteristik data, yaitu random data, sorted data, reversed data, dan nearly sorted data, dengan variasi ukuran dataset mulai dari 1.000 hingga 10.000 elemen. Waktu eksekusi algoritma diukur menggunakan fungsi time.perf_counter() untuk memperoleh hasil pengukuran yang presisi. Hasil penelitian menunjukkan bahwa karakteristik data memiliki pengaruh yang signifikan terhadap performa algoritma sorting. Insertion Sort menunjukkan performa yang baik pada dataset kecil dan data yang hampir terurut, namun kurang efisien pada dataset berukuran besar karena kompleksitas waktunya $O(n^2)$. Sebaliknya, Merge Sort dan Quick Sort menunjukkan performa yang lebih stabil pada berbagai kondisi dataset dengan kompleksitas rata-rata O(n log n). Temuan penelitian ini diharapkan dapat membantu dalam menentukan algoritma sorting yang paling sesuai berdasarkan karakteristik data yang diproses.

Kata kunci: algoritma sorting, insertion sort, merge sort, quick sort, performa algoritma.

Abstract− *Sorting algorithms are a fundamental component of computer science that are widely used in various data processing processes. Selecting the right sorting algorithm is crucial because each algorithm has different performance characteristics depending on the data conditions being processed. This study aims to analyze the influence of data characteristics on the performance of several sorting algorithms. The algorithms tested in this study are Insertion Sort, Merge Sort, and Quick Sort. The research method used is a computational experiment by implementing the three algorithms using the Python programming language. Tests were conducted on four types of data characteristics: random data, sorted data, reversed data, and nearly sorted data, with dataset sizes ranging from 1,000 to 10,000 elements. Algorithm execution time was measured using the time.perf_counter() function to obtain precise measurement results. The results show that data characteristics have a significant influence on the performance of sorting algorithms. Insertion Sort shows good performance on small datasets and nearly sorted data, but is less efficient on large datasets due to its $O(n^2)$ time complexity. In contrast, Merge Sort and Quick Sort demonstrated more stable performance across a variety of datasets, with an average complexity of O(n log n). The findings of this study are expected to help determine the most appropriate sorting algorithm based on the characteristics of the data being processed.*

Keywords: *sorting algorithm, insertion sort, merge sort, quick sort, algorithm performance.*

1. PENDAHULUAN

Algoritma pengurutan (*sorting*) merupakan salah satu konsep dasar yang paling penting dan paling sering digunakan dalam ilmu komputer. Proses pengurutan data diterapkan secara luas dalam berbagai bidang, mulai dari pengolahan data, pencarian informasi, sistem basis data, hingga pengembangan perangkat lunak modern. Data yang sudah terurut memungkinkan proses pencarian dan pengolahan informasi dilakukan dengan jauh lebih cepat dan efisien sebagaimana halnya metode *binary search* yang hanya bisa bekerja pada data dalam kondisi terurut (Goodrich et al., 2024).

Seiring dengan meningkatnya volume data yang harus diproses oleh sistem komputasi modern, pemilihan algoritma pengurutan yang tepat menjadi suatu keharusan. Setiap algoritma sorting memiliki karakteristik dan tingkat efisiensi yang berbeda tergantung pada ukuran dan kondisi data yang diproses. Beberapa algoritma mungkin bekerja secarqa optimal pada dataset berukuran

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 3, No. 12, Tahun 2026
ISSN 3025-0919 (media online)
Hal 3049-3057

kecil, namun menjadi tidak efisien ketika dihadapkan pada dataset berukuran besar atau dengan data tertentu (Bakare et al., 2024).

Tiga algoritma sorting yang menjadi fokus penelitian ini adalah *Insertion Sort*, *Merge Sort*, dan *Quick Sort*. *Insertion Sort* merupakan algoritma sederhana dengan kompleksitas $O(n^2)$ pada kasus rata-rata, namun mampu mencapai O(n) pada kondisi data telah atau hampir terurut. *Merge* dan *Sort Quick* merupakan algoritma berbasis strategi *divide* and *conquer* dengan kompleksitas waktu rata-rata O(n log n), sehingga dikenal lebih efisien untnuk dataset berukuran besar (Bhargava, 2024).

Meski kompleksitas teoritis ketiga algoritma tersebur telah banyak diuji dalam literatur, performa aktual sebuah algoritma pada kondisi nyata seringkali dipengaruhi oleh karakteristik data yang diproses. Fenomena *best-case* dan *worst-case* dari masing-masing algoritma perlu dibuktikan secara empiris untuk memberikan gambaran yang komprehensif. Oleh karena itu, penelitian ini dilakukan untuk mengukur dan membandingkan waktu eksekusi ketiga algoritma tersebut secara langsung menggunakan berbagai kondisi data sintetis.

Penelitian ini menggunakan empat skenario karakteristik data, yaitu data acak *(random)*, data terurut *(sorted)*, data terurut terbalik *(reversed)*, dan data hampir terurut *(nearly sorted)*, dengan variasi ukuran dataset mulai dari 1.000 hingga 10.000 elemen. Eksperimen dilakukan menggunakan bahasa pemrograman Python dengan pengukuran waktu berbasis fungsi time.per_counter() untuk memperoleh resolusi waktu yang tinggi dan hasil yang dapat direproduksi.

Hasil dari penelitian ini diharapkan dapat memberikan panduan praktis bagi para pengembang perangkat lunak dalam memilih algoritma sorting yang paling sesuai berdasarkan karakteristik data yang merea hadapi, sehingga efisiensi proses pengolahan data dapat ditingkatkan secara optimal.

2. METODE

2.1 Jenis dan Pendekatan Penelitian

Penelitian ini menggunakan pendekatan eksperimen kuantitatif berbasis komputasi (computational experimental research), yang bertujuan untuk mengkaji secara empiris pengaruh karakteristik data terhadap performa algoritma pengurutan. Pendekatan ini dipilih karena memungkinkan proses pengukuran kinerja algoritma dilakukan secara objektif, terkontrol, dan dapat direplikasi.

Eksperimen dilakukan dengan mengimplementasikan algoritma secara langsung, kemudian menguji kinerjanya pada berbagai skenario dataset yang telah dirancang sebelumnya. Hasil pengujian diukur dalam bentuk waktu eksekusi, yang selanjutnya dianalisis untuk mengidentifikasi pola performa algoritma pada kondisi data yang berbeda.

2.2. Objek Penelitian

Objek dalam penelitian ini adalah performa tiga algoritma pengurutan, yaitu Insertion Sort, Merge Sort, dan Quick Sort. Pemilihan ketiga algoritma tersebut didasarkan pada perbedaan karakteristik kompleksitas waktu yang dimilikinya, sehingga dapat memberikan gambaran komparatif yang representatif.

Insertion Sort merupakan algoritma dengan kompleksitas waktu kuadratik , yang cenderung sensitif terhadap ukuran data. Sementara itu, Merge Sort dan Quick Sort merupakan algoritma berbasis divide and conquer dengan kompleksitas waktu rata-rata , yang secara teoritis lebih efisien untuk dataset berukuran besar. Perbedaan ini menjadi landasan dalam menganalisis pengaruh karakteristik data terhadap kinerja masing-masing algoritma.

2.3. Jenis dan Karakteristik Data

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 3, No. 12, Tahun 2026
ISSN 3025-0919 (media online)
Hal 3049-3057

Data yang digunakan dalam penelitian ini merupakan data sintetis yang dihasilkan secara terprogram guna memastikan kontrol penuh terhadap distribusi dan struktur data. Penggunaan data sintetis memungkinkan peneliti untuk menciptakan kondisi pengujian yang konsisten dan bebas dari bias eksternal.

