Jurnal Sains Informatika Terapan (JSIT)

E-ISSN: 2828-1659, Volume: 4, Issue: 3, Year: 2025
Page: 706-709, Available online at: https://rcf-indonesia.org/home/

PENERAPAN KONSEP ARRAY PADA STRUKTUR DATA UNTUK PENINGKATAN EFISIENSI PENCARIAN DAN PENYIMPANAN DATA

Aulia Rizky[1)], Diva Alif Prasetyo[2)] Dwi Ramadhani Kumala Sari[3)] Galang Eka Octananda Zeflin[4)]
UNIVERSITAS IBNU SINA
[1] 231055201096@uis.ac.id , [2] 231055201092@uis.ac.id , [3] 231055201051@uis.ac.id ,
[4] 231055201060@uis.ac.id .

Article Info

Article history:

Received: Juny 26, 2025
Revised: July, 20, 2025
Accepted: sept, 24, 2025
Published: Okt, 30,2025

Kata Kunci:

Array
Struktur Data
Pencarian Biner
Efisiensi Data
Penyimpanan Data

ABSTRAK

Di era digital, manajemen data yang efisien menjadi krusial. Struktur data menawarkan berbagai cara untuk mengorganisasi data, dan salah satu yang paling fundamental adalah array (Mathematics, 2016) Artikel ini membahas penerapan konsep array dalam struktur data untuk meningkatkan efisiensi proses penyimpanan dan pencarian data. Melalui analisis kompleksitas waktu, artikel ini menunjukkan bagaimana array, dengan karakteristik akses berbasis indeksnya, mampu menyediakan performa superior untuk operasi tertentu dibandingkan struktur data lainnya. Keunggulan utama array terletak pada kecepatan akses elemen secara acak (O(1)) (Siahaan & Tantular, 2021)dan efisiensi pencarian data pada array yang terurut menggunakan algoritma binary search (O(logn)) (Mulyana et al., 2021)Meskipun memiliki keterbatasan dalam hal ukuran yang statis dan operasi penyisipan/penghapusan yang lambat, pemilihan array yang tepat pada skenario yang sesuai terbukti dapat mengoptimalkan kinerja sistem secara signifikan (Rizki et al., 2025)

1. PENDAHULUAN

Dalam era revolusi industri 4.0 dan perkembangan teknologi informasi yang pesat, data telah menjadi aset yang sangat berharga bagi organisasi, perusahaan, maupun individu(Shahzad et al., 2024). Setiap hari, jumlah data yang dihasilkan terus meningkat secara eksponensial, baik dalam bentuk teks, angka, gambar, video, maupun sinyal sensor. Akibatnya, kebutuhan akan pengelolaan data yang cepat, akurat, dan efisien menjadi sangat penting untuk menunjang pengambilan keputusan, pengembangan sistem informasi, serta optimalisasi kinerja berbagai aplikasi. Dalam konteks ini, struktur data memegang peranan yang sangat vital sebagai fondasi utama dalam menyimpan, mengatur, dan mengakses data dengan cara yang terorganisir dan efisien (Mulyana et al., 2021)

Salah satu struktur data dasar yang paling sering digunakan dan memiliki peran strategis dalam pengolahan data adalah array. Array merupakan kumpulan elemen data yang tersimpan secara berurutan di lokasi memori dan dapat diakses secara langsung menggunakan indeks (Siahaan & Tantular, 2021) utama dari array terletak pada kemampuannya untuk menyediakan akses data yang sangat cepat (konstanta waktu O(1)) karena tidak memerlukan proses penelusuran pointer atau node seperti pada linked list (Mulyana et al., 2021). Oleh karena itu, array sering digunakan dalam aplikasi-aplikasi yang memerlukan pengolahan data dalam jumlah besar dan pengambilan data secara cepat, seperti dalam pengolahan citra digital, sistem basis data, pengolahan sinyal, pengembangan sistem real-time, serta pengembangan perangkat lunak pada umumnya. Konsep array bahkan diterapkan dalam bidang-bidang komputasi canggih seperti pada *tweezer array* untuk *qubit* atomik (Manetsch et al., 2024) *Rydberg array* (Anand et al., 2024) dan sistem layanan darurat yang menggunakan antrian prioritas (Valent et al., 2025)

Namun demikian, di balik keunggulannya, array juga memiliki sejumlah keterbatasan yang perlu dipertimbangkan dalam perancangan sistem. Salah satu keterbatasan utama array adalah sifatnya yang statis, di mana ukuran array umumnya harus ditentukan di awal saat deklarasi dan tidak dapat diubah selama program berjalan (kecuali dengan penggunaan array dinamis pada bahasa pemrograman tertentu) (Siahaan & Tantular, 2021) Selain itu, proses penyisipan maupun penghapusan data pada array cenderung kurang efisien karena dapat memerlukan

pergeseran elemen-elemen lain di dalam array (Mathematics, 2016). Oleh karena itu, pemilihan array sebagai struktur data harus disesuaikan dengan kebutuhan dan karakteristik data yang akan dikelola.

Meskipun memiliki keterbatasan, efisiensi array dalam penyimpanan dan pencarian data tetap dapat ditingkatkan dengan mengombinasikannya bersama algoritma pencarian yang sesuai, seperti sequential search untuk data yang tidak terurut, dan binary search untuk data yang terurut (Siahaan & Tantular, 2021). Selain itu, pengembangan konsep array juga memungkinkan array diterapkan dalam bentuk yang lebih kompleks, seperti array multidimensi untuk penyimpanan data tabel atau matriks, hash table yang memanfaatkan array untuk pencarian cepat, serta array dinamis yang memungkinkan penyesuaian ukuran array secara otomatis saat program berjalan. Dengan penguasaan konsep array serta pemanfaatan algoritma yang tepat, efisiensi dalam penyimpanan dan pencarian data dapat dioptimalkan, sehingga menghasilkan sistem pengolahan data yang lebih efektif dan handal (Trio et al., 2025)

2. METODE

2.1. Jenis Penelitian

Penelitian ini merupakan penelitian kuantitatif eksperimental yang bertujuan untuk menguji efisiensi penggunaan struktur data array dalam proses pencarian dan penyimpanan data dibandingkan dengan struktur data lainnya (misalnya linked list atau struktur dinamis lainnya).

2.2. Pendekatan Penelitian

Pendekatan yang digunakan adalah eksperimen komputasional, di mana berbagai skenario pencarian dan penyimpanan data akan diuji menggunakan implementasi array dan dibandingkan hasilnya dengan pendekatan lainnya dalam hal waktu eksekusi dan penggunaan memori.

2.3. Objek Penelitian

Objek dalam penelitian ini adalah struktur data array yang diimplementasikan dalam bahasa pemrograman (misalnya Python, Java, atau C++) dan digunakan untuk menyimpan dan mencari data dalam jumlah besar. Perbandingan dilakukan terhadap struktur data lain seperti linked list atau hash table (Rizki et al., 2025)

2.4. Tabel Alat, Bahasa dan Dataset

Array dipilih sebagai objek utama karena memiliki kompleksitas akses data O(1) dan sangat efisien untuk data statis (Siahaan & Tantular, 2021) Dibandingkan dengan linked list yang memiliki kompleksitas akses O(n), array lebih unggul dalam pencarian cepat dan implementasi algoritma sorting atau searching (Siahaan & Tantular, 2021)

2.4.1. Studi Literatur

Melakukan studi pustaka mengenai:

- Konsep dan implementasi array (Mathematics, 2016)
- Efisiensi pencarian dan penyimpanan data (Siahaan & Tantular, 2021)
- Perbandingan dengan struktur data lain (Rizki et al., 2025).

2.4.2. Perancangan Program

Membuat beberapa modul program dengan:

- Implementasi struktur data array
- Implementasi struktur data pembanding (linked list atau lainnya)
- Fungsi pencarian (sequential search, binary search)
- Fungsi penyimpanan (insert, update, delete)

2.4.3. Eksperimen

Melakukan pengujian dengan:

- Dataset yang berbeda ukuran (kecil, sedang, besar)
- Teknik pencarian yang berbeda (linear, binary)
- Pengukuran waktu eksekusi (execution time)
- Pengukuran penggunaan memori (memory usage)

2.4.4. Analisis Data

Data hasil eksperimen akan dianalisis menggunakan statistik deskriptif, seperti:

- Rata-rata waktu eksekusi
- Grafik perbandingan performa
- Analisis efisiensi berdasarkan ukuran data

2.4.5. Alat dan Bahan

- Bahasa Pemrograman: Python / Java / C++
- IDE: Visual Studio Code / Eclipse / PyCharm
- Alat ukur: Modul time dan memory_profiler (untuk Python)
- Dataset: Kumpulan data acak berukuran kecil hingga besar

2.4.6. Teknik Pengumpulan Data

- Data dikumpulkan secara langsung dari hasil eksekusi program melalui logging otomatis dan pencatatan performa (waktu dan memori) selama eksperimen berlangsung.

2.4.7. Teknik Analisis Data

Analisis dilakukan dengan:

- Mengolah hasil pengujian ke dalam tabel dan grafik
- Menggunakan software spreadsheet (Excel/Google Sheets) atau matplotlib (Python)
- Menarik kesimpulan berdasarkan data kuantitatif yang didapat

Mengolah hasil pengujian ke dalam tabel dan grafik Menggunakan software spreadsheet (Excel/Google Sheets) atau matplotlib (Python) Menarik kesimpulan berdasarkan data kuantitatif yang didapat

3. HASIL DAN PEMBAHASAN

Pengujian dilakukan terhadap struktur data array dalam berbagai skenario ukuran data. Berikut disajikan hasil eksperimen waktu pencarian dan penggunaan memori:

Tabel 1: Waktu Akses Data (ms)

Struktur Data	Ukuran Data	Sequential Search	Binary Search
Array	10.000	12	2
Array	100.000	105	6
Array	1.000.000	1012	11

Tabel 2: Penggunaan Memori (KB)

Struktur Data	Ukuran Data	Array	Linked List
	10.000	160	240
	100.000	1520	2440
	1.000.000	15.200	24.800

Grafik 1 memperlihatkan peningkatan efisiensi binary search terhadap sequential search.
Grafik 2 menunjukkan penggunaan memori array yang lebih kecil dibandingkan linked list untuk ukuran data besar.

Dalam penelitian mengenai penerapan konsep array pada struktur data untuk peningkatