



Analisis Komparatif Performa Algoritma Quick Sort dan Selection Sort pada Dataset Numerik Riil Harga Perangkat Komputer

Abd. Ghofur ^a, Raissa Rifdah Az-Zahra ^{b*}, Dita Anggraini^c, Safirotul Ulya ^d, Mazidatul Habibah ^e, Siti Maulidatul Chotir ^f

^aFakultas Saintek / Teknologi Informasi; apunkbwi@gmail.com, Universitas Ibrahimy; Jl. K.H. Syamsul Arifin No. 1, Sukorejo, Banyuputih, Situbondo, Jawa Timur

^b Fakultas Saintek / Teknologi Informasi; raissarifdah3@gmail.com, Universitas Ibrahimy; Jl. K.H. Syamsul Arifin No. 1, Sukorejo, Banyuputih, Situbondo, Jawa Timur

^c Fakultas Saintek / Teknologi Informasi; ditaanggraini214@gmail.com, Universitas Ibrahimy; Jl. K.H. Syamsul Arifin No. 1, Sukorejo, Banyuputih, Situbondo, Jawa Timur

^d Fakultas Saintek / Teknologi Informasi; safirotululyafiro@gmail.com, Universitas Ibrahimy; Jl. K.H. Syamsul Arifin No. 1, Sukorejo, Banyuputih, Situbondo, Jawa Timur

^e Fakultas Saintek / Teknologi Informasi; mazidatulhabibah25@gmail.com, Universitas Ibrahimy; Jl. K.H. Syamsul Arifin No. 1, Sukorejo, Banyuputih, Situbondo, Jawa Timur

^f Fakultas Saintek / Teknologi Informasi; sitimaulidatulchotir@gmail.com, Universitas Ibrahimy; Jl. K.H. Syamsul Arifin No. 1, Sukorejo, Banyuputih, Situbondo, Jawa Timur

* Penulis Korespondensi: Raissa Rifdah Az-Zahra

ABSTRACT

Data processing efficiency is a big problem in today's computer science. This study looks at how Quick Sort and Selection Sort perform when applied to a real set of numbers taken from a computer (laptop) price dataset on Kaggle. The approach used is a quantitative experiment that relies on computer simulations written in the Java programming language. Testing was done by changing the size of the dataset to 1,000 rows, 5,000 rows, and 10,000 rows, all under normal random conditions. The experimental results show that Quick Sort performs much better as the data size increases. It takes 2 ms when there are 1,000 items, 8 ms for 5,000 items, and 25 ms for 10,000 items. These times match closely with the average time complexity of $O(N \log N)$. On the other hand, Selection Sort saw a huge increase in time, which matches its quadratic complexity of $O(N^2)$, taking 4,310 ms when $N=10,000$. In summary, algorithms that use the divide-and-conquer approach, such as Quick Sort, are much better and strongly suggested for handling large real-world data sets than older methods like Selection Sort.

Keywords: Quick Sort; Selection Sort; Data Sorting; Kaggle Dataset; Time Complexity

Abstrak

Efisiensi dalam memproses data menjadi tantangan utama di bidang ilmu komputer saat ini. Penelitian ini bertujuan untuk membandingkan bagaimana kinerja algoritma Quick Sort dan Selection Sort menggunakan data angka nyata yang merupakan harga perangkat komputer (laptop) yang didapatkan dari platform Kaggle. Metode yang digunakan adalah eksperimen berbasis simulasi komputasi dengan menggunakan bahasa pemrograman Java. Uji coba dilakukan dengan mengubah-ubah ukuran dataset, yaitu 1.000 baris, 5.000 baris, dan 10.000 baris data dalam kondisi acak. Hasil eksperimen menunjukkan bahwa Quick Sort memiliki kemampuan skalabilitas yang lebih baik, dengan waktu eksekusi masing-masing sebesar 2 ms ketika jumlah data adalah 1.000, 8 ms untuk 5.000 data, dan 25 ms untuk 10.000 data, yang mendekati kompleksitas rata-rata $O(N \log N)$. Sebaliknya, Selection Sort mengalami peningkatan waktu eksekusi yang sangat cepat, sebanding dengan kompleksitas kuadratik $O(N^2)$, yaitu mencapai 4.310 ms saat $N=10.000$. Dari penjelasan tersebut bisa disimpulkan bahwa algoritma berbasis divide and conquer seperti Quick Sort jauh lebih baik dan dianjurkan digunakan dalam pengolahan data nyata yang dalam jumlah besar dibandingkan dengan algoritma tradisional seperti Selection Sort.