Karakteristik data yang diuji dalam penelitian ini meliputi empat kategori utama, yaitu:

1. Data acak (random data), yang tidak memiliki pola tertentu dan merepresentasikan kondisi umum.
2. Data terurut (sorted data), yang mencerminkan kondisi terbaik bagi beberapa algoritma tertentu.
3. Data terbalik (reversed data), yang sering kali menjadi kondisi terburuk bagi algoritma sederhana seperti Insertion Sort.
4. Data hampir terurut (nearly sorted data), yang merepresentasikan kondisi realistis di mana data telah mengalami sebagian proses pengurutan.

Keempat karakteristik tersebut dipilih untuk memberikan cakupan kondisi yang komprehensif dalam mengevaluasi performa algoritma.

2.4 Ukuran dan Variasi Data

Pengujian dilakukan dengan beberapa variasi jumlah data, yaitu:

- 1.000 data
- 3.000 data
- 5.000 data
- 7.000 data
- 10.000 data

Variasi ukuran data ini digunakan untuk melihat pengaruh peningkatan jumlah data terhadap waktu eksekusi masing-masing algoritma.

2.5 Teknik Pengumpulan Data

Pengumpulan data dilakukan melalui proses eksekusi program yang telah diimplementasikan menggunakan bahasa pemrograman Python. Setiap algoritma dijalankan pada dataset dengan karakteristik dan ukuran yang identik untuk memastikan konsistensi pengujian.

Pengukuran waktu eksekusi dilakukan menggunakan fungsi time.perf_counter() yang tersedia dalam Python, yang memiliki resolusi tinggi dalam mengukur durasi proses komputasi. Waktu eksekusi dihitung sejak algoritma mulai dijalankan hingga proses pengurutan selesai.Untuk meningkatkan validitas dan reliabilitas data, setiap skenario pengujian dilakukan secara berulang, kemudian nilai waktu eksekusi yang diperoleh dicatat dan dianalisis.

2.6 Teknik Analisis Data

Analisis data dilakukan dengan pendekatan komparatif, yaitu membandingkan waktu eksekusi antar algoritma pada setiap variasi karakteristik dan ukuran data. Hasil pengujian kemudian dianalisis untuk:

1. Mengidentifikasi pola performa algoritma pada kondisi data tertentu
2. Menentukan algoritma yang paling efisien pada masing-masing skenario
3. Membandingkan hasil empiris dengan kompleksitas waktu teoritis

Data yang diperoleh disajikan dalam bentuk tabel dan grafik untuk mempermudah interpretasi serta memperjelas perbedaan performa antar algoritma.

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 3, No. 12, Tahun 2026
ISSN 3025-0919 (media online)
Hal 3049-3057

2.7 Prosedur Penelitian

Prosedur penelitian dilakukan secara sistematis melalui beberapa tahapan sebagai berikut:

1. Melakukan studi literatur terkait algoritma sorting dan analisis kompleksitas
2. Merancang serta mengimplementasikan algoritma Insertion Sort, Merge Sort, dan Quick Sort menggunakan bahasa pemograman Python
3. Membuat dataset dengan berbagai karakteristik yang telah ditentukan
4. Melakukan pengujian algoritma pada setiap dataset
5. Mengukur dan mencatat waktu eksekusi
6. Mengolah serta menganalisis data hasil pengujian
7. Menyusun laporan penelitian dalam bentuk artikel ilmiah

3. ANALISA DAN PEMBAHASAN

Hasil eksperimen disajikan untuk membandingkan performa algoritma Insertion Sort, Merge Sort, dan Quick Sort pada berbagai karakteristik data, yaitu random, sorted, reversed, dan nearly sorted dengan ukuran data 1.000 hingga 10.000 elemen.

Pengukuran dilakukan berdasarkan waktu eksekusi, dan hasilnya disajikan dalam bentuk tabel serta grafik untuk mempermudah analisis dan perbandingan antar algoritma.

3.1 Pengujian pada Random Data

Random data merupakan dataset yang nilai-nilainya dihasilkan secara acak tanpa pola tertentu. Kondisi ini merepresentasikan situasi umum dalam berbagai kasus komputasi di dunia nyata.

Tabel 3.1 Hasil Pengujian Algoritma Sorting pada Random Data

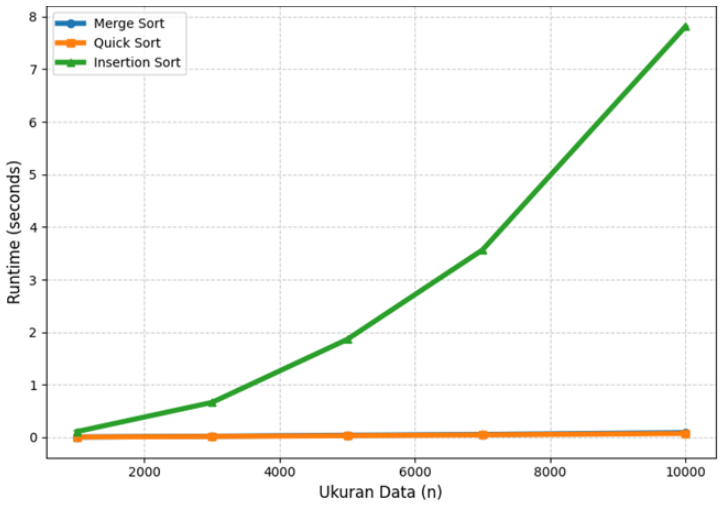
Ukuran Data	Insertion Sprt	Merge Sort	Quick Sort
1000	0.108316	0.006423	0.005464
3000	0.664020	0.021471	0.017330
5000	1.863044	0.040331	0.031799
7000	3.56957	0.054830	0.046765
10000	7.812050	0.090478	0.073230

Berdasarkan hasil pengujian yang ditunjukkan pada Tabel 4.1, terlihat bahwa waktu eksekusi algoritma meningkat seiring dengan bertambahnya ukuran data yang diproses. Namun peningkatan tersebut terjadi dengan pola yang berbeda pada setiap algoritma.

Algoritma Insertion Sort menunjukkan peningkatan waktu eksekusi yang sangat signifikan ketika ukuran data semakin besar. Hal ini disebabkan karena algoritma tersebut memiliki kompleksitas waktu rata-rata sebesar $O(n^2)$ sehingga jumlah perbandingan dan perpindahan elemen meningkat secara drastis ketika ukuran data bertambah.

Sebaliknya, algoritma Merge Sort dan Quick Sort menunjukkan performa yang jauh lebih stabil. Kedua algoritma ini memiliki kompleksitas waktu rata-rata sebesar $O(n \log n)$ sehingga pertumbuhan waktu eksekusinya tidak meningkat secara drastis ketika ukuran data bertambah.

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 3, No. 12, Tahun 2026
ISSN 3025-0919 (media online)
Hal 3049-3057



Gambar 3.1 Grafik Perbandingan Waktu Eksekusi pada Random Data

Berdasarkan Gambar 3.1 dapat dilihat bahwa garis performa Insertion Sort meningkat sangat tajam dibandingkan algoritma lainnya. Hal ini menunjukkan bahwa algoritma tersebut kurang efisien ketika digunakan untuk mengurutkan data acak dengan ukuran yang besar.

3.2 Pengujian pada Sorted Data

Sorted data merupakan dataset yang telah berada dalam kondisi terurut secara menaik. Pengujian ini bertujuan untuk menganalisis performa algoritma ketika data sudah berada dalam kondisi ideal.

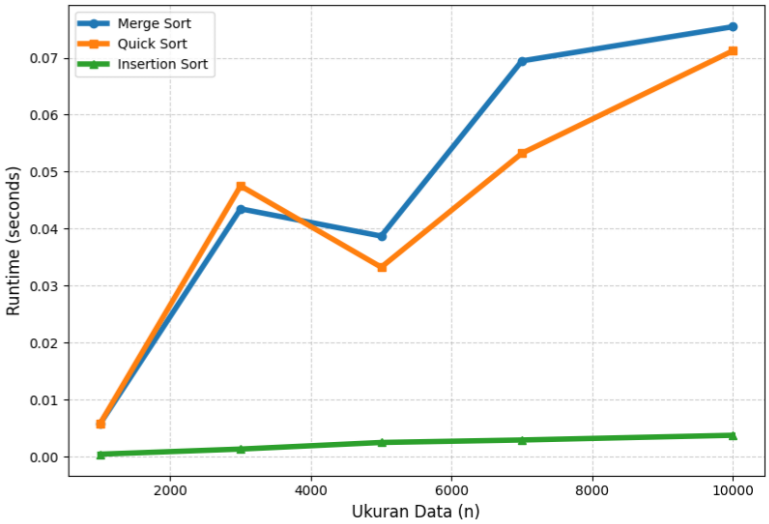
Tabel 3.2 Hasil Pengujian Algoritma Sorting pada Sorted Data

Ukuran Data	Insertion Sprt	Merge Sort	Quick Sort
1000	0.0004233	0.005718	0.005751
3000	0.001323	0.043483	0.047512
5000	0.002479	0.038705	0.033240
7000	0.002916	0.069447	0.053231
10000	0.003745	0.075448	0.071217

Hasil pengujian menunjukkan bahwa algoritma Insertion Sort memiliki performa yang sangat baik pada kondisi data yang telah terurut. Hal ini terjadi karena algoritma tersebut hanya perlu melakukan sedikit perbandingan untuk memastikan bahwa setiap elemen telah berada pada posisi yang benar.

Pada kondisi terbaiknya, Insertion Sort memiliki kompleksitas waktu O(n) sehingga waktu eksekusi yang dihasilkan menjadi sangat kecil meskipun ukuran data meningkat.

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 3, No. 12, Tahun 2026
ISSN 3025-0919 (media online)
Hal 3049-3057



Gambar 3.2 Grafik Perbandingan Performa Algoritma Sorting pada Sorted Data

Grafik pada Gambar 3.2 menunjukkan bahwa waktu eksekusi Insertion Sort tetap sangat kecil dibandingkan algoritma lainnya. Hal ini menunjukkan bahwa algoritma tersebut sangat efisien ketika digunakan pada data yang telah terurut.

3.3 Pengujian pada Reversed Data

Pengujian berikutnya dilakukan menggunakan data dengan kondisi reversed, yaitu data yang terurut secara terbalik dari nilai terbesar ke nilai terkecil.

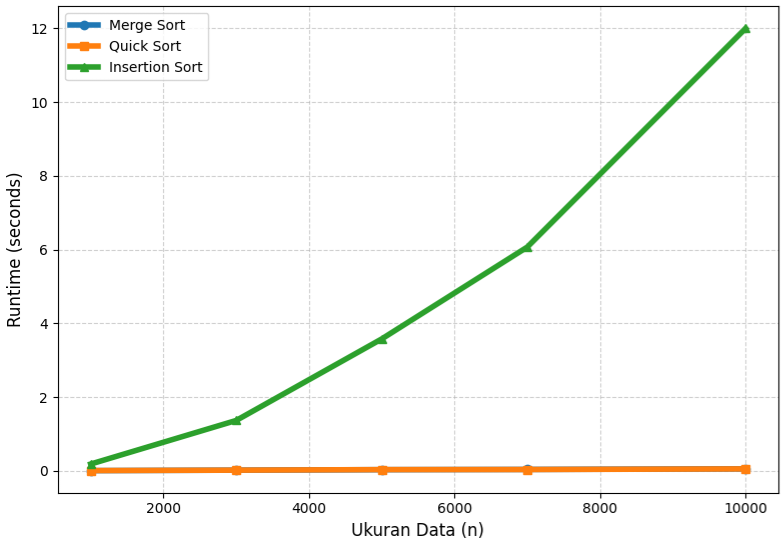
Tabel 3.3 Hasil Pengujian Algoritma Sorting pada Reversed Data

Ukuran Data	Insertion Sprt	Merge Sort	Quick Sort
1000	0.183634	0.005623	0.006534
3000	1.365674	0.017617	0.017938
5000	3.576832	0.031718	0.032788
7000	6.068029	0.037335	0.035335
10000	11.997608	0.054024	0.052535

Pada kondisi ini terlihat bahwa algoritma Insertion Sort mengalami penurunan performa yang sangat signifikan. Hal ini terjadi karena setiap elemen pada data harus dipindahkan ke posisi yang benar melalui proses perbandingan yang berulang.

