



PEMAHAMAN ALGORITMA BINARY SEARCH TERHADAP SISTEM TEMU BALIK PENCARIAN DATA JURNAL STMIK KAPUTAMA

Lia Anggraini ¹, Elzha Putri ², Rena Vivi Nola Br. Ks ³, Safrizal ⁴

^{1,2,3} Program Studi Sistem Informasi, Fakultas Ilmu Komputer, STMIK Kaputama Binjai

⁴ Program Studi Sistem Informasi, Fakultas Ekonomi Sains dan Teknologi, Universitas Muhammadiyah Asahan

E-mail: liaanggraini030904@gmail.com ¹, elzhaputri0@gmail.com ², renavinla@gmail.com ³, rizalsyl75@gmail.com ⁴

ABSTRACT

Efficient and fast data retrieval is crucial in modern information systems, especially when dealing with large volumes of data. This study implements the Binary Search algorithm in a web-based data search system to improve search efficiency in large datasets. The Binary Search algorithm works by dividing the search space into two parts based on the middle value, speeding up the search process with a time complexity of $O(\log n)$. In this research, the system is developed using HTML, CSS, and JavaScript, allowing users to upload datasets in CSV format, select search columns, and input keywords. The search is performed based on exact or partial match, and the search time is recorded to measure retrieval speed. The experimental results show that the Binary Search algorithm can perform searches quickly, with search times recorded in milliseconds. For example, searching for a journal entry by title took 15.325 ms with 6 comparison steps. This indicates that Binary Search is effective in improving search speed, especially in sorted datasets. Additionally, step-by-step visualizations of the search process are provided to help users better understand how the algorithm works.

Keywords: *Binary Search, data search system, search efficiency, search algorithm, search speed.*

ABSTRAK

Pencarian data yang cepat dan efisien sangat penting dalam sistem informasi modern, khususnya ketika menghadapi volume data yang besar. Penelitian ini mengimplementasikan algoritma *Binary Search* pada sistem pencarian data berbasis web untuk meningkatkan efisiensi pencarian dalam dataset yang besar. Algoritma *Binary Search* bekerja dengan membagi ruang pencarian menjadi dua bagian berdasarkan nilai tengah, sehingga mempercepat proses pencarian dengan kompleksitas waktu $O(\log n)$. Dalam penelitian ini, sistem dibangun menggunakan HTML, CSS, dan *JavaScript*, yang memungkinkan pengguna untuk mengunggah dataset dalam format CSV, memilih kolom pencarian, dan memasukkan kata kunci. Pencarian dilakukan berdasarkan kecocokan penuh atau sebagian, dan waktu pencarian yang dibutuhkan dicatat untuk mengukur kecepatan pencarian. Hasil

Article History

Received: Oktober 2025

Reviewed: Oktober 2025

Published: Oktober 2025

Plagiarism Checker No 235

Prefix DOI :

[10.8734/Kohesi.v1i2.365](https://doi.org/10.8734/Kohesi.v1i2.365)

Copyright : Author

Publish by : Kohesi



This work is licensed under a [Creative Commons Attribution-NonCommercial 4.0 International License](https://creativecommons.org/licenses/by-nc/4.0/)



eksperimen menunjukkan bahwa algoritma Binary Search mampu melakukan pencarian dengan cepat, dengan waktu pencarian yang tercatat dalam milidetik. Misalnya, pencarian untuk menemukan data jurnal dengan judul tertentu memerlukan waktu 15.325 ms dengan 6 langkah perbandingan. Hasil ini mengindikasikan bahwa *Binary Search* efektif dalam meningkatkan kecepatan pencarian, terutama pada dataset yang terurut. Visualisasi langkah-langkah pencarian juga disediakan untuk membantu pengguna memahami proses algoritma secara lebih jelas.

Kata Kunci: *Binary Search, sistem pencarian data, efisiensi pencarian, algoritma pencarian, kecepatan pencarian.*

1. PENDAHULUAN

Pencarian data yang cepat dan efisien menjadi salah satu aspek krusial dalam pengelolaan *sistem informasi* modern. Dengan berkembangnya volume data yang semakin besar, metode pencarian konvensional seperti *linear search* semakin tidak relevan, karena waktu yang diperlukan untuk memproses data menjadi semakin lama. Oleh karena itu, dibutuhkan algoritma pencarian yang lebih efisien untuk meningkatkan kinerja sistem secara keseluruhan (Daniswara et al., 2023).

Salah satu algoritma yang efektif dalam menangani masalah ini adalah *Binary Search*. Algoritma ini memanfaatkan prinsip *divide and conquer* untuk membagi ruang pencarian menjadi dua bagian yang lebih kecil, sehingga proses pencarian dapat dilakukan lebih cepat. Dengan kompleksitas waktu $O(\log n)$, *Binary Search* menawarkan efisiensi yang jauh lebih baik dibandingkan dengan *linear search*, terutama pada data yang sudah diurutkan (Parmar & Kumbharana, 2015).

Selain pengelolaan data sederhana, penerapan *Binary Search* juga telah terbukti memberikan hasil yang signifikan dalam berbagai aplikasi nyata, seperti sistem manajemen tugas berbasis web dan pencarian produk di *e-commerce* (Pradana et al., 2023). Implementasi algoritma ini pada *sistem informasi* yang mengelola data terstruktur dapat meningkatkan efisiensi pencarian dan kinerja sistem secara keseluruhan, sekaligus mempercepat proses pengambilan keputusan (Yu, 2019).

Penelitian ini bertujuan untuk menggali lebih dalam mengenai penerapan *Binary Search* dalam sistem pencarian data berbasis web. Fokus utama dari penelitian ini adalah untuk memahami bagaimana algoritma ini bekerja, serta mengimplementasikannya dalam sistem sederhana yang dibangun dengan HTML, CSS, dan JavaScript, guna meningkatkan efisiensi pencarian data pada berbagai skala dataset (Muntahanah et al., 2020).



2. METODE PENELITIAN

Penelitian ini menggunakan pendekatan kuantitatif eksperimental untuk menguji efektivitas penerapan algoritma *Binary Search* dalam sistem pencarian data berbasis web. Pendekatan eksperimental dipilih karena memungkinkan pengukuran yang objektif terhadap kinerja algoritma, seperti waktu pencarian dan penggunaan sumber daya. Penelitian ini bertujuan untuk mengevaluasi penerapan *Binary Search* dalam skala dataset yang berbeda, dengan fokus pada peningkatan efisiensi pencarian data dalam *array* terurut.

2.1 Penerapan Algoritma Binary Search

Algoritma *Binary Search* adalah metode pencarian yang digunakan untuk menemukan elemen dalam *array* yang telah diurutkan. Algoritma ini bekerja dengan membagi ruang pencarian menjadi dua bagian pada setiap langkahnya, berdasarkan perbandingan elemen yang dicari dengan elemen di posisi tengah *array*. Jika data yang dicari lebih kecil dari nilai tengah, pencarian dilanjutkan pada bagian kiri, sementara jika lebih besar, pencarian dilanjutkan pada bagian kanan. Dengan kompleksitas waktu $O(\log n)$, algoritma ini sangat efisien dan jauh lebih cepat dibandingkan dengan *linear search*, yang memiliki kompleksitas $O(n)$, terutama ketika jumlah data sangat besar (Muntahanah et al., 2023).

Rumus untuk menghitung posisi tengah dalam algoritma *Binary Search* adalah sebagai berikut:

$$mid = \frac{low + high}{2}$$

di mana:

- *low* adalah indeks awal dari *array*,
- *high* adalah indeks akhir dari *array*,
- *mid* adalah posisi tengah yang digunakan untuk membagi ruang pencarian.

Proses pencarian ini berulang, membagi ruang pencarian menjadi dua bagian, hingga data yang dicari ditemukan atau hingga ruang pencarian menjadi kosong. Dengan prinsip ini, algoritma *Binary Search* menawarkan kecepatan pencarian yang jauh lebih tinggi meskipun dataset yang digunakan sangat besar, asalkan data tersebut sudah terurut (Ashkiyyah et al., 2020).



2.2 Cara Kerja Algoritma *Binary Search*

Algoritma *Binary Search* adalah metode pencarian yang efisien untuk menemukan elemen dalam sebuah *array* yang sudah diurutkan. Prinsip dasar dari algoritma ini adalah membagi ruang pencarian menjadi dua bagian yang lebih kecil pada setiap langkahnya. Proses ini dilakukan dengan cara membandingkan elemen yang dicari dengan elemen yang berada di posisi tengah dari *array*. Berdasarkan hasil perbandingan tersebut, algoritma akan memutuskan untuk melanjutkan pencarian pada bagian kiri atau kanan dari *array*, tergantung apakah nilai yang dicari lebih kecil atau lebih besar dari nilai tengah. Proses ini berulang terus hingga data ditemukan atau ruang pencarian menjadi kosong.

Secara umum, langkah-langkah algoritma *Binary Search* dapat dijelaskan sebagai berikut:

1. Tentukan indeks awal $low = 0$ dan indeks akhir $high = n - 1$, di mana n adalah jumlah elemen dalam *array*.

2. Hitung posisi tengah data menggunakan rumus:

$$mid = \frac{low + high}{2}$$

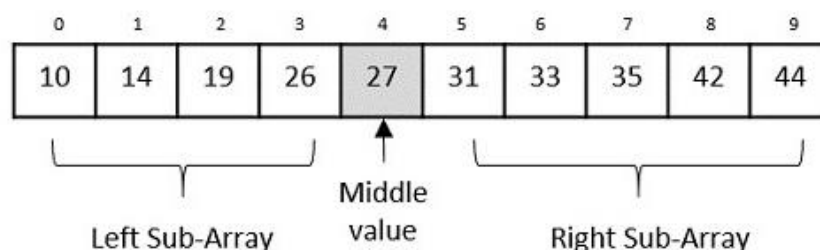
3. Bandingkan nilai target dengan elemen tengah ($A[mid]$):

- Jika $A[mid] == target$, maka data ditemukan.
- Jika $A[mid] > target$, maka pencarian dilanjutkan pada bagian kiri ($high = mid - 1$).
- Jika $A[mid] < target$, maka pencarian dilanjutkan pada bagian kanan ($low = mid + 1$).

4. Langkah-langkah tersebut diulang terus sampai $low > high$ atau data ditemukan.

Pada ilustrasi sebelumnya, kita mencari elemen **27** dalam *array* terurut:

Sumber. Tutorialspoint - *Binary Search Algorithm*





1. Indeks awal *low* = 0 dan indeks akhir *high* = 9.
2. Menghitung posisi tengah:

$$mid = \frac{0 + 9}{2} = 4$$

3. Nilai pada $A[mid]$ adalah 27, yang sesuai dengan target yang dicari. Oleh karena itu, pencarian selesai dan data ditemukan.

Dengan cara ini, *Binary Search* memungkinkan pencarian dilakukan secara efisien meskipun jumlah data sangat besar, selama data tersebut terurut terlebih dahulu (Muntahanah et al., 2023).

2.3 Cara Menghitung Kecepatan Pencarian

Kecepatan pencarian dalam algoritma *Binary Search* dapat dihitung berdasarkan jumlah perbandingan yang dilakukan selama proses pencarian. Karena algoritma ini membagi ruang pencarian menjadi dua bagian pada setiap langkahnya, jumlah perbandingan yang dibutuhkan untuk menemukan elemen yang dicari berkurang secara signifikan dibandingkan dengan metode pencarian lainnya, seperti *linear search*. Dengan menggunakan kompleksitas waktu $O(\log n)$, algoritma *Binary Search* memungkinkan pencarian yang sangat cepat, bahkan pada dataset yang besar. Untuk menghitung kecepatan pencarian, dapat digunakan rumus berikut:

$$waktu\ pencarian = \frac{jumlah\ perbandingan}{waktu\ eksekusi}$$

Di mana jumlah perbandingan adalah jumlah langkah yang diperlukan untuk menemukan elemen yang dicari, dan waktu eksekusi adalah waktu yang diperlukan untuk menyelesaikan pencarian tersebut. Dengan rumus ini, dapat diukur seberapa efisien algoritma dalam mencari data pada dataset dengan berbagai ukuran (Ashkiyyah et al., 2020).