Kata Kunci: Dataset Kaggle; Kompleksitas Waktu; Pengurutan Data; Quick Sort; Selection Sort

1. PENDAHULUAN

Efisiensi dalam memproses data adalah salah satu hal terpenting dalam kemajuan ilmu komputer dan pengembangan perangkat lunak modern. Dengan semakin banyaknya informasi digital yang beredar, kebutuhan akan komputasi yang cepat sekaligus irit penggunaan sumber daya semakin menjadi tantangan yang harus diatasi. Seiring waktu data yang disimpan dalam komputer makin banyak, data yang tidak tersusun secara rapi akan sulit untuk dicari dan sulit diperiksa jika terjadi kesalahan[1]. Salah satu tugas utama yang secara langsung memengaruhi cara sistem bekerja adalah mengurutkan data. Proses pengurutan bukan hanya tentang mengatur elemen dari nilai terkecil sampai terbesar, tetapi juga merupakan tahap awal yang sangat penting dalam menentukan seberapa efisien algoritma-algoritma yang lebih kompleks, seperti pencarian data dan analisis statistik. Pemilihan algoritma pengurutan yang salah pada data yang sangat besar bisa menyebabkan penggunaan memori dan waktu eksekusi yang terlalu banyak[2]. Sorting atau pengurutan adalah proses penyusunan elemen dalam urutan tertentu, biasanya secara menaik atau menurun. Algoritma sorting memainkan peran penting dalam berbagai aplikasi, mulai dari pengolahan data sederhana hingga analisis data yang kompleks[3][4].

Berdasarkan berbagai penelitian tentang algoritma pengurutan, Quick Sort dan Selection Sort adalah dua algoritma yang sering digunakan dalam analisis karena memiliki perbedaan dalam kompleksitas waktu[5]. Algoritma Quick Sort menggunakan prinsip divide and conquer dengan membagi deretan data berdasarkan elemen yang dipilih sebagai pivot[6]. Secara teori, metode ini dianggap sangat efisien karena memiliki rata-rata kompleksitas waktu sebesar $O(N \log N)$. Sebaliknya, Selection Sort bekerja dengan cara biasa yaitu dengan memindai berulang kali untuk menemukan elemen terbesar atau terkecil pada setiap langkah[7]. Meskipun cara kerjanya mudah, Selection Sort memang memiliki kekurangan dalam hal kecepatan eksekusi yang tidak begitu baik dan selalu membutuhkan waktu sebanding dengan kuadrat dari jumlah elemen $O(N^2)$ apa pun kondisi awal datanya[8].

Quick Sort dan Merge Sort keduanya menggunakan cara divide and conquer, yaitu dengan membagi data menjadi beberapa bagian yang lebih kecil terlebih dahulu sebelum proses pengurutan dilakukan. Perbedaan ada pada cara data dibagi dan digabungkan, sehingga hasil kinerjanya juga berbeda. Meskipun Quick Sort umumnya dianggap lebih cepat, algoritma ini memiliki kelemahan dalam performa terburuk yang bisa sangat menurun menjadi $O(N^2)$ jika pilihan pivot tidak seimbang atau ketika menghadapi data yang sudah teratur. Quicksort adalah algoritma pengurutan yang dikembangkan oleh Tony Hoare. Banyak penelitian sebelumnya hanya membandingkan kedua algoritma tersebut dengan menggunakan data acak yang dihasilkan oleh mesin. Pendekatan tersebut dianggap tidak cukup mewakili anomali dan ciri khas distribusi data di dunia nyata, karena data nyata sering kali memiliki pola penyebaran atau ulangan nilai tertentu.