Akibatnya, waktu eksekusi yang dibutuhkan menjadi sangat besar terutama ketika ukuran data meningkat.

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 3, No. 12, Tahun 2026
ISSN 3025-0919 (media online)
Hal 3049-3057



Gambar 3.3 Grafik Perbandingan Performa Algoritma Sorting pada Reversed Data

Grafik 3.3 menunjukkan bahwa waktu eksekusi Insertion Sort meningkat sangat tajam dibandingkan Merge Sort dan Quick Sort.

3.4 Pengujian pada Nearly Sorted Data

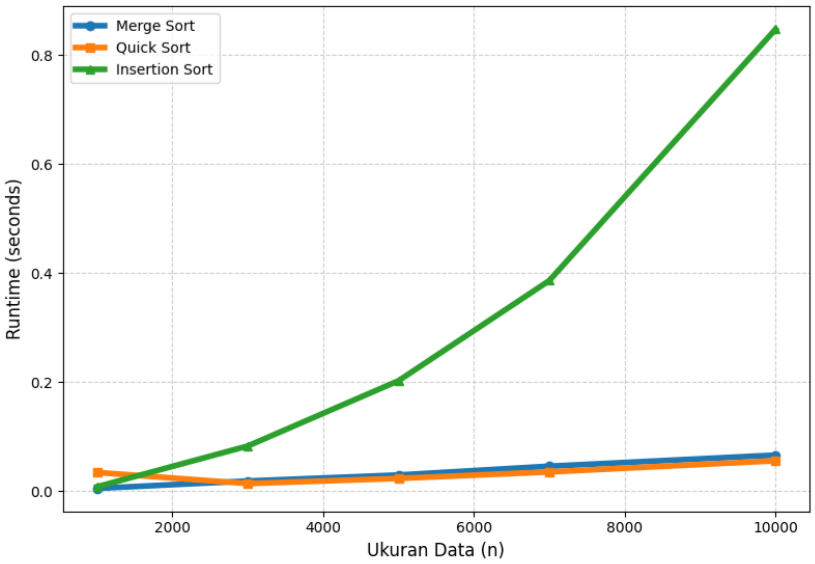
Pengujian terakhir dilakukan menggunakan data dengan kondisi nearly sorted, yaitu data yang hampir terurut namun masih terdapat beberapa elemen yang tidak berada pada posisi yang tepat.

Tabel 3.4 Hasil Pengujian Algoritma Sorting pada Nearly Sorted Data

Ukuran Data	Insertion Sprt	Merge Sort	Quick Sort
1000	0.007779	0.004843	0.034082
3000	0.082625	0.018393	0.013733
5000	0.202465	0.029428	0.023154
7000	0.386309	0.045521	0.035057
10000	0.847557	0.065906	0.055507

Hasil pengujian menunjukkan bahwa performa Insertion Sort pada kondisi nearly sorted lebih baik dibandingkan ketika digunakan pada random data atau reversed data.

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 3, No. 12, Tahun 2026
ISSN 3025-0919 (media online)
Hal 3049-3057



Gambar 3.4 Grafik Perbandingan Performa Algoritma Sorting pada Nearly Sorted Data

Berdasarkan grafik 3.4 Insertion Sort menunjukkan peningkatan waktu eksekusi yang paling signifikan seiring bertambahnya jumlah data. Sementara itu, Merge Sort dan Quick Sort memiliki performa yang lebih stabil dengan peningkatan yang relatif lebih kecil.

Selain itu, Quick Sort cenderung sedikit lebih cepat dibandingkan Merge Sort pada sebagian besar ukuran data. Hal ini menunjukkan bahwa algoritma dengan kompleksitas lebih efisien dibandingkan Insertion Sort untuk dataset berukuran besar.

3.5 Analisis Perbandingan Performa Algoritma

Berdasarkan seluruh hasil eksperimen yang telah dilakukan, dapat disimpulkan bahwa karakteristik data memiliki pengaruh yang signifikan terhadap performa algoritma sorting. Algoritma Insertion Sort menunjukkan performa yang sangat baik pada data yang telah terurut maupun hampir terurut, namun mengalami penurunan performa yang signifikan pada data acak dan data terbalik. Sementara itu, algoritma Merge Sort menunjukkan performa yang paling stabil pada seluruh jenis karakteristik data. Quick Sort juga memiliki performa yang relatif baik dan cepat pada berbagai kondisi, meskipun mengalami sedikit fluktuasi tergantung pada kondisi awal data. Hal ini disebabkan karena Merge Sort dan Quick Sort menggunakan pendekatan divide and conquer sehingga kinerjanya tidak terlalu dipengaruhi oleh urutan data dibandingkan dengan Insertion Sort. Dengan demikian, pemilihan algoritma sorting yang tepat sangat penting untuk meningkatkan efisiensi pemrosesan data, terutama pada dataset berukuran besar dan dengan karakteristik yang beragam.

4. KESIMPULAN

Berdasarkan penelitian, karakteristik data berpengaruh penting terhadap performa algoritma sorting. Pengujian pada *Insertion Sort, Merge Sort,* dan *Quick Sort* menunjukkan bahwa tiap algoritma memiliki keunggulan tergantung kondisi data.

Insertion Sort efektif untuk dataset kecil atau hamper terurut karena membutuhkan sedikit operasi, namun performanya menurun drastic pada data besar atau acak ($O(n^2)$). Sementara itu, *Merge Sort* dan *Quick Sort* lebih stabil dengan kompleksitas rata-rata O(n log n), sehingga lebih efisien untuk dataset besar. *Quick Sort* juga cenderung lebih cepat karena bekerja secara *in-place.*

Secara umum, pemilihan algoritma harus disesuaikan dengan karakteristik data: *Insertion Sort* untuk data kecil/hampir terurut, sedangkan *Merge Sort* dan *Quick Sort* untuk data besar atau acak.

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 3, No. 12, Tahun 2026
ISSN 3025-0919 (media online)
Hal 3049-3057

REFERENCES

Bakare, K. A., Okewu, A. A., Abiola, Z. A., Jaji, A., & Muhammed, A. (2024). A comparative study of sorting algorithms: Efficiency and performance in Nigerian data systems. FUDMA Journal of Sciences.

Bhargava, A. Y. (2024). Grokking algorithms (2nd ed.). Manning Publications.

Carter, J., et al. (2023). Fundamentals of algorithm complexity analysis. Journal of Computer Science Education, 18(2), 45–67.

Fahreza, A., & Suhartono. (2024). Quicksort and mergesort performance comparison in the Flutter framework. Journal of Informatics and Science Media.

Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2024). Data structures and algorithms in Python (2nd ed.). Wiley.

Mamatnabiyev, Z., & Zhaparov, M. (2024). Comparative analysis of sorting algorithms used in competitive programming. Journal of Emerging Technologies and Computing.

Wibowo, F. R., & Faisal, M. (2024). Comparative analysis of sorting algorithms: TimSort Python and classical sorting methods. JISA (Jurnal Informatika dan Sains).

Saputra, M. S. A., Meidiansyah, M. R., Sanputra. A. D., & Wardana, J. (2025). Comparative Analysis of Searching and Sorting Algorithms in Student Data Processing. International Journal of Education, Technology and Others (IJEIT). 8(3). 148-157.