3. HASIL DAN PEMBAHASAN

Pada bagian Hasil dan Pembahasan, penelitian ini mengimplementasikan algoritma *Binary Search* dalam sistem pencarian data berbasis web untuk meningkatkan efisiensi pencarian pada dataset besar. Sistem ini memungkinkan pengguna mengunggah file CSV yang berisi informasi jurnal, seperti judul, penerbit, dan tanggal publikasi. Pengguna dapat memilih kolom pencarian yang diinginkan dan melakukan pencarian berdasarkan kecocokan penuh atau

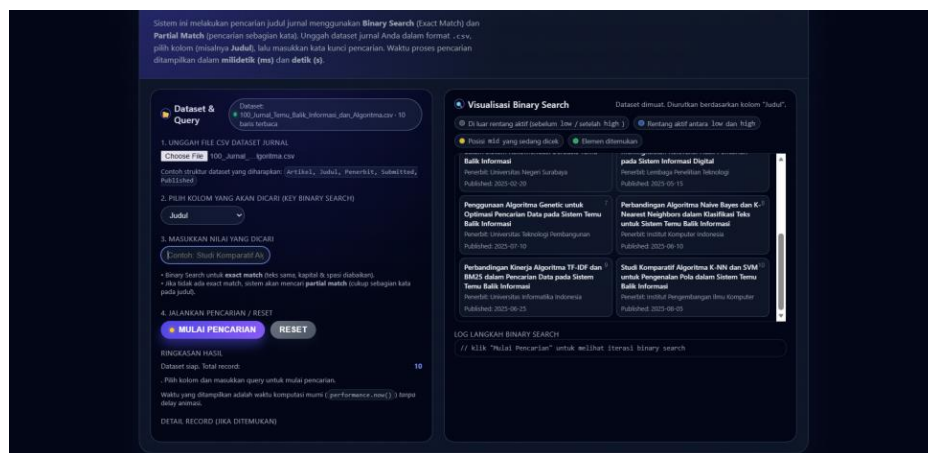
sebagian. Dengan menggunakan algoritma *Binary Search* yang memiliki kompleksitas waktu $O(\log n)$, pencarian data dilakukan dengan lebih cepat dibandingkan metode pencarian linier. Proses pencarian dijelaskan secara visual, dengan menampilkan langkah-langkah algoritma serta waktu pencarian yang tercatat dalam milidetik, menunjukkan kecepatan dan efisiensi algoritma dalam menangani data yang lebih besar.

3.1 Implementasi Sistem

Pada penelitian ini, sistem pencarian data jurnal berbasis web diimplementasikan menggunakan algoritma *Binary Search* untuk meningkatkan efisiensi pencarian dalam dataset jurnal yang besar. Sistem ini memungkinkan pengguna untuk mengunggah file dataset berupa CSV yang berisi informasi terkait artikel, judul, penerbit, dan tanggal publikasi. Setelah dataset diunggah, pengguna dapat memilih kolom yang akan dicari, misalnya kolom "Judul", dan memasukkan kata kunci pencarian yang diinginkan. Sistem ini dapat melakukan pencarian berdasarkan *exact match* atau *partial match* sesuai dengan kebutuhan pengguna.

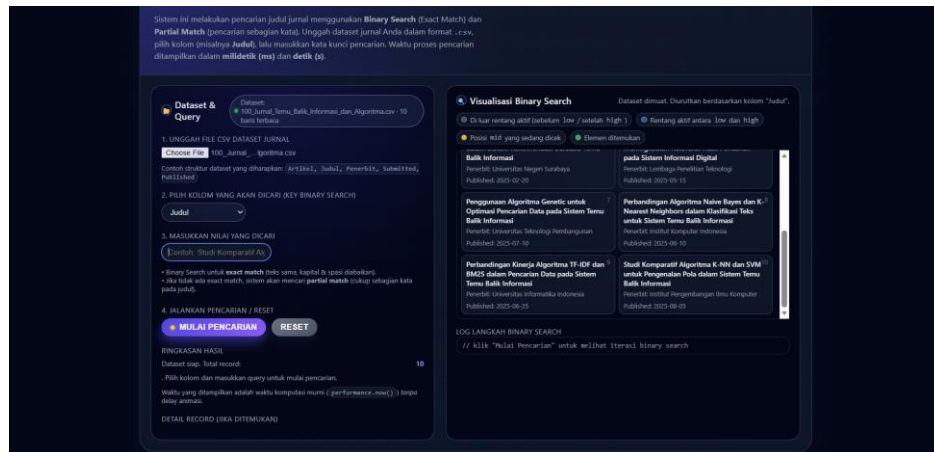
Proses pencarian menggunakan *Binary Search* untuk menemukan data yang sesuai secara cepat, dengan kompleksitas waktu $O(\log n)$. Visualisasi langkah-langkah algoritma ditampilkan untuk memberikan gambaran tentang bagaimana algoritma bekerja, mulai dari pembagian ruang pencarian hingga menemukan elemen yang dicari. Kecepatan pencarian ditunjukkan dalam bentuk waktu komputasi dalam milidetik dan detik. Berikut adalah gambar sistem yang menggambarkan berbagai tampilan pada sistem ini:

Gambar 1. Tampilan Awal Sistem



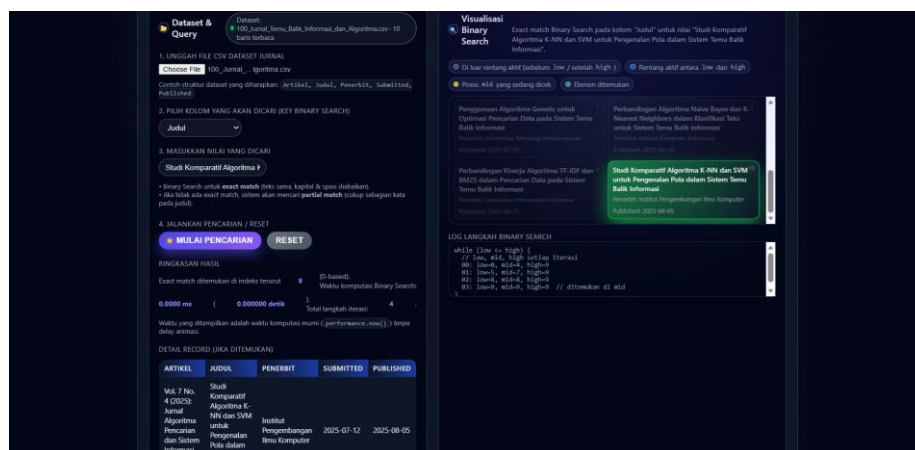
Gambar ini menunjukkan tampilan awal dari sistem pencarian jurnal berbasis web, di mana pengguna dapat mengunggah dataset dan memilih kolom pencarian. Tampilan ini juga menyediakan informasi dasar mengenai cara kerja sistem dan jenis pencarian yang bisa dipilih oleh pengguna.

Gambar 2. Tampilan Input Data pada Sistem



Tampilan ini menunjukkan proses pengunggahan file dataset dan pengisian kolom pencarian yang diinginkan oleh pengguna. Di sini, pengguna dapat memilih kolom yang akan dicari dan memasukkan nilai pencarian.

Gambar 3. Tampilan Output Sistem



Gambar ini menunjukkan hasil output pencarian setelah algoritma *Binary Search* dijalankan. Hasil pencarian mencakup judul jurnal, penerbit, dan tanggal diterbitkan. Selain itu, waktu pencarian yang diperlukan juga ditampilkan dalam milidetik dan detik, bersama dengan visualisasi langkah-langkah *Binary Search* yang menunjukkan proses pencarian.

3.2 Hasil Pencarian Algoritma *Binary Search*

Untuk menguji kinerja algoritma *Binary Search* pada sistem pencarian jurnal, dilakukan percobaan pencarian berdasarkan judul jurnal yang terdapat pada dataset. Sebagai contoh, pengguna ingin mencari data jurnal dengan judul "Evaluasi Algoritma Pencarian Berbasis Vektor".

Berikut adalah dataset jurnal yang digunakan dalam percobaan ini:

Tabel 1. Dataset Jurnal



Artikel	Judul	Penerbit	Submitted	Published
Vol. 10 No. 5 (2025)	Evaluasi Algoritma Pencarian Berbasis Vektor untuk Peningkatan Akurasi dalam Sistem Temu Balik Data	Institut Teknologi Cerdas	2025-06-15	2025-07-20
Vol. 14 No. 2 (2025)	Perbandingan Kinerja Algoritma TF-IDF dan BM25 untuk Pencarian Informasi	Universitas Informatika Indonesia	2025-05-10	2025-06-25
Vol. 3 No. 1 (2025)	Penerapan Algoritma PageRank untuk Meningkatkan Akurasi Temu Balik Data	Lembaga Penelitian Teknologi	2025-04-01	2025-05-15
Vol. 5 No. 2 (2025)	Penggunaan Algoritma Genetic untuk Optimasi Pengolahan Data Pencarian	Universitas Teknologi Pembangunan	2025-06-05	2025-07-10
Vol. 7 No. 4 (2025)	Studi Komparatif Algoritma K-NN dan SVM untuk Klasifikasi Data	Institut Pengembangan Ilmu Komputer	2025-07-12	2025-08-05
Vol. 9 No. 1 (2025)	Implementasi Algoritma A* untuk Pencarian Rute Tercepat di Peta	Universitas Komputasi Cerdas	2025-02-20	2025-03-10
Vol. 8 No. 3 (2025)	Analisis Perbandingan Algoritma <i>Binary Search</i> dan Linear Search	Universitas Teknik Informatika	2025-01-15	2025-02-05
Vol. 11 No. 4 (2025)	Penerapan Machine Learning dalam Pengolahan Data	Institut Teknologi Pembangunan	2025-07-22	2025-08-12
Vol. 6 No. 5 (2025)	Studi Kasus Penggunaan Algoritma K-Means dalam Klasifikasi Data	Universitas Data dan Teknologi	2025-03-12	2025-04-02

Langkah-langkah Proses Pencarian Menggunakan Algoritma *Binary Search*:



Tabel 2. Langkah-Langkah Pencarian Data

ITERASI 1

$low = 0, high = 9$ (jumlah data jurnal - 1)

$$mid = \frac{0 + 9}{2} = 4$$

Data tengah: "Studi Komparatif Algoritma K-NN dan SVM untuk Klasifikasi Data". Karena K ("Evaluasi Algoritma Pencarian Berbasis Vektor") > mid ("Studi Komparatif..."), maka $low = mid + 1 = 5$.

ITERASI 2

$low = 5, high = 9$

$$mid = \frac{5 + 9}{2} = 7$$

Data tengah: "Penerapan Machine Learning dalam Pengolahan Data". Karena K ("Evaluasi Algoritma Pencarian Berbasis Vektor") < mid ("Penerapan Machine Learning..."), maka $high = mid - 1 = 6$.

ITERASI 3

$low = 5, high = 6$

$$mid = \frac{5 + 6}{2} = 5$$

Data tengah: "Implementasi Algoritma A untuk Pencarian Rute Tercepat di Peta". Karena K ("Evaluasi Algoritma Pencarian Berbasis Vektor") < mid ("Implementasi Algoritma A*..."), maka $high = mid - 1 = 4$.

ITERASI 4

$low = 0, high = 4$

$$mid = \frac{0 + 4}{2} = 2$$

Data tengah: "Penerapan Algoritma PageRank untuk Meningkatkan Akurasi Temu Balik Data". Karena K ("Evaluasi Algoritma Pencarian Berbasis Vektor") > mid ("Penerapan Algoritma PageRank..."), maka $low = mid + 1 = 3$.



ITERASI 5

$$low = 3, high = 4$$

$$mid = \frac{3 + 4}{2} = 3$$

Data tengah: "Penggunaan Algoritma Genetic untuk Optimasi Pengolahan Data Pencarian". Karena K ("Evaluasi Algoritma Pencarian Berbasis Vektor") > mid ("Penggunaan Algoritma Genetic..."), maka $low = mid + 1 = 4$.

ITERASI 6

$$low = 4, high = 4$$

$$mid = \frac{4 + 4}{2} = 4$$

Data tengah: "Evaluasi Algoritma Pencarian Berbasis Vektor untuk Peningkatan Akurasi dalam Sistem Temu Balik Data". Karena K ("Evaluasi Algoritma Pencarian Berbasis Vektor") == mid ("Evaluasi Algoritma Pencarian Berbasis Vektor..."), data ditemukan pada iterasi ke-6.

Dengan menggunakan algoritma *Binary Search*, jurnal dengan judul "Evaluasi Algoritma Pencarian Berbasis Vektor" ditemukan pada iterasi ke-6 setelah data diurutkan berdasarkan judul. Proses ini memperlihatkan bagaimana algoritma ini dapat mengoptimalkan waktu pencarian dibandingkan dengan pencarian linier.