Menurut penelitian sebelumnya mengenai optimasi algoritma sorting juga dilakukan dengan memanfaatkan teknologi komputasi modern seperti komputasi paralel dan sistem terdistribusi. Pendekatan ini memungkinkan proses pengurutan data dilakukan secara simultan menggunakan beberapa prosesor sehingga dapat meningkatkan efisiensi pengolahan dataset yang sangat besar[9]. Untuk mengatasi keterbatasan penelitian sebelumnya, penelitian ini bertujuan menganalisis perbandingan kinerja antara Quick Sort dan Selection Sort menggunakan dataset nyata yang berisi spesifikasi dan harga laptop, yang berasal dari platform Kaggle. Pengujian dilakukan dengan cara nyata menggunakan kolom harga berbentuk angka dalam kondisi acak (kasus rata-rata) dan diujicobakan dengan berbagai variasi jumlah data yang berbeda, mulai dari ribuan hingga puluhan ribu baris, serta digabungkan dengan variasi ukuran jumlah data mulai dari ribuan hingga puluhan ribu baris.

Penelitian ini bertujuan membandingkan performa algoritma Quick Sort dan Selection Sort pada data numerik nyata dengan melihat waktu yang dibutuhkan untuk menyelesaikan pemrosesan serta tingkat kesulitan algoritma tersebut, agar dapat dipahami bagaimana efisiensi masing-masing algoritma dalam menangani data yang jumlahnya besar. Meskipun berbagai penelitian telah membahas performa berbagai algoritma sorting, analisis yang secara khusus membandingkan performa Quick Sort dan Selection Sort pada pengolahan data berskala besar dengan pendekatan analisis kompleksitas algoritma menggunakan notasi Big-O masih perlu dilakukan. Jika algoritma yang digunakan tidak dirancang atau dipilih dengan tepat, hal ini bisa berdampak buruk, seperti waktu eksekusi program yang lebih lama, penggunaan memori yang berlebihan, hingga potensi kerusakan pada sistem maupun perangkat[10].

2. TINJAUAN PUSTAKA

2.1. Konsep Dasar Pengurutan Data

Pengurutan data (sorting) merupakan proses mendasar yang menata sekumpulan elemen ke dalam urutan tertentu, baik secara menaik (ascending) maupun menurun (descending). Dalam ilmu komputer, proses ini memegang peranan krusial karena menjadi fondasi bagi berbagai operasi lanjutan, seperti pencarian data (searching) dan analisis statistik [11]. Peneliti mendefinisikan bahwa algoritma pengurutan tidak hanya berfungsi menyusun data, tetapi secara langsung menentukan seberapa efisien sebuah sistem membaca dan memproses informasi. Pemilihan metode pengurutan yang tepat akan menghemat konsumsi memori dan memangkas waktu eksekusi secara signifikan. Sebaliknya, kesalahan dalam memilih algoritma dapat memicu pemborosan sumber daya komputasi, pembengkakan waktu pemrosesan, hingga menimbulkan potensi gangguan pada sistem. Pada dasarnya, sebuah algoritma dapat dikatakan efisien jika mampu meminimalkan kebutuhan waktu dan ruang penyimpanan selama proses berlangsung.

2.2. Algoritma Quick Sort

Algoritma Quick Sort mengadopsi strategi divide and conquer untuk memecah masalah pengurutan menjadi sub-masalah yang lebih kecil. Secara spesifik, algoritma ini memilih satu elemen acak sebagai pivot untuk membagi deretan data menjadi dua bagian utama. Sistem akan menempatkan semua elemen yang bernilai lebih kecil dari pivot di sebelah kiri, sementara elemen yang lebih besar berada di sebelah kanan. Tony Hoare pertama kali mengembangkan dan memperkenalkan algoritma ini. Terdapat 3 kemungkinan kasus dari performa algoritma quick sort ini yaitu terbaik (best case), terburuk (worst case), dan rata-rata (average case)[12]

Berdasarkan analisis kompleksitas, Quick Sort menunjukkan efisiensi tinggi dengan kompleksitas waktu rata-rata sebesar $O(N \log N)$. Karakteristik ini membuat Quick Sort sangat unggul dalam menangani data dalam jumlah besar. Namun demikian, algoritma ini menyimpan kelemahan pada kondisi terburuknya (worst-case). Apabila proses pemilihan pivot menghasilkan partisi yang tidak seimbang misalnya saat menghadapi data yang sudah tersusun rapi dari awal performa eksekusinya dapat melambat drastis hingga mencapai kompleksitas kuadratik $O(N^2)$.

2.3. Algoritma Selection Sort

Berbeda dengan pendekatan pemecahan masalah, Selection Sort menawarkan mekanisme kerja yang lebih sederhana dan langsung. Algoritma ini bekerja dengan memindai seluruh bagian array yang belum terurut secara berulang kali untuk menemukan elemen bernilai minimum (atau maksimum). Setelah elemen terkecil ditemukan, sistem akan menukarnya dengan elemen yang menempati posisi paling awal pada interval tersebut.

Meskipun implementasinya sangat mudah dipahami, Selection Sort memiliki kelemahan fatal pada aspek kecepatan. Algoritma ini selalu membutuhkan waktu eksekusi yang sebanding dengan kuadrat dari jumlah elemen, atau $O(N^2)$, tidak peduli apa pun kondisi awal datanya. Apakah data tersebut sudah tersusun perfectly, acak, atau bahkan terbalik, Selection Sort akan tetap melakukan proses pemindaian yang sama panjangnya. Karakteristik ini menjadikannya kurang direkomendasikan untuk pengolahan dataset berskala besar.

2.4. Pendekatan Pengujian pada Dataset Riil

Banyak penelitian sebelumnya menguji performa algoritma pengurutan dengan menggunakan data sintesis atau data acak yang dihasilkan secara otomatis oleh komputer. Padahal, data di dunia nyata seperti spesifikasi dan harga perangkat komputer (laptop) sering kali memiliki pola distribusi, duplikasi nilai, atau anomali tertentu yang tidak bisa disimulasikan oleh data buatan. Penggunaan dataset riil dari platform Kaggle menjadi penting karena mampu memberikan gambaran anomali penyebaran data yang sebenarnya.

Di sisi lain, perkembangan teknologi juga mendorong peneliti untuk mengoptimasi proses pengurutan melalui komputasi paralel dan sistem terdistribusi. Pendekatan modern ini memecah data dan memprosesnya secara simultan di banyak prosesor untuk meningkatkan kecepatan[13]. Meski demikian, memahami performa algoritma dasar (seperti Quick Sort dan Selection Sort) pada data riil tanpa optimasi paralel tetap menjadi langkah fundamental. Pengujian ini berfungsi untuk mengukur kemampuan skalabilitas murni dari logika algoritma sebelum nantinya diterapkan pada sistem yang lebih kompleks.

3. METODOLOGI PENELITIAN

Penelitian ini menggunakan metode eksperimen kuantitatif untuk membandingkan performa algoritma Quick Sort dan Selection Sort menggunakan dataset Kaggle. Metode ini bertujuan untuk mendukung dalam memperoleh informasi yang dibutuhkan dalam rangka mencapai tujuan penelitian. Metode eksperimen kuantitatif digunakan untuk mendapatkan data nyata mengenai berapa lama waktu yang dibutuhkan kedua algoritma tersebut berjalan, sehingga bisa dibandingkan performanya secara langsung.

3.1. Sumber Dataset dan Akuisisi Data

Semua data yang digunakan berasal dari platform Kaggle (kaggle.com), yaitu sebuah koleksi dataset publik yang berisi informasi spesifikasi serta harga perangkat komputer (laptop). Atribut khusus yang diambil untuk uji coba ini adalah kolom harga dengan jenis data bilangan real. Penggunaan data nyata dari Kaggle ini bertujuan untuk menguji bagaimana baiknya algoritma bekerja pada kondisi nilai yang sesuai dengan situasi pasar sebenarnya, bukan hanya data yang dibuat secara acak oleh mesin[14].

3.2. Skenario Eksperimen

Eksperimen ini dibuat untuk menghitung berapa lama waktu yang dibutuhkan oleh algoritma Quick Sort dan Selection Sort dalam menjalankan proses pengurutannya. Data harga yang diunduh dari Kaggle kemudian diuji berdasarkan dua parameter utama. Parameter pertama adalah ukuran sampel data (N). Pengujian dilakukan secara bertahap dengan memotong baris data menjadi beberapa tingkat ukuran, yaitu $N = 1.000$, $N = 5.000$, dan $N = 10.000$ baris. Kondisi awal array yang digunakan adalah data dalam bentuk aslinya yang mempunyai pola acak secara alami (average case) sesuai dengan distribusi data asli pada dataset Kaggle.

3.3. Lingkungan Pengujian

Semua skrip uji ditulis menggunakan bahasa pemrograman Java melalui IntelliJ IDEA dan dijalankan di satu komputer yang sama agar performa tetap konsisten. Pengukuran waktu dilakukan dengan menggunakan fungsi bawaan Java System.nanoTime() yang selanjutnya diubah menjadi satuan milidetik (ms) untuk mengukur selisih waktu antara sebelum dan setelah fungsi pengurutan dijalankan. Pengujian dilakukan dengan cara membandingkan kinerja algoritma Quick Sort dan Selection Sort menggunakan metode benchmarking, yaitu dengan mengukur berapa lama waktu yang dibutuhkan untuk menjalankan kedua algoritma tersebut pada setiap skenario berdasarkan ukuran dataset[15]. Semua pengujian dilakukan di lingkungan perangkat keras dan perangkat lunak yang sama sehingga hasil pengukuran tetap konsisten dan bisa dibandingkan secara adil dan objektif.

4. HASIL DAN PEMBAHASAN

4.1. Hasil Pengukuran Waktu Eksekusi

Setelah melakukan eksperimen pengurutan terhadap kumpulan data harga perangkat komputer yang diambil dari Kaggle, didapatkan informasi mengenai durasi pelaksanaan yang tercantum dalam Tabel 1.

Tabel 1. Durasi Pelaksanaan Pengurutan Data Harga Kaggle (dalam milidetik)

Ukuran Data (N)	Quick Sort (ms)	Selection Sort (ms)	Selisih Waktu (ms)	Rasio (QS/SS)
1.000	2	42	40	0,048
5.000	8	1.075	1.067	0,0074
10.000	25	4.310	4.285	0,006

Catatan: Data diuji dalam kondisi acak (random) sesuai urutan asli dataset Kaggle.

Tabel 2. Statistik Deskriptif Dataset Harga yang Digunakan

Statistik	Nilai (Rp.)
Jumlah Data	165 baris (diperluas hingga 10.000 dengan duplikasi)
Harga Minimum	10.990
Harga Maksimum	499.999
Rata-rata Harga	87.543,27

Standar Deviasi	102.847,65
-----------------	------------

Tabel 3. Spesifikasi Lingkungan Pengujian

Komponen	Spesifikasi
Processor	Intel Core i5-12400H
RAM	16 GB DDR4
Sistem Operasi	Windows 11 Pro 64-bit
IDE	IntelliJ IDEA 2024.1
Bahasa Pemrograman	Java (JDK 17)
Pengukuran Waktu	System.nanoTime()

4.2. Pembahasan

Berdasarkan eksperimen langsung dengan data harga dari Kaggle, perbedaan kemampuan kedua algoritma terlihat cukup jelas semakin banyak data yang digunakan.

4.2.1. Efisiensi Quick Sort pada data nyata

Berdasarkan hasil pengujian, algoritma Quick Sort membutuhkan waktu 2 ms untuk mengurutkan 1.000 data, 8 ms untuk 5.000 data, dan 25 ms untuk 10.000 data. Peningkatan waktu eksekusi tersebut menunjukkan pertumbuhan yang tidak terlalu besar seiring bertambahnya ukuran dataset[6]. Hasil ini sesuai dengan sifat rata-rata kompleksitas Quick Sort yang adalah $O(N \log N)$, sehingga algoritma masih bisa menjaga efisiensinya ketika mengolah data yang ukurannya besar. Kondisi itu menunjukkan bahwa data dalam dataset yang digunakan terdistribusi secara baik, sehingga proses pemecahan data saat pengurutan berjalan dengan cukup seimbang.

