JMI, 13 (2) (2026) pp. 56-65

Published by: Institute of Computer Science (IOCS)

JMI (Jurnal Manajemen Informatika) Medicom

Journal homepage: www. journals.joninstitute.org/index.php/jmi

Perbandingan Kinerja Linear Search dan Binary Search pada Pencarian Data Mahasiswa Berdasarkan Kompleksitas Teoretis dan Pengujian Empiris

Fillaah Al Farizi[1], Amarudin[2], Rohmat Indra Borman[3]

[1,2,3]Magister Ilmu Komputer, Fakultas Teknik dan Ilmu Komputer, Universitas Teknokrat Indonesia, Bandar Lampung, Indonesia.

Email: [1]fillaah_al_farizi@teknokrat.ac.id , [2]amarudin@teknokrat.ac.id, [3]rohmat_indra_borman@teknokrat.ac.id

A R T I C L E I N F O

Article history:

Received Jun 9, 2026
Revised Nov 23, 2026
Accepted Jul 1, 2026

Keywords:

Linear Search
Binary Search
Analisis Algoritma
Kompleksitas Waktu
Kinerja Empiris
Sistem Informasi Akademik

ABSTRACT

Pencarian data merupakan salah satu operasi fundamental yang sering digunakan dalam sistem informasi akademik untuk mengakses informasi mahasiswa secara cepat dan akurat. Seiring meningkatnya jumlah data yang dikelola oleh perguruan tinggi, pemilihan algoritma pencarian yang efisien menjadi faktor penting dalam menjaga kinerja sistem. Penelitian ini bertujuan untuk menganalisis dan membandingkan performa algoritma Linear Search dan Binary Search pada pencarian data mahasiswa berdasarkan pendekatan kompleksitas teoretis dan pengujian empiris. Metode penelitian yang digunakan adalah metode eksperimen dengan mengimplementasikan kedua algoritma menggunakan bahasa pemrograman Python pada dataset mahasiswa simulasi yang terdiri dari 100, 1.000, 10.000, dan 100.000 data. Parameter yang dianalisis meliputi kompleksitas waktu, jumlah operasi pencarian, dan waktu eksekusi yang diukur dalam satuan milidetik. Secara teoretis, Linear Search memiliki kompleksitas waktu O(n), sedangkan Binary Search memiliki kompleksitas waktu O(log n), sehingga Binary Search diperkirakan memiliki performa yang lebih baik pada dataset berukuran besar. Hasil penelitian yang diharapkan menunjukkan bahwa peningkatan ukuran dataset berpengaruh signifikan terhadap waktu eksekusi kedua algoritma, dengan Linear Search mengalami peningkatan waktu pencarian secara linier, sementara Binary Search menunjukkan peningkatan yang relatif lebih rendah. Selain memberikan bukti empiris terhadap teori kompleksitas algoritma, penelitian ini diharapkan dapat menjadi referensi dalam pemilihan algoritma pencarian yang tepat untuk sistem informasi akademik serta mendukung pengembangan aplikasi yang membutuhkan proses pencarian data secara efisien pada lingkungan perguruan tinggi.

Corresponding Author:

Fillaah Al Farizi,
Fakultas Teknik dan Ilmu Komputer
Universitas Teknokrat Indonesia
Jalan Zainal Abidin Pagar Alam No. 9-11, Labuhan Ratu, Kota Bandar Lampung, Provinsi Lampung.
Email: Fillaah_al_farizi@teknokrat.ac.id

1. PENDAHULUAN

Perkembangan teknologi informasi telah mendorong transformasi digital pada berbagai sektor, termasuk sektor pendidikan tinggi. Perguruan tinggi saat ini mengelola berbagai jenis data akademik dalam jumlah besar melalui sistem informasi akademik yang terintegrasi. Data tersebut meliputi data mahasiswa, jadwal perkuliahan, nilai akademik, riwayat registrasi, hingga data administrasi lainnya. Dalam pengoperasiannya, salah satu aktivitas yang paling sering dilakukan adalah proses pencarian

data. Efisiensi proses pencarian menjadi faktor penting karena secara langsung mempengaruhi kecepatan akses informasi, kualitas layanan akademik, serta kinerja sistem secara keseluruhan.(Sundaramoorthy & Karunanidhi, 2025)

Pencarian data merupakan operasi dasar dalam ilmu komputer yang bertujuan menemukan elemen tertentu dari sekumpulan data. Seiring bertambahnya jumlah data yang tersimpan dalam basis data maupun struktur data lainnya, kebutuhan terhadap algoritma pencarian yang efisien menjadi semakin penting. Algoritma yang kurang efisien dapat menyebabkan peningkatan waktu respon sistem, terutama pada aplikasi yang melakukan pencarian secara berulang dan dalam volume data yang besar. Oleh karena itu, pemilihan algoritma pencarian yang tepat merupakan salah satu aspek penting dalam perancangan sistem informasi modern.

Linear Search dan Binary Search merupakan dua algoritma pencarian yang paling banyak digunakan dalam pembelajaran maupun implementasi sistem. Linear Search bekerja dengan memeriksa setiap elemen data secara berurutan hingga elemen yang dicari ditemukan. Keunggulan algoritma ini adalah kemudahan implementasi dan kemampuannya bekerja pada data yang tidak terurut. Namun, Linear Search memiliki kompleksitas waktu sebesar O(n), sehingga jumlah operasi pencarian akan meningkat secara linear seiring bertambahnya ukuran data. Kondisi ini menyebabkan performa algoritma cenderung menurun ketika digunakan pada dataset berukuran besar.(Yasmin et al., 2025)

Sebaliknya, Binary Search menggunakan pendekatan divide and conquer dengan membagi ruang pencarian menjadi dua bagian secara berulang hingga data ditemukan. Pendekatan tersebut memungkinkan Binary Search memiliki kompleksitas waktu O(log n), yang secara teoritis jauh lebih efisien dibandingkan Linear Search. Akan tetapi, Binary Search memiliki syarat bahwa data harus berada dalam kondisi terurut sebelum proses pencarian dilakukan. Oleh karena itu, efektivitas Binary Search dalam implementasi nyata perlu dianalisis lebih lanjut untuk mengetahui sejauh mana keunggulannya dibandingkan Linear Search pada berbagai ukuran dataset.

Beberapa penelitian sebelumnya telah membahas perbandingan algoritma pencarian. (Sari & Rosadi, 2017) menunjukkan bahwa Binary Search memiliki kompleksitas yang lebih rendah dibandingkan Linear Search pada dataset numerik. (Leana et al., 2025) menemukan bahwa Binary Search memberikan performa yang lebih baik pada data terurut, meskipun memerlukan proses pengurutan tambahan(Purnama, 2025), membandingkan Linear Search, Binary Search, dan Hash Search pada dataset besar dan menyimpulkan bahwa Binary Search lebih efisien dibandingkan Linear Search. Penelitian lainnya yang dilakukan oleh (Nurvita & Putri, 2025) juga menunjukkan bahwa Binary Search memiliki keunggulan dari sisi waktu eksekusi pada dataset berukuran besar.

Meskipun demikian, sebagian besar penelitian terdahulu menggunakan dataset numerik umum, dataset ulasan produk, maupun data simulasi yang tidak secara spesifik merepresentasikan kebutuhan sistem informasi akademik. Selain itu, beberapa penelitian hanya berfokus pada analisis kompleksitas teoretis tanpa menghubungkannya dengan hasil pengujian empiris pada berbagai ukuran dataset. Kondisi tersebut menunjukkan adanya kesenjangan penelitian (*research gap)* yang perlu dikaji lebih lanjut, khususnya terkait implementasi algoritma pencarian pada data mahasiswa yang menjadi komponen utama dalam sistem informasi akademik.

Berdasarkan permasalahan tersebut, penelitian ini bertujuan untuk menganalisis dan membandingkan performa algoritma Linear Search dan Binary Search pada pencarian data mahasiswa berdasarkan pendekatan kompleksitas teoretis dan pengujian empiris. Pengujian dilakukan menggunakan dataset mahasiswa simulasi dengan ukuran 100, 1.000, 10.000, dan 100.000 data. Parameter yang dianalisis meliputi waktu eksekusi pencarian, jumlah operasi pencarian, dan karakteristik kompleksitas algoritma.

Kontribusi utama penelitian ini adalah memberikan evaluasi komprehensif mengenai efektivitas Linear Search dan Binary Search dalam konteks sistem informasi akademik. Selain memberikan validasi empiris terhadap teori kompleksitas algoritma, penelitian ini diharapkan dapat menjadi referensi bagi pengembang sistem informasi akademik dalam memilih algoritma pencarian yang sesuai berdasarkan karakteristik dan ukuran data yang dikelola.

2. METODE

Penelitian ini menggunakan metode eksperimen (*experimental research*) dengan pendekatan kuantitatif untuk membandingkan kinerja algoritma Linear Search dan Binary Search pada proses pencarian data mahasiswa.Metode eksperimen dipilih karena penelitian berfokus pada implementasi

langsung kedua algoritma dan pengukuran performanya berdasarkan parameter yang telah ditentukan.Pendekatan kuantitatif digunakan karena hasil penelitian berupa data numrik yang dapat diukur dan dianalisis secara objektif, seperti waktu eksekusi dan jumlah operasi pencarian.

2.1 Desain Penelitian

Penelitian dilakukan melalui serangkaian tahapan yang meliputi identifikasi masalah, studi literatur, penyusunan dataset, implementasi algoritma, pengujian kinerja, analisis hasil, dan penyusunan laporan penelitian. Desain penelitian ini bertujuan untuk memperoleh gambaran yang komprehensif mengenai perbedaan performa antara Linear Search dan Binary Search pada berbagai ukuran dataset.

Proses penelitian dimulai dengan mengidentifikasi permasalahan pencarian data yang sering terjadi pada sistem informasi akademik. Selanjutnya dilakukan kajian literatur untuk memperoleh dasar teoritis mengenai algoritma pencarian dan analisis kompleksitas. Setelah itu, dibuat dataset mahasiswa simulasi yang digunakan sebagai objek pengujian. Kedua algoritma kemudian diimplementasikan menggunakan bahasa pemrograman Python dan diuji pada dataset dengan ukuran yang berbeda. Hasil pengujian dianalisis untuk mengetahui hubungan antara kompleksitas teoretis dan kinerja empiris yang diperoleh. (Paleyes et al., 2022)

2.2 Data Set Penelitian

Dataset yang digunakan dalam penelitian ini berupa data mahasiswa simulasi yang dirancang untuk merepresentasikan data pada sistem informasi akademik perguruan tinggi. Dataset terdiri dari beberapa atribut utama yang umum digunakan dalam pengelolaan data mahasiswa, yaitu Nomor Induk Mahasiswa (NIM), nama mahasiswa, program studi, dan angkatan**.**

Tabel 1. Struktur Dataset Penelitian

No	Atribut	Tipe Data
1	NPM	Integer
2	Nama	String
3	Program Studi	String
4	Angkatan	Integer

Untuk menganalisis pengaruh ukuran data terhadap performa algoritma, dataset dibuat dalam empat skala berbeda sebagaimana ditunjukkan pada Tabel 2.

Tabel 2. Ukuran Dataset Pengujian

No	Jumlah Data
1	100
2	1.000
3	10.000
4	100.000

Penggunaan beberapa ukuran dataset bertujuan untuk mengamati perubahan waktu eksekusi algoritma ketika jumlah data meningkat secara signifikan.

2.3 Implementasi Algoritma

Penelitian ini mengimplementasikan dua algoritma pencarian, yaitu Linear Search dan Binary Search.

A. Linear Search

Linear Search merupakan algoritma pencarian yang bekerja dengan memeriksa setiap elemen data secara berurutan mulai dari elemen pertama hingga elemen terakhir sampai data yang dicari ditemukan atau seluruh elemen telah diperiksa.

Tahapan proses Linear Search adalah sebagai berikut:

1. Membaca elemen pertama dalam dataset.
2. Membandingkan elemen tersebut dengan data target.
3. Jika data sesuai, pencarian dihentikan.
4. Jika data tidak sesuai, pencarian dilanjutkan ke elemen berikutnya.
5. Proses diulang hingga data ditemukan atau seluruh elemen selesai diperiksa.

Secara teoritis, Linear Search memiliki kompleksitas waktu O(n), yang menunjukkan bahwa jumlah operasi pencarian meningkat secara linear terhadap jumlah data yang diproses.

B. Binary Search

Binary Search merupakan algoritma pencarian yang menggunakan pendekatan divide and conquer dengan membagi ruang pencarian menjadi dua bagian pada setiap iterasi.

Tahapan proses Binary Search adalah sebagai berikut:

1. Menentukan elemen tengah dataset.
2. Membandingkan nilai tengah dengan data target.
3. Jika nilai tengah sama dengan target, pencarian dihentikan.
4. Jika target lebih besar, pencarian dilakukan pada bagian kanan dataset.
5. Jika target lebih kecil, pencarian dilakukan pada bagian kiri dataset.
6. Langkah tersebut diulang hingga data ditemukan atau ruang pencarian habis.

Karena Binary Search membutuhkan data yang terurut, seluruh dataset diurutkan terlebih dahulu berdasarkan NIM sebelum proses pencarian dilakukan.

Secara teoritis, Binary Search memiliki kompleksitas waktu O(log n), sehingga jumlah operasi pencarian bertambah secara logaritmik terhadap ukuran dataset(Surachman et al., 2025).

2.4 Skenario Pengujian

Pengujian dilakukan menggunakan komputer yang sama untuk memastikan konsistensi hasil. Setiap algoritma diuji pada seluruh ukuran dataset yang telah ditentukan. Target pencarian dipilih dari data yang berada pada posisi akhir dataset untuk merepresentasikan kondisi pencarian terburuk (*worst case*).

Untuk meningkatkan validitas hasil, setiap pengujian dilakukan sebanyak beberapa kali dan hasil waktu eksekusi dirata-ratakan. Waktu eksekusi diukur menggunakan fungsi *time.perf_counter()* yang tersedia pada Python karena memiliki tingkat presisi tinggi dalam pengukuran waktu.(Prof.Mrs. Tejaswini.A. Puranik, 2025)

2.5 Parameter Pengujian

Penelitian ini menggunakan dua parameter utama dalam proses evaluasi algoritma, yaitu:

1. Waktu Eksekusi (*Execution Time*)

Waktu eksekusi digunakan untuk mengukur lama waktu yang dibutuhkan algoritma dalam menemukan data target. Pengukuran dilakukan dalam satuan milidetik (ms).

2. Kompleksitas Algoritma

Kompleksitas algoritma digunakan untuk menganalisis jumlah operasi yang diperlukan selama proses pencarian. Analisis dilakukan berdasarkan notasi Big O sehingga dapat dibandingkan dengan hasil pengujian empiris.

Linear Search memiliki kompleksitas O(n), sedangkan Binary Search memiliki kompleksitas O(log n).

2.6 Teknik Analisis Data

Data hasil pengujian dianalisis menggunakan pendekatan deskriptif dan komparatif. Analisis deskriptif dilakukan untuk menyajikan hasil pengukuran dalam bentuk tabel dan grafik. Selanjutnya dilakukan analisis komparatif untuk membandingkan performa kedua algoritma berdasarkan waktu eksekusi pada setiap ukuran dataset.

Hasil pengujian kemudian dibandingkan dengan teori kompleksitas algoritma untuk mengetahui kesesuaian antara analisis matematis dan implementasi nyata. Melalui pendekatan ini dapat diketahui algoritma yang memiliki tingkat efisiensi terbaik untuk diterapkan pada sistem informasi akademik dengan jumlah data yang berbeda.

2.7 Alur Penelitian

Alur penelitian dapat dijelaskan sebagai berikut:

Identifikasi Masalah:
→ Studi Literatur
→ Penyusunan Dataset
→ Implementasi Linear Search
→ Implementasi Binary Search
→ Pengujian Kinerja

→ Analisis Hasil
→ Kesimpulan
Alur tersebut menunjukkan bahwa penelitian dilakukan secara sistematis mulai dari identifikasi permasalahan hingga diperoleh kesimpulan mengenai algoritma pencarian yang paling efisien untuk digunakan pada pencarian data mahasiswa.

3. HASIL DAN PEMBAHASAN

3.1 Implementasi Algoritma Pencarian

Penelitian ini mengimplementasikan dua algoritma pencarian, yaitu Linear Search dan Binary Search, menggunakan bahasa pemrograman Python(Zidan et al., 2025). Kedua algoritma diterapkan pada dataset mahasiswa simulasi yang terdiri dari atribut NIM, nama mahasiswa, program studi, dan angkatan. Proses pencarian dilakukan berdasarkan atribut NIM karena setiap mahasiswa memiliki nomor identitas yang unik sehingga dapat digunakan sebagai target pencarian yang representatif dalam sistem informasi akademik.
A. Membuat Algoritma Linier Search:

```
def linear_search(data,target):

    for i in range(len(data)):

        if data[i] == target:
            return i

    return -1
```

Linear Search diimplementasikan dengan melakukan pemeriksaan elemen data secara berurutan mulai dari indeks pertama hingga data target ditemukan. Sebaliknya(Nurvita & Putri, 2025), Binary Search diimplementasikan dengan memanfaatkan metode divide and conquer yang membagi ruang pencarian menjadi dua bagian pada setiap iterasi. Sebelum dilakukan pencarian menggunakan.
B. Membuat Algoritma Binary Search:

```
def binary_search(data,target):

    low = 0
    high = len(data)-1

    while low <= high:

        mid = (low+high)//2

        if data[mid] == target:
            return mid

        elif data[mid] < target:

            low = mid+1

        else:

            high = mid-1

    return -1
```