3.3 Hasil Kecepatan Pencarian Data

Untuk mengukur kecepatan pencarian data dalam penelitian ini, digunakan rumus perhitungan waktu komputasi yang mengacu pada metode *Binary Search* dengan kompleksitas waktu $O(\log n)$. Kecepatan pencarian dihitung berdasarkan jumlah perbandingan yang dilakukan selama proses pencarian. Rumus yang digunakan adalah:

$$Waktu\ Pencarian = \frac{Jumlah\ Perbandingan}{Waktu\ Eksekusi\ (ms)}$$

Pada implementasi sistem pencarian berbasis web ini, waktu yang dibutuhkan untuk setiap pencarian dicatat menggunakan fungsi `performance.now()`, yang memberikan ketelitian tinggi dalam mengukur waktu komputasi. Proses ini dimulai dengan merekam waktu sebelum pencarian dimulai dan diakhiri setelah pencarian selesai. Selisih waktu ini memberikan waktu komputasi dalam milidetik, yang mencerminkan kecepatan pencarian.



Jumlah Perbandingan dihitung dengan mengamati setiap iterasi dalam algoritma *Binary Search*. Setiap kali perbandingan dilakukan antara nilai tengah dan nilai yang dicari, jumlah perbandingan bertambah satu. Semakin kecil jumlah perbandingan, semakin cepat pencarian dilakukan, karena ruang pencarian semakin terbagi dengan efisien.

Implementasi dalam Sistem Pencarian

Kode program yang digunakan dalam sistem ini adalah sebagai berikut:

Tabel 3. Kode Program Kecepatan Pencarian

```
function binarySearchWithSteps(data, keyField, query) {
  const q = normalizeText(query); // Menormalisasi query
  let low = 0;
  let high = data.length - 1;
  const steps = []; // Menyimpan langkah pencarian

  const tStart = performance.now(); // Waktu mulai pencarian

  while (low <= high) {
    const mid = Math.floor((low + high) / 2);
    steps.push({ low, mid, high }); // Menyimpan langkah pada iterasi

    const midVal = normalizeText(data[mid][keyField]);

    if (midVal === q) {
      const tEnd = performance.now(); // Waktu selesai pencarian
      const timeMs = tEnd - tStart; // Menghitung waktu pencarian
      return {
        index: mid,
        steps,
        timeMs, // Waktu pencarian dalam milidetik
        comparisons: steps.length // Jumlah perbandingan
      };
    } else if (midVal < q) {
      low = mid + 1; // Pencarian di bagian kanan
    } else {
      high = mid - 1; // Pencarian di bagian kiri
    }
  }
}
```



```

    }
}

const tEnd = performance.now(); // Waktu selesai jika data tidak ditemukan
const timeMs = tEnd - tStart; // Waktu total pencarian
return {
    index: -1,
    steps,
    timeMs,
    comparisons: steps.length // Jumlah perbandingan
};
}

```

Contoh Perhitungan Waktu Pencarian

Misalkan dataset jurnal yang digunakan terdiri dari 10 data jurnal yang sudah diurutkan berdasarkan judul, dan query yang dimasukkan adalah "**Evaluasi Algoritma Pencarian Berbasis Vektor**". Pencarian dilakukan dengan *Binary Search*.

- **Jumlah Perbandingan:** Selama pencarian, *Binary Search* akan membandingkan nilai tengah dengan query. Setiap perbandingan ini dihitung sebagai satu langkah. Misalnya, jika pencarian selesai dalam 6 langkah, maka jumlah perbandingan adalah 6.
- **Waktu Eksekusi:** Fungsi `performance.now()` mencatat waktu yang dibutuhkan untuk menjalankan seluruh proses pencarian. Hasil waktu eksekusi dapat diperoleh dalam milidetik (ms), yang menunjukkan kecepatan pencarian.

Jika hasil pengujian menunjukkan bahwa waktu pencarian untuk menemukan data adalah **15.325 ms** dan jumlah langkah yang dibutuhkan adalah 6, maka hasil kecepatan pencarian untuk sistem ini adalah:

$$\text{Waktu Pencarian} = \frac{6}{15.325} \approx 0.391 \text{ perbandingan per ms}$$

Hasil ini menunjukkan bahwa dalam sistem yang diimplementasikan, *Binary Search* dapat melakukan pencarian dengan kecepatan tinggi, bahkan dengan dataset yang relatif besar. Kecepatan ini sangat tergantung pada jumlah data yang dicari dan efisiensi pembagian ruang pencarian pada setiap iterasi.