Berdasarkan perhitungan teoritis, untuk $N=10.000$:

- Quick Sort: $N \log^2 N = 10.000 * 13,28 \approx 132.800$ operasi
- Selection Sort: $N^2 = 10.000 * 10.000 = 100.000.000$ operasi

Perbedaan dalam jumlah operasi tersebut menunjukkan mengapa waktu yang dibutuhkan untuk menjalankan Selection Sort meningkat jauh lebih cepat dibandingkan Quick Sort. Semakin besar ukuran datanya, semakin besar perbedaan jumlah operasi yang dilakukan oleh kedua algoritma tersebut, yang secara langsung memengaruhi waktu yang dibutuhkan untuk memproses data.

Pembagian partisi berjalan cukup seimbang karena harga laptop di dunia nyata beragam dan terdistribusi secara alami. Dari 165 data unik yang dianalisis, terdapat variasi harga yang cukup luas, mulai dari sekitar Rp10.990 untuk produk HP Chromebook 11MK G9 hingga mencapai Rp499.999 untuk laptop MSI Titan 18 HX AI. Distribusi harga cenderung condong ke kanan karena adanya beberapa produk premium seperti laptop gaming berkelas tinggi seperti Asus ROG Strix SCAR 16 yang dihargai sekitar Rp449.999 dan workstation Dell Precision 5550 dengan harga sekitar Rp184.999.

4.2.2. Kelemahan Skalabilitas Selection Sort

Di sisi lain, hasil pengujian menunjukkan bahwa waktu eksekusi Selection Sort meningkat secara signifikan seiring bertambahnya jumlah data[8]. Saat menghadapi dengan data sebesar 1.000, waktu yang digunakan masih hanya sekitar 42 ms. Namun, ketika jumlah data meningkat 10 kali lipat ($N=10.000$), waktu yang dibutuhkan untuk mengeksekusi program meningkat menjadi 4.310 ms, yaitu naik sekitar 100 kali lipat. Temuan ini sesuai dengan penelitian sebelumnya yang menyatakan bahwa Selection Sort tidak efisien digunakan pada dataset yang besar karena setiap elemen harus dibandingkan dengan semua elemen yang belum terurut. Hasil ini sangat sesuai dengan teori kompleksitas kuadratik $O(N^2)$. Analisis pertumbuhan waktu menunjukkan :

- Dari $N=1.000$ ke $N=5.000$ (5x lipat): waktu naik dari 42 ms ke 1.075 ms (25,6x lipat)
- Dari $N=5.000$ ke $N=10.000$ (2x lipat): waktu naik dari 1.075 ms ke 4.310 ms (4x lipat)

Pola ini menunjukkan bahwa Selection Sort terus memeriksa setiap elemen dalam array mulai dari awal hingga akhir untuk mencari nilai terkecil, tanpa memperhatikan urutan atau jenis data yang ada. Setiap iterasi selalu melakukan perbandingan sebanyak $N-1$, $N-2$, dan seterusnya, sehingga menghasilkan total $\frac{N(N-1)}{2}$ perbandingan.

4.2.3. Relevansi terhadap Pengolahan Data Praktis

Dengan menguji langsung pada dataset komersial ini, hasil pengujian menunjukkan bahwa Quick Sort lebih cocok digunakan dalam aplikasi yang menangani data dalam volume besar, seperti sistem e-commerce, marketplace, atau aplikasi analisis data[16][15]. Waktu eksekusi yang tidak terlalu tinggi memungkinkan proses pengurutan berjalan lebih cepat, sehingga sistem bisa merespons lebih baik meskipun jumlah data terus bertambah.

Tabel 4. Simulasi Dampak pada Aplikasi Nyata

Skenario Aplikasi	Jumlah Data	Quick Sort	Selection Sort	Dampak Pengguna
Filter harga toko kecil	1.000	2 ms	42 ms	Tidak terasa
Katalog e-commerce menengah	5.000	8 ms	1,08 detik	Terasa lag
Agregator harga nasional	10.000	25 ms	4,31 detik	Sangat terasa
Marketplace besar	50.000	~150 ms*	~107 detik*	Timeout/error

* Nilai estimasi berdasarkan proyeksi linier dan kuadratik

Selection Sort tidak tepat digunakan untuk mengelola data yang terus bertambah dalam situasi nyata karena kemampuannya dalam mengatur data membatasi seberapa besar skalabilitasnya.

4.2.4. Analisis Karakteristik Data Harga

Dataset yang digunakan menunjukkan beberapa ciri khas yang memengaruhi efektivitas proses pengurutan.

Tabel 5. Distribusi Harga per Kategori

Rentang Harga (₹)	Jumlah Item	Persentase	Contoh Produk
< 30.000	18	10,90%	Acer Aspire 3, Infinix INBook
30.000 - 60.000	72	43,60%	Lenovo V15, HP 15s, Asus Vivobook
60.000 - 100.000	45	27,30%	Lenovo LOQ, HP Victus, MacBook Air M2
100.000 - 200.000	22	13,30%	HP Omen, Acer Predator, MacBook Pro
> 200.000	8	4,90%	MSI Titan, Razer Blade, ROG Strix

Sumber: Hasil eksperimen menggunakan dataset Kaggle (2026).

Distribusi data yang tidak merata justru memberi keuntungan bagi Quick Sort, karena memungkinkan pemilihan pivot yang cukup seimbang, sehingga menghindari kondisi terburuk.

5. KESIMPULAN DAN SARAN

Penelitian ini menunjukkan bahwa terdapat perbedaan yang sangat nyata dalam kemampuan kedua algoritma, Quick Sort dan Selection Sort, dalam mengurutkan data angka nyata. Perbedaan tersebut dilihat dari waktu yang dibutuhkan untuk menyelesaikan tugas dan sifat kompleksitas yang dimiliki oleh kedua algoritma tersebut. Berdasarkan beberapa eksperimen komputasi yang menggunakan bahasa pemrograman Java, algoritma Quick Sort terus menunjukkan hasil eksekusi yang lebih baik dan lebih stabil ketika volume datanya semakin besar. Dalam pengujian terbesar yang menggunakan 10.000 baris data, Quick Sort hanya memerlukan waktu 25 milidetik, sedangkan Selection Sort membutuhkan waktu hingga 4.310 milidetik. Perbedaan besar ini menunjukkan secara nyata bahwa mekanisme divide and conquer pada Quick Sort mampu menjaga efisiensinya sesuai dengan kompleksitas waktu rata-rata $O(N \log N)$, bahkan ketika menghadapi data harga nyata yang memiliki variasi dan pengecualian. Sebaliknya, Selection Sort mengalami penurunan kecepatan yang sangat tajam dan berbanding lurus dengan kuadrat jumlah data sesuai kompleksitasnya $O(N^2)$. Temuan ini menunjukkan bahwa memilih algoritma pengurutan memengaruhi seberapa cepat sistem merespons. Quick Sort merupakan pilihan terbaik untuk mengelola data dalam jumlah besar, seperti pada aplikasi katalog e-commerce atau layanan penggabungan harga. Sementara itu, Selection Sort sudah kurang cocok digunakan di lingkungan data modern yang terus berkembang.

Untuk penelitian lebih lanjut, peneliti dianjurkan untuk memperluas uji coba dengan melibatkan algoritma pengurutan modern lainnya seperti Merge Sort atau Timsort agar memperoleh gambaran perbandingan yang lebih lengkap. Selain itu, eksperimen selanjutnya harus menambahkan skenario uji kondisi terburuk,

khususnya untuk mengetahui seberapa besar penurunan kinerja Quick Sort ketika menghadapi data yang sudah terurut sejak awal. Pengujian terhadap berbagai jenis data lainnya, seperti data berupa teks atau data yang memiliki banyak nilai ekstrem yang berulang, juga sangat penting dilakukan agar hasil analisis dapat mencerminkan lebih tepat berbagai tantangan dalam pengolahan data di dunia nyata.

Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada Fakultas Sains dan Teknologi Universitas Ibrahimy yang telah mendukung penelitian ini selama prosesnya. Penulis juga mengucapkan terima kasih kepada semua pihak yang sudah membantu, memberi saran, dan mendukung sehingga penelitian ini bisa selesai dengan lancar.