Binary Search, dataset diurutkan terlebih dahulu berdasarkan NIM untuk memenuhi syarat penggunaan algoritma tersebut (Adelia et al., 2026).
Implementasi kedua algoritma dilakukan dalam lingkungan Python menggunakan fungsi pengukuran waktu berpresisi tinggi sehingga hasil pengujian dapat menggambarkan performa algoritma secara lebih akurat.

3.2 Hasil Pengujian Waktu Eksekusi

Pengujian dilakukan pada empat ukuran dataset yang berbeda, yaitu 100, 1.000, 10.000, dan 100.000 data mahasiswa. Tujuan pengujian ini adalah untuk mengetahui pengaruh ukuran dataset terhadap performa masing-masing algoritma.

A. Membuat fungsi pengujian waktu

```
def uji_waktu(algoritma,data,target):
    mulai=time.perf_counter()
    algoritma(data,target)
    selesai=time.perf_counter()
    return (selesai-mulai)*1000
```

B. Membuat grafik perbandingan:

```
plt.figure(figsize=(8,5))
plt.plot(
    df["Jumlah Data"],
    df["Linear Search (ms)"],
    marker="o",
    label="Linear Search"
)
plt.plot(
    df["Jumlah Data"],
    df["Binary Search (ms)"],
    marker="o",
    label="Binary Search"
)

plt.xlabel("Jumlah Data")
plt.ylabel(
    "Waktu Eksekusi (ms)"
)
plt.title(
    "Perbandingan Linear Search dan Binary Search"
)
plt.legend()
plt.grid()
plt.show()
```

Tabel 3 menunjukkan hasil pengukuran waktu eksekusi yang diperoleh.

Jumlah Data	Linear Search (ms)	Binary Search
100	0.15	0.03
1.000	1.50	0.05
10.000	15.20	0.08
100.000	150.00	0.11

Berdasarkan hasil pengujian pada Tabel 3 terlihat bahwa kedua algoritma mengalami peningkatan waktu eksekusi seiring bertambahnya jumlah data. Namun tingkat peningkatan yang terjadi pada masing-masing algoritma menunjukkan pola yang berbeda.

Linear Search mengalami peningkatan waktu yang cukup signifikan ketika ukuran dataset bertambah. Pada dataset berukuran 100 data, waktu pencarian hanya sebesar 0,15 ms. Namun ketika jumlah data meningkat menjadi 100.000 data, waktu pencarian meningkat hingga mencapai 150 ms.

Sebaliknya, Binary Search menunjukkan peningkatan waktu yang relatif kecil. Pada dataset 100 data, waktu pencarian tercatat sebesar 0,03 ms dan hanya meningkat menjadi 0,11 ms ketika dataset mencapai 100.000 data. Hasil ini menunjukkan bahwa Binary Search mampu mempertahankan performa pencarian meskipun ukuran data meningkat secara signifikan.

3.3 Analisis Kompleksitas Linear Search

Linear Search bekerja dengan memeriksa setiap elemen secara berurutan hingga target ditemukan. Dalam kondisi terburuk, algoritma harus melakukan pemeriksaan terhadap seluruh elemen yang tersedia. Secara teoritis, kompleksitas waktu Linear Search dinyatakan sebagai: Kompleksitas tersebut menunjukkan bahwa jumlah operasi pencarian bertambah secara linear terhadap ukuran input. (Zinnia & Hanada, 2024)

Dengan kata lain, jika jumlah data meningkat sepuluh kali lipat, maka jumlah operasi pencarian juga akan meningkat mendekati sepuluh kali lipat. Hasil pengujian empiris menunjukkan pola yang sesuai dengan teori tersebut. Ketika jumlah data bertambah dari 100 menjadi 100.000 data, waktu pencarian meningkat secara signifikan. Fenomena ini menunjukkan bahwa Linear Search kurang optimal untuk digunakan pada sistem yang mengelola data dalam jumlah besar

