Infoman's : Jurnal Ilmu-ilmu Informatika dan Manajemen Vol. 19 No.2 November (2025)
DOI: 10.33481/infomans.vxxixx

Algoritma Linear Search dan Binary Search Berdasarkan Ukuran dan Kondisi Keterurutan Data

[1]Gilang Abdul Rahman, [2]Dani Indra Junaedi, [3]Deris Santika
[1,2,3]Universitas Sebelas April
Jl. Angkrek Situ No.19, Sumedang, Jawa Barat 45323
email : [1]240160121039@student.unsap.ac.id, [2]dani@unsap.ac.id, [3]deris@unsap.ac.id

ABSTRACT
This study compares the performance of Linear Search and Binary Search in data retrieval under varying dataset sizes and ordering conditions. Rather than relying solely on theoretical complexity, Binary Search is evaluated end-to-end by including the required sorting step prior to searching. A quantitative experiment is conducted in Python 3.13.9 using shuffled unique integer arrays with sizes ranging from $n = 100$to $n = 1{,}000{,}000$. Four target scenarios are tested: target located at the beginning, middle, end, and target not found. The primary metrics are execution time and summary statistics (median and mean) computed from repeated runs for each scenario. The results indicate that for a single search on initially unsorted data, the Sorting+Binary approach tends to yield a higher total time than Linear Search because sorting dominates the overall cost, while the binary search component itself remains comparatively small. The contribution of this work is an end-to-end evaluation that accounts for sorting overhead and provides practical guidelines for selecting the appropriate search algorithm across dataset sizes and query scenarios. These findings highlight that algorithm selection should account for data characteristics and preprocessing overhead; Binary Search is most beneficial when data is already sorted or when sorting costs can be amortized across repeated queries.

Keywords **-** Binary search, Linear search, Sorting, Time complexity

1. Introduction

Pencarian data (searching) merupakan operasi fundamental pada sistem informasi, baik pada berkas, basis data, maupun struktur data di memori. Dua pendekatan umum adalah Linear Search yang memeriksa elemen satu per satu, dan Binary Search yang membagi ruang pencarian secara berulang pada data yang sudah terurut. Secara teoretis, Linear Search memiliki kompleksitas O(n) sedangkan Binary Search O(log n)[1-3]. Namun, pada data awal yang tidak terurut, Binary Search umumnya membutuhkan tahap sorting terlebih dahulu sehingga biaya totalnya menjadi kombinasi biaya sorting dan biaya pencarian[4]. Kompleksitas dan dampak biaya sorting terhadap kinerja pencarian end-to-end juga menjadi perhatian dalam kajian analisis kompleksitas algoritma sorting. Pada konteks pencarian sekali (one-shot query), biaya sorting ini dapat mendominasi, sedangkan pada skenario pencarian berulang pada dataset yang sama, biaya sorting dapat diamortisasi sehingga pendekatan berbasis Binary Search menjadi lebih menarik[5,6].

Sebagian studi komparatif hanya membandingkan waktu pencarian (search-only) atau melakukan pengujian pada data yang sudah terurut, sehingga kurang merepresentasikan kondisi nyata ketika data awalnya acak/tidak terurut dan Binary Search harus didahului sorting. Sejumlah penelitian juga membandingkan beberapa metode pencarian termasuk Linear, Binary, hingga Hash Search pada karakteristik data tertentu, yang menunjukkan bahwa performa dapat dipengaruhi oleh sifat data dan metode yang digunakan[7-9]. Selain itu, konfigurasi uji (ukuran data, posisi target, serta jumlah pengulangan) sering tidak dijelaskan secara rinci sehingga sulit direplikasi. Penelitian ini mengevaluasi kinerja end-to-end Linear Search pada data acak dibandingkan pendekatan "Sorting + Binary Search"

pada beberapa ukuran data dan beberapa kasus target, menggunakan implementasi Python yang dapat direplikasi serta menghasilkan dataset benchmark untuk dianalisis lebih lanjut.

Tujuan penelitian ini adalah: membandingkan waktu eksekusi Linear Search dan Sorting Binary pada variasi ukuran data dan skenario posisi target, menganalisis komponen biaya (sorting vs binary), dan menyusun rekomendasi pemilihan algoritma berdasarkan karakteristik data serta frekuensi pencarian. Kontribusi utama penelitian ini adalah pengukuran biaya total sorting+pencarian pada Binary Search untuk data awal tidak terurut, evaluasi hingga skala 10^6, serta estimasi titik impas jumlah pencarian agar biaya sorting dapat diamortisasi.

Berdasarkan tujuan tersebut, penelitian ini dilaksanakan melalui eksperimen kuantitatif berbasis benchmarking untuk mengukur dan membandingkan waktu eksekusi Linear Search serta pendekatan Binary Search setelah sorting secara end-to-end. Pengujian dilakukan pada variasi ukuran data dan beberapa skenario posisi target agar hasil yang diperoleh tidak bias terhadap satu kondisi tertentu. Untuk meningkatkan keandalan pengukuran, setiap skenario dieksekusi berulang dan diringkas menggunakan statistik deskriptif (median dan rata-rata). Rincian desain eksperimen, pembentukan dataset, prosedur pengukuran waktu, serta parameter pengujian dijelaskan pada bagian Research Method berikut.

2. Research Method

Penelitian ini menggunakan metode kuantitatif-eksperimental melalui benchmarking terkontrol untuk membandingkan performa Linear Search dan Binary Search dalam proses pencarian data. Eksperimen dilakukan pada perangkat laptop (Lenovo model 83DV) dengan sistem operasi Windows 11 64-bit, prosesor Intel Core i5-13450HX, RAM 12 GB. Implementasi dan pengukuran dijalankan menggunakan Python 3.13.9. Dataset dibangkitkan berupa array bilangan unik dari 0 sampai $n-1$yang kemudian diacak (tidak terurut). Ukuran data yang diuji adalah:

$$n \in \{100,\ 1000,\ 10000,\ 100000,\ 1000000\}$$

Skenario target yang diuji meliputi: target di awal, target di tengah, target di akhir, dan target tidak ditemukan. Linear Search dijalankan langsung pada data acak (tidak terurut). Untuk skenario Binary Search, data terlebih dahulu disorting menggunakan fungsi built-in Python `sorted()` (yang mengimplementasikan varian TimSort), kemudian dilakukan pencarian biner. Analisis kompleksitas algoritma sorting menjadi penting karena komponen ini dapat mendominasi biaya total pada pencarian end-to-end[10-13]**.** Setiap ukuran data n diuji pada empat kasus target: elemen di awal, di tengah, di akhir, dan target tidak ditemukan. Pengukuran waktu menggunakan `time.perf_counter_ns()` agar resolusi cukup tinggi untuk microbenchmark dan sejalan dengan praktik pengukuran performa pada lingkungan Python[14-17]. Waktu eksekusi dikumpulkan melalui pengulangan (repeats) yang menyesuaikan ukuran data. Setiap iterasi melakukan warm-up ringan, pemanggilan `gc.collect()` untuk mengurangi pengaruh garbage collector, kemudian mencatat durasi Linear Search, durasi sorting, durasi Binary Search, dan total durasi "Sorting + Binary Search"[18]. Statistik yang dilaporkan adalah median dan rata-rata (mean) untuk merangkum data serta mengurangi pengaruh outlier dan noise lingkungan[19]. Untuk skenario pencarian berulang pada dataset yang sama (sorting dilakukan sekali), titik impas jumlah query m diperkirakan dengan:

$$m \approx \frac{Tsort}{Tlinear - TbinaryTsort} \qquad (1)$$

(dengan syarat $T_{\text{linear}} > T_{\text{binary}}$). Nilai $m$digunakan sebagai indikator kapan "sort sekali lalu binary berulang" mulai lebih kompetitif dibanding melakukan Linear Search berulang pada data acak.