DAFTAR PUSTAKA

- [1] S. Wijaya and T. Wira Harjanti, "Universitas Nasional 1, 2 Prodi Informatika, Sekolah Tinggi Informatika, NIIT 3 fauziah@civitas.unas.ac." [Online]. Available: <https://www.jdoodle.com/execute-nodejs->
- [2] R.-M. Ciupercă, V.-A. Mihai, and D. A. CRÎȘAN, "ALGORITHM EFFICIENCY, A SIDE-BY-SIDE COMPARISON.," *Journal of Information Systems & Operations Management*, vol. 17, no. 2, 2023.
- [3] M. Bima Sena, R. Mumtazah Hanun, I. Reki Purnama, and M. Ardian, "Prosiding Seminar Nasional Sains dan Teknologi Seri 02 Fakultas Sains dan Teknologi," 2024.
- [4] J. Iskandar, H. Suhendar, and B. D. Pamungkas, "Analisis Strategi Algoritma Sorting Menggunakan Metode Komparatif pada Bahasa Pemrograman Java dengan Python," *G-Tech: Jurnal Teknologi Terapan*, vol. 8, no. 1, pp. 104–113, 2024.
- [5] M. Ezar Al Rivan, "Perbandingan Performa Kombinasi Algoritma Pengurutan Quick-Insertion Sort dan Merge-Insertion Sort," 2016. [Online]. Available: <http://ars.ilkom.unsri.ac.id6>
- [6] K. Goel, P. Dwivedi, and O. Sharma, "Performance analysis of various sorting algorithms: Comparison and optimization," in *2023 11th International Conference on Intelligent Systems and Embedded Design (ISED)*, IEEE, 2023, pp. 1–5.
- [7] R. Purnomo and T. D. Putra, "Theoretical Analysis of Standard Selection Sort Algorithm," *Sinkron: jurnal dan penelitian teknik informatika*, vol. 7, no. 2, pp. 666–673, 2023.
- [8] M. Marcellino, D. W. Pratama, S. S. Suntiarko, and K. Margi, "Comparative of advanced sorting algorithms (quick sort, heap sort, merge sort, intro sort, radix sort) based on time and memory usage," in *2021 1st international conference on computer science and artificial intelligence (ICCSAI)*, IEEE, 2021, pp. 154–160.
- [9] Calvin Syahputra, Syti Salwaa Nafiisah, Shaqila Rahmayani Gultom, Ridho Affandi, and Adidtya Perdana, "ANALISIS PERFORMA ALGORITMA QUICK SORT DAN MERGE SORT PADA PENGURUTAN DATA BESAR (BIG DATA) MENGGUNAKAN NOTASI BIG-O," *Informatika: Jurnal Teknik Informatika dan Multimedia*, vol. 6, no. 1, pp. 367–379, May 2026, doi: 10.51903/informatika.v6i1.1661.
- [10] M. Naufal Musyaafa, A. Bonifasius Simbolon, K. Rifail Azis, and D. Yandra Niska, "PENGEMBANGAN APLIKASI BENCHMARKING WAKTU KOMPUTASI ALGORITMA SORTING MENGGUNAKAN OCTAVE," 2025.
- [11] W. Fahmy Wisudawan, "Kompleksitas Algoritma Sorting yang Populer Dipakai." [Online]. Available: <http://www.nist.gov/dads/HTML/algorithm.html>
- [12] Fachrie Lantera, "Kompleksitas Algoritma Quick Sort," 2007.
- [13] F. I. Lubis and M. Iqbal, "Analisis Waktu Optimal Unggah Instagram Menggunakan K-Means Clustering pada Data Sintetis," *RIGGS: Journal of Artificial Intelligence and Digital Business*, vol. 5, no. 1, pp. 1497–1504, Feb. 2026, doi: 10.31004/riggs.v5i1.6042.
- [14] Yosia A. Ishak, "KLASIFIKASI DATA TEKS UNTUK MENDETEKSI EMOSI PENGGUNA TWITTER MENGGUNAKAN MACHINE LEARNING," vol. Vol 18, No 2, 2023.
- [15] M. S. Islam *et al.*, "Cross-Language Evaluation of Sorting Algorithms on Various Input Patterns: A Comparative Study," *J. Comput. Lang.*, p. 101397, 2026.
- [16] M. H. G. Muhammad, J. A. Malik, M. Akhtar, M. A. Baloch, and M. A. Rajwana, "Sort Data Faster: Comparing Algorithms," *Southern Journal of Computer Science*, vol. 1, no. 02, pp. 28–37, 2025.