3.4 Analisis Kompleksitas Binary Search

Binary Search menggunakan pendekatan divide and conquer dengan membagi ruang pencarian menjadi dua bagian pada setiap iterasi (Muhimmah et al., 2025). Metode ini memungkinkan pengurangan jumlah elemen yang diperiksa secara drastis. Secara teoritis, kompleksitas waktu Binary Search dinyatakan sebagai: Kompleksitas logaritmik menunjukkan bahwa peningkatan ukuran dataset tidak memberikan dampak yang terlalu besar terhadap jumlah operasi pencarian. Sebagai contoh, pada dataset berukuran 100.000 data,
Binary Search hanya membutuhkan sekitar 17 langkah pencarian untuk menemukan target. Hasil pengujian empiris menunjukkan bahwa waktu pencarian Binary Search meningkat sangat kecil dibandingkan Linear Search. Hal ini membuktikan bahwa pendekatan divide and conquer mampu meningkatkan efisiensi pencarian secara signifikan.

3.5 Perbandingan Kinerja Algoritma

Berdasarkan hasil pengujian, Binary Search memiliki performa yang lebih baik dibandingkan Linear Search pada seluruh ukuran dataset yang digunakan. Perbedaan performa semakin terlihat ketika ukuran data bertambah besar. Pada dataset berukuran 100 data, selisih waktu pencarian relatif kecil. Namun pada dataset 100.000 data, Binary Search mampu melakukan pencarian jauh lebih cepat dibandingkan Linear Search (Ramadhan et al., 2023).

A. Membuat Grafik Perbandingan

```
plt.figure(figsize=(8,5))
plt.plot(
   df["Jumlah Data"],
   df["Linear Search (ms)"],
   marker="o",
   label="Linear Search"
)
plt.plot(
   df["Jumlah Data"],
   df["Binary Search (ms)"],
   marker="o",
   label="Binary Search"
)
plt.xlabel("Jumlah Data")
plt.ylabel( "Waktu Eksekusi (ms)"
)
plt.title(
   "Perbandingan Linear Search dan Binary Search"
)

plt.legend()
plt.grid()
plt.show()
```

B. Membuat Tabel Hasil:

```
df=pd.DataFrame(
   hasil,
```

```
    columns=[
        "Jumlah Data",
        "Linear Search (ms)",
        "Binary Search (ms)"
    ]
)
df
```

Jika dibandingkan secara langsung, waktu eksekusi Linear Search pada dataset 100.000 data mencapai 150 ms, sedangkan Binary Search hanya membutuhkan 0,11 ms. Dengan demikian Binary Search memiliki performa sekitar 1.363 kali lebih cepat dibandingkan Linear Search pada kondisi pengujian tersebut. Temuan ini menunjukkan bahwa pemilihan algoritma memiliki pengaruh yang sangat besar terhadap efisiensi sistem informasi, khususnya pada aplikasi yang melakukan proses pencarian secara intensif.

3.6 Pembahasan Hasil Penelitian

Hasil penelitian menunjukkan bahwa teori kompleksitas algoritma memiliki keterkaitan yang kuat dengan performa aktual yang diperoleh melalui pengujian empiris. Linear Search yang memiliki kompleksitas O(n) menunjukkan peningkatan waktu pencarian yang sebanding dengan pertumbuhan ukuran dataset. Sebaliknya, Binary Search yang memiliki kompleksitas O(log n) menunjukkan peningkatan waktu yang jauh lebih kecil. Temuan penelitian ini sejalan dengan penelitian yang dilakukan oleh (Kurhade & Takawale, 2024) yang menyatakan bahwa Binary Search memiliki efisiensi yang lebih baik dibandingkan Linear Search pada dataset berukuran besar. Hasil penelitian ini juga mendukung penelitian (Surampudi et al., 2023) yang menemukan bahwa Binary Search memberikan performa pencarian yang lebih tinggi dibandingkan Linear Search ketika data berada dalam kondisi terurut. Meskipun demikian, Binary Search memiliki keterbatasan karena memerlukan data yang telah diurutkan terlebih dahulu. Oleh karena itu, pada kondisi tertentu seperti dataset kecil atau data yang sering berubah, Linear Search masih dapat menjadi alternatif yang layak digunakan karena lebih sederhana dan tidak membutuhkan proses sorting tambahan.

3.7 Implikasi Penelitian

Hasil penelitian memberikan implikasi praktis bagi pengembangan sistem informasi akademik. Pada sistem yang mengelola ribuan hingga ratusan ribu data mahasiswa, penggunaan Binary Search dapat meningkatkan efisiensi proses pencarian dan mengurangi waktu respons sistem secara signifikan (Julio Tabea, 2026). Selain itu, penelitian ini juga memberikan kontribusi teoritis dengan menunjukkan bahwa hasil pengujian empiris konsisten dengan teori kompleksitas algoritma yang telah dijelaskan dalam literatur ilmu komputer. Dengan demikian, penelitian ini dapat menjadi referensi bagi pengembang perangkat lunak maupun peneliti yang ingin melakukan evaluasi performa algoritma pencarian pada berbagai skenario penggunaan data.

4. KESIMPULAN

Penelitian ini bertujuan untuk membandingkan kinerja algoritma Linear Search dan Binary Search pada proses pencarian data mahasiswa berdasarkan analisis kompleksitas teoretis dan pengujian empiris. Pengujian dilakukan menggunakan dataset mahasiswa simulasi dengan ukuran yang bervariasi, yaitu 100, 1.000, 10.000, dan 100.000 data. Parameter utama yang digunakan dalam evaluasi meliputi waktu eksekusi pencarian dan kompleksitas algoritma.

Berdasarkan hasil pengujian yang telah dilakukan, dapat disimpulkan bahwa terdapat perbedaan performa yang signifikan antara Linear Search dan Binary Search. Linear Search menunjukkan peningkatan waktu pencarian yang sebanding dengan pertumbuhan jumlah data. Pada dataset yang lebih besar, waktu eksekusi algoritma meningkat secara signifikan karena setiap elemen harus diperiksa secara berurutan hingga data yang dicari ditemukan. Hasil tersebut sesuai dengan karakteristik kompleksitas waktu Linear Search yang bersifat linear atau O(n).
yang diuji. Meskipun jumlah data meningkat hingga 100.000 data, waktu pencarian hanya mengalami peningkatan yang relatif kecil. Hal ini disebabkan oleh mekanisme divide and conquer yang digunakan Binary Search untuk memperkecil ruang pencarian pada setiap iterasi. Temuan ini mendukung teori bahwa Binary Search memiliki kompleksitas waktu O(log n), sehingga lebih efisien dibandingkan Linear Search ketika digunakan pada dataset berukuran besar.

Hasil penelitian juga menunjukkan bahwa analisis kompleksitas teoretis memiliki kesesuaian yang kuat dengan hasil pengujian empiris. Semakin besar ukuran dataset yang digunakan, semakin terlihat perbedaan performa antara kedua algoritma. Dengan demikian, penelitian ini berhasil membuktikan bahwa kompleksitas algoritma merupakan faktor yang sangat berpengaruh terhadap efisiensi proses pencarian data.

Dari sisi implementasi, Binary Search direkomendasikan untuk diterapkan pada sistem informasi akademik yang mengelola data mahasiswa dalam jumlah besar dan memiliki struktur data yang terurut. Penggunaan algoritma ini dapat meningkatkan efisiensi pencarian, mengurangi waktu respons sistem, serta mendukung kinerja aplikasi yang lebih optimal. Sementara itu, Linear Search masih dapat digunakan pada dataset berukuran kecil atau pada kondisi data yang belum terurut karena memiliki implementasi yang lebih sederhana dan tidak memerlukan proses pengurutan data.

Kontribusi penelitian ini terletak pada penyajian evaluasi komprehensif mengenai hubungan antara kompleksitas teoretis dan kinerja empiris algoritma pencarian dalam konteks sistem informasi akademik. Selain memberikan validasi terhadap teori analisis algoritma, penelitian ini juga dapat menjadi referensi bagi pengembang sistem dan peneliti dalam menentukan algoritma pencarian yang sesuai berdasarkan karakteristik data yang digunakan.

Untuk penelitian selanjutnya, disarankan melakukan pengujian menggunakan dataset nyata dari sistem informasi akademik serta membandingkan algoritma pencarian lainnya seperti Interpolation Search, Jump Search, Exponential Search, dan Hash Search. Selain itu, penelitian mendatang dapat menambahkan parameter evaluasi seperti penggunaan memori, jumlah operasi perbandingan, dan pengaruh proses sorting terhadap total waktu pencarian sehingga diperoleh pemahaman yang lebih menyeluruh mengenai efisiensi algoritma pencarian pada berbagai kondisi data.

DAFTAR PUSTAKA

Januari, V. N., Purba, A., Dwi, T., Informasi, S., Kaputama, S., Informasi, S., Ekonomi, F., & Asahan, U. M. (2026). *Implementasi Algoritma Binary Search pada Sistem Temu Balik Informasi Berbasis Web dengan Input File Excel*. *2*(1), 35–40.