DOI: 10.33481/infomans.vxxixx

3. Result and Analysis

Secara umum, pada pencarian tunggal terhadap data yang awalnya tidak terurut, pendekatan Sorting+Binary menghasilkan waktu total lebih besar dibanding Linear Search karena waktu sorting mendominasi total biaya. Waktu binary search sendiri relatif kecil dibanding komponen sorting. Hasil pengujian worst-case (target tidak ditemukan).

Table 1. Worst-case (target tidak ditemukan), median waktu (ms).

n	Linear (ms)	Sorting (ms)	Binary (ms)	Total (ms)	Total/Linear	m (query)
100	0.011	0.017	0.004	0.021	1.88	2.36
1.000	0.075	0.144	0.006	0.150	2.01	2.08
10.000	0.680	1.667	0.007	1.674	2.46	2.48
100.000	7.591	17.733	0.008	17.740	2.34	2.34
1.000.000	143.786	336.768	0.013	336.780	2.34	2.34

Rasio Total/Linear lebih besar dari 1 untuk seluruh ukuran data, sehingga pada pencarian tunggal Sorting+Binary lebih lambat daripada Linear Search. Namun, titik impas m berada sekitar 2-3 pencarian pada dataset yang sama untuk skenario worst-case, yang mengindikasikan bahwa biaya sorting dapat diamortisasi relatif cepat bila query dilakukan berulang pada dataset yang sama.

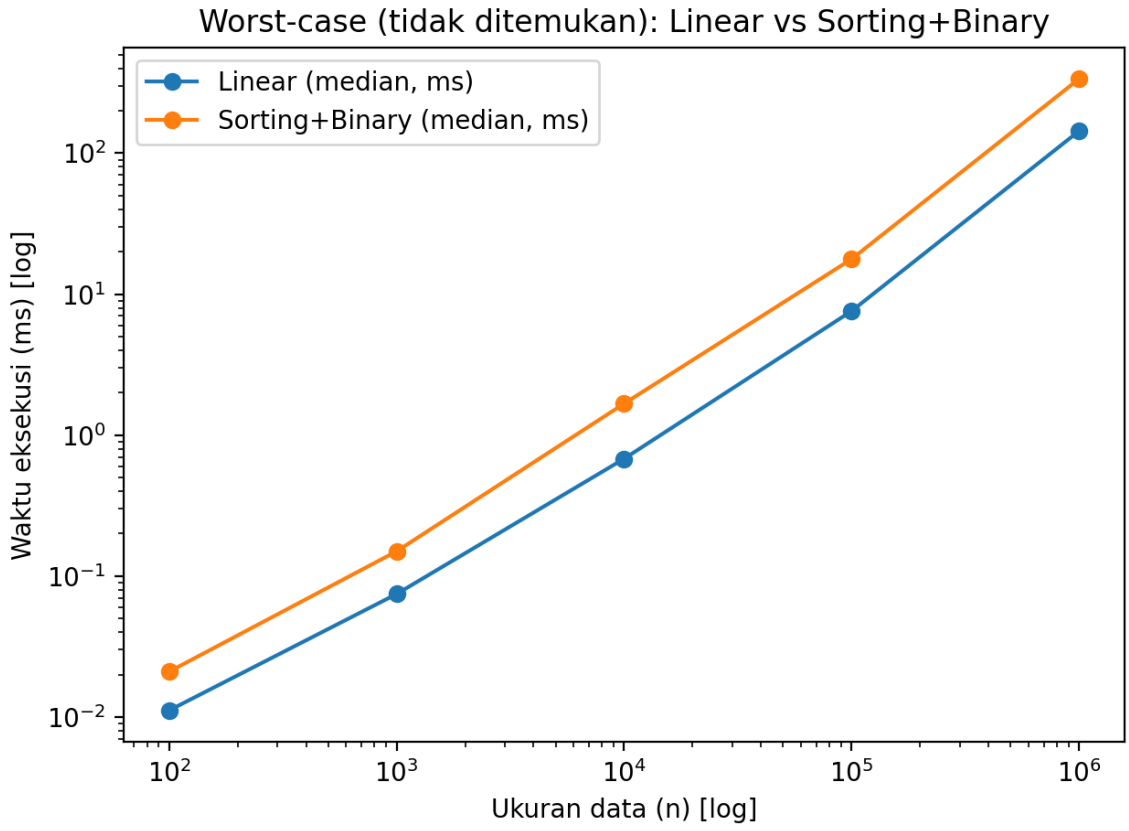

Figure 1. Worst-case (tidak ditemukan): Linear vs Sorting+Binary (median, skala log-log).