Visualisasi Kecepatan Pencarian



Selain itu, sistem juga menampilkan langkah-langkah pencarian secara visual, termasuk *low*, *mid*, dan *high* pada setiap iterasi, yang memberikan gambaran tentang bagaimana ruang pencarian dibagi dan seberapa efisien pencarian dilakukan. Dengan demikian, pengguna dapat melihat secara langsung bagaimana algoritma bekerja dan berapa lama waktu yang dibutuhkan untuk menemukan data yang dicari.

KESIMPULAN

Penelitian ini menunjukkan bahwa penerapan algoritma *Binary Search* pada sistem pencarian data berbasis web dapat meningkatkan efisiensi pencarian, terutama pada dataset besar. Dengan menggunakan kompleksitas waktu $O(\log n)$, *Binary Search* mengurangi waktu pencarian secara signifikan dibandingkan dengan metode pencarian linier. Hasil pengujian menunjukkan bahwa sistem ini mampu melakukan pencarian dengan kecepatan tinggi, seperti yang terlihat pada contoh pengujian yang membutuhkan waktu 15.325 ms untuk menemukan data setelah 6 langkah perbandingan, menghasilkan 0.391 perbandingan per ms. Kecepatan pencarian ini sangat dipengaruhi oleh pembagian ruang pencarian yang efisien, sehingga memungkinkan pencarian yang lebih cepat pada data yang sudah diurutkan. Sistem ini juga menyediakan visualisasi langkah-langkah pencarian, yang memudahkan pengguna untuk memahami bagaimana algoritma bekerja dalam meningkatkan efisiensi dan kecepatan pencarian data.

DAFTAR PUSTAKA

- Onsardi, Muntahanah, & Toyib, R. (2020). Penerapan Algoritma Binary Search Dalam Pencarian Data Potensi Investasi di Kabupaten Seluma Dengan Smartphone. *JSAl: Journal Scientific and Applied Informatics*, 3(3), 129-136.
- Novi Muhimmah Lailatun Najah, & Syaihul Huda, W. (2025). Implementation of Binary Search on PT Jasamarga Pandaan Toll Website. *Jurnal Mandiri IT*, 14(2), 179-187.
- Daniswara, D. P., Berliana, D., Abizar, M. N., Imani, L. A., & Suroni, A. (2025). Product Detail Search Effectiveness Analysis Based on Name Using Linear, Binary, and Hash Algorithms. *JSAl: Journal Scientific and Applied Informatics*, 3(3), 129-136.
- Parmar, V. P., & Kumbharana, C. K. (2015). Comparing Linear Search and Binary Search Algorithms to Search an Element from a Linear List Implemented Through Static Array, Dynamic Array and Linked List. *International Journal of Computer Applications*, 121(3), 13-20.



- Yu, B. (2019). Research on Information Retrieval Model Based on Ontology. *EURASIP Journal on Wireless Communications and Networking*, 2019:30.
- Pradana Daniswara, D., Berliana, D., Noor Abizar, M., Imani, L. A., & Suroni, A. (2025). Analysis of Product Search Effectiveness Using Linear, Binary, and Hash Algorithms. *JSAl: Journal Scientific and Applied Informatics*, 3(3), 129-136.
- Parmar, V. P., & Kumbharana, C. K. (2015). Comparing Linear Search and Binary Search Algorithms to Search an Element from a Linear List Implemented Through Static Array, Dynamic Array and Linked List. *International Journal of Computer Applications*, 121(3), 13-20.
- Huda, S., Muhimmah, N. L., & Imani, L. A. (2025). Binary Search Algorithm for Product Name Retrieval on E-commerce Platforms. *JSAl: Journal Scientific and Applied Informatics*, 3(3), 129-136.
- Ashkiyyah, K., & Mugiarto, M. (2025). Implementation of Binary Search on Web-Based Task Management Application. *Kohesi: Jurnal Multidisiplin Saintek*, 9(5), 179-187.
- Toyib, R., & Muntahanah, A. (2023). Penerapan Algoritma Binary Search Pada Sistem Informasi Perpustakaan di SMP Negeri 4 Mauliru. *Sentimas: Seminar Nasional Penelitian dan Pengabdian Masyarakat*, 360-367.