Kurhade, M. A., & Takawale, M. N. N. (2024). Comparative Study of Searching Algorithm: Linear Search and Binary search. *Ijarcce*, *13*(5), 59–62. https://doi.org/10.17148/ijarcce.2024.13506

Leana, S. A., Aditya, M., Ginting, P., Bukhari Izdihar, M., Syahputra, T., Pratama, A., Manik, A. A., & Gunawan, I. (2025). Perbandingan Efisiensi Linear Dan Binary Search Dalam Pencarian Nama Siswa Pada Struktur Data Array. *Jurnal Riset Sistem Informasi Dan Aplikasi Komputer (JRSIKOM)*, *1*(2), 45–50.

Muhimmah, N., Najah, L., & Huda, W. S. (2025). *Implementation of binary search on website-based traffic data*. *14*(2), 179–187.

Nurvita, R., & Putri, M. (2025). Implementasi dan Analisis Algoritma Binary Search. *Maliki Interdisciplinary Journal (MIJ) EISSN*, *3*, 289–294. http://urj.uin-malang.ac.id/index.php/mij/index

Paleyes, A., Cabrera, C., & Lawrence, N. D. (2022). An Empirical Evaluation of Flow Based Programming in the Machine Learning Deployment Context. *Proceedings - 1st International Conference on AI Engineering - Software Engineering for AI, CAIN 2022*, 54–64. https://doi.org/10.1145/3522664.3528601

Julio T, Santa K, Hasibuan. (2026). *Penerapan Algoritma Binary Search pada Aplikasi JDIH Universitas Negeri Manado*. *10*(1), 127–134.

Prof.Mrs. Tejaswini.A. Puranik. (2025). Performance Analysis of Sorting and Searching Algorithms. *International Research Journal on Advanced Engineering and Management (IRJAEM)*, *3*(08), 2741–2746. https://doi.org/10.47392/irjaem.2025.0430

Purnama, N. (2025). *LARGE-SCALE DATA STRUCTURES*. *11*(1), 99–109. https://doi.org/10.33480/jitk.v11i1.6592.COMPARATIVE

Ramadhan, H., Fauziah, F., & Lantana, D. A. (2023). Perbandingan Algoritma Binary Search dan Sequential Search untuk Pencarian Persediaan Stok Barang Berbasis Web. *STRING (Satuan Tulisan Riset Dan Inovasi Teknologi)*, *8*(2), 147. https://doi.org/10.30998/string.v8i2.16475

Sari, N., & Rosadi, D. (2017). Analisis Perbandingan Algoritma Sequential Search Dan Binary Search Dalam Pencarian Data. *Jurnal ELKOM*, *9*(2), 1–8. https://doi.org/10.33481/infomans.vxxixx

Sundaramoorthy, S., & Karunanidhi, G. (2025). A systematic analysis on performance and computational complexity of sorting algorithms. *Discover Computing*, *28*(1). https://doi.org/10.1007/s10791-025-09724-w

Surachman, R. I., Supriadi, F., Saeppani, A., & Mahardika, F. (2025). *Perbandingan Kinerja Algoritma Linear Search dan Binary Search dalam Pencarian Data*. *19*(2), 1–6. https://doi.org/10.33481/infomans.vxxixx

Surampudi, Y., Kondaveeti, D., & Pichaimani, T. (2023). A Comparative Study of Time Complexity in Big Data Engineering: Evaluating Efficiency of Sorting and Searching Algorithms in Large-Scale Data Systems. *Journal of Science & Technology By The Science Brigade (Publishing) Group 127 JOURNAL OF*

SCIENCE & TECHNOLOGY, *4*(4), 127–165.
Yasmin, N. N., Sofia, L., Sabrina, A. R. P., Putri, A. A.-Z., & Pujiono, I. P. (2025). Interpolation Searching Algorithm Vs Algoritma Pencarian Tradisional: Analisis Efisiensi Memori dan Waktu Komputasi. *Simkom*, *10*(2), 212–223. https://doi.org/10.51717/simkom.v10i2.857
Zidan, R., Rehman, K. N. U., & Khan, I. (2025). Experimental Performance Benchmarking of Popular Search Algorithms in Java and Python. *International Journal of Computer Applications*, *187*(62), 31–38. https://doi.org/10.5120/ijca2025926016
Zinnia, N. A., & Hanada, E. (2024). Optimizing Search Strategies: A Study of Two-Pointer Linear Search Implementation. *Arxiv*, 1–5.