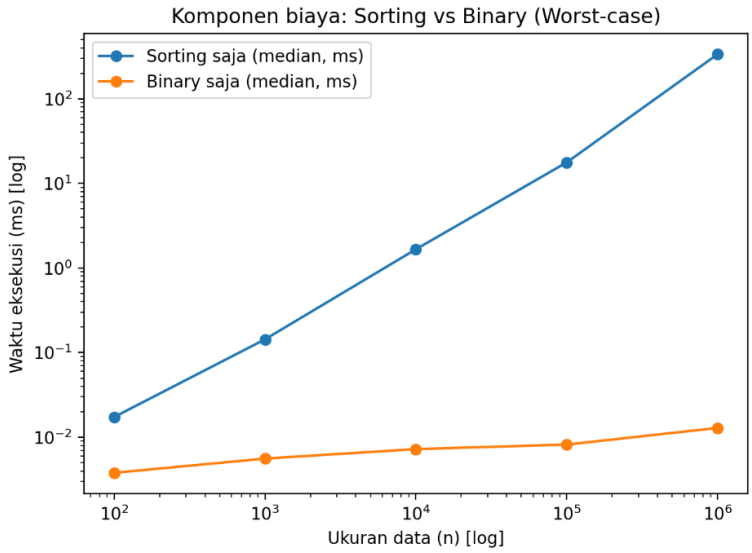

Figure 2. Komponen biaya pada worst-case: sorting vs binary search (median, skala log-log).

Hasil target di tengah (mendekati average-case)

Table 2. Target di tengah, median waktu (ms).

n	Linear (ms)	Sorting (ms)	Binary (ms)	Total (ms)	Total/Linear	m (query)
100	0.009	0.018	0.004	0.022	2.48	3.84
1.000	0.042	0.145	0.005	0.150	3.58	3.97
10.000	0.382	1.702	0.007	1.709	4.47	4.53
100.000	4.862	18.684	0.007	18.691	3.84	3.85
1.000.000	36.052	292.942	0.014	292.957	8.13	8.13

Untuk pencarian tunggal pada data acak, Sorting+Binary tetap lebih lambat dari Linear Search. Titik impas pada skenario target di tengah lebih besar (sekitar 4-8 pencarian), artinya sorting baru layak ketika pencarian dilakukan cukup sering pada dataset yang sama.

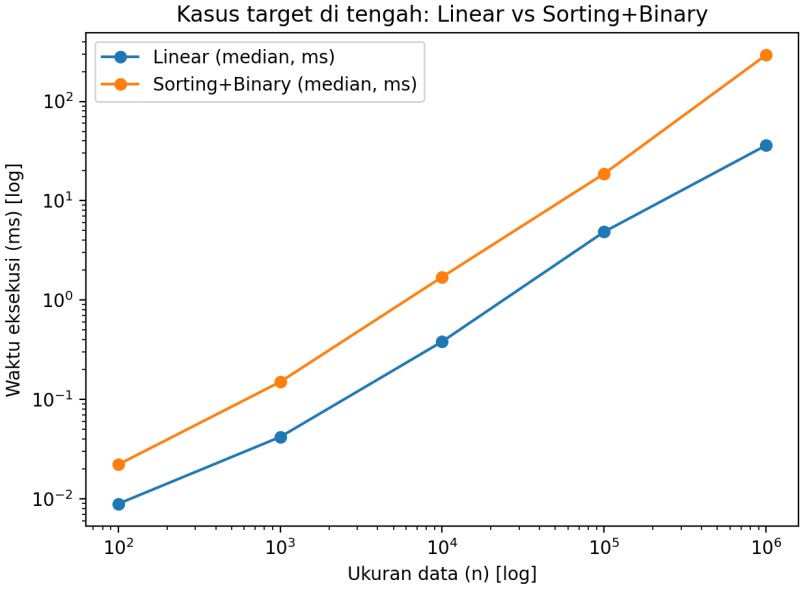

Figure 3. Target di tengah: Linear vs Sorting+Binary (median, skala log-log).

Hasil menunjukkan dua rekomendasi utama: jika data awal tidak terurut dan hanya membutuhkan pencarian sekali, Linear Search lebih efisien (tanpa overhead sorting); jika dataset relatif tetap dan pencarian dilakukan berulang, strategi "sort sekali lalu binary" menjadi masuk akal ketika jumlah query melebihi titik impas m (sekitar 2-3 pada worst-case dan 4-8 pada target tengah pada data uji ini).

4. Conclusion

Penelitian ini membandingkan Linear Search pada data tidak terurut dengan pendekatan Binary Search setelah sorting (biaya end-to-end = sorting + pencarian) pada variasi ukuran data n = 100 hingga n = 1.000.000 serta beberapa skenario posisi target. Hasil menunjukkan bahwa untuk pencarian tunggal pada data awal acak, Linear Search lebih efisien dibanding Sorting+Binary karena biaya sorting mendominasi waktu eksekusi, sedangkan biaya binary search sendiri relatif kecil. Namun, ketika dataset yang sama digunakan untuk pencarian berulang, biaya sorting dapat diamortisasi; berdasarkan perhitungan titik impas, pendekatan "sort sekali lalu binary" mulai kompetitif setelah sekitar 2-3 kali pencarian pada worst-case dan sekitar 4-8 kali pencarian pada skenario target di tengah. Dengan demikian, pemilihan algoritma pencarian sebaiknya mempertimbangkan keterurutan data, frekuensi query pada dataset yang sama, dan biaya pra-pemrosesan.

Kontribusi penelitian: menyajikan evaluasi kinerja end-to-end yang memasukkan biaya sorting pada Binary Search untuk data awal tidak terurut, menyediakan hasil empiris pada skala data hingga 10^6 dan beberapa skenario target, serta memberikan pedoman praktis pemilihan algoritma melalui estimasi titik impas jumlah pencarian pada dataset yang sama.

References

[1] A. Kurhade and N. N. Takawale, "Comparative Study of Searching Algorithm: Linear Search and Binary Search," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 13, no. 5, pp. 59–62, doi: 10.17148/IJARCCE.2024.13506.

[2] N. A. Zinnia and E. Hanada, "Optimizing Search Strategies: A Study of Two-Pointer Linear Search Implementation and Binary Search Approaches," 2024. [Online]. Available: https://arxiv.org/abs/2406.16729

[3] R. Zidan, K. N. U. Rehman, and I. Khan, "Experimental Performance Benchmarking of Popular Search Algorithms in Java and Python," *Int. J. Comput. Appl.*, vol. 187, no. 62, pp. 31–38, Dec. 2025, doi: 10.5120/ijca2025926016.

[4] A. S. Mohammed, Ş. E. Amrahov, and F. V Çelebi, "Interpolated binary search: An efficient hybrid search algorithm on ordered datasets," *Eng. Sci. Technol. an Int. J.*, vol. 24, no. 5, pp. 1072–1079, 2021, doi: https://doi.org/10.1016/j.jestch.2021.02.009.

[5] Wahana, A., Firmansyah, E., Al Rosyid, H. I., Fuadi, R. S., & Maylawati, D. S. A. (2021). Fuzzy Tahani Method in the Recommendation System for Selecting Mountain Tourism Destinations in West Java.

[6] Tamrin, M. A., Rizki, B., Nodas, A., Rahman, A., & Firmansyah, E. (2020). Perbandingan PenggunaanMetode Topsis dan Metode AHP dalam Penilaian Kinerja pada Karyawan (PT XYZ).Infoman's: JurnalIlmu-ilmu Informatika dan Manajemen,14(1).

[7] Firmansyah, E., Herdiana, D., & Yuniarto, D. (2020, October). Examining readiness of e-Learningimplementation using information system readiness impact model. In2020 8th International Conferenceon Cyber and IT Service Management (CITSM)(pp. 1-5). IEEE.

[8] Firmansyah, E., Rosmawati, R., Fuadi, R. S., Fauzy, D., & Ramdhani, M. A. (2019, December). Designof expert system to determine the proper diet using harmony search method. InJournal of Physics:Conference Series(Vol. 1402, No. 7, p. 077006). IOP Publishing.

[9] Firmansyah, E., & Helmiawan, M. A. (2025). Pengukuran Kesiapan Transformasi Digital Desa Kaduwulung Menuju Desa Cerdas Berbasis SNI ISO 37122: 2019 Melalui Pemetaan Data Desa.Infoman's: Jurnal Ilmu-ilmu Informatika dan Manajemen,19(1).

[10] Firmansyah, E., Herdiana, D., Yuniarto, D., & Junaedi, D. I. (2021, September). The K-Nearest Neighbor Algorithm for the Classification of Internet Users in Rural Campus. In 2021 9th International Conference on Cyber and IT Service Management (CITSM) (pp. 1-6). IEEE.

[11] Zaliluddin, D., Bastian, A., Yuliani, M. S. S., Firmansyah, E., & Sumaryana, Y. (2024). Engaging Teens in History through a Mobile Game Utilizing the Fisher-Yates Shuffle Algorithm and Honeycomb UX

Design. *International Journal of Interactive Mobile Technologies*, *18*(22).

[12] Syaripudin, U., Zaenal, R., Duri, M. F. A., Firmansyah, E., & Rahman, A. (2019, December). Comparison between Naïve Bayes and certainty factor to predict big five personality. In Journal of Physics: Conference Series (Vol. 1402, No. 7, p. 077030). IOP Publishing.

[13] Rahmayani, R., Firmansyah, E., & Hikmah, H. U. (2025). Inovasi Layanan Antar Jemput Paket Surat PTPos Indonesia Berdasarkan Penjualan dan Minat Beli JOVISHE: Journal of Visionary Sharia Economy,4(1), 33-47.

[14] Firmansyah, E., Rahman, A. B. A., & Subiyakto, A. A. (2023). Pengukuran Kesiapan Kota Cerdas Berdasarkan SNI ISO 37122: 2019.Infoman's: Jurnal Ilmu-ilmu Informatika dan Manajemen,17(2).

[15] Zulfikar,W.B., Irfan, M., Ghufron, M., Jumadi, J., & Firmansyah, E. (2020). Marketplace affiliatespotential analysis using cosine similarity and vision-based page segmentation.Bulletin of ElectricalEngineering and Informatics,9(6), 2492-2498.

[16] Ramadhan, N. D., Fhatturohmah, S., Ramadhani, S., & Firmansyah, E. (2023). Analysis of Digital WalletUsage on Consumptive Lifestyle.Journal of Islamic Economics and Business,3(2), 118-136.

[17] B. M. Al-aydi and S. S. Abu-naser, "Comparative Study of Traditional and AI-Enhanced Sorting Algorithms : Comparative Study of Traditional and AI-Enhanced Sorting Algorithms : QuickSort , MergeSort , HeapSort , and TimSort," vol. 9, no. September, pp. 70–79, 2025.

[18] M. S. Meidiansyah, M. Sanputra, and A. Wardana, "Comparative Analysis of Searching and Sorting Algorithms in Student Data Processing," *Int. J. Educ. Inf. Technol. Others*, vol. 2025, no. 3, pp. 148–157, 2025, [Online]. Available: https://jurnal.peneliti.net/index.php/IJEIT/article/view/12718

[19] Python Software Foundation, "time — Time access and conversions — Python 3.14.2 documentation." [Online]. Available: https://docs.python.org/3/library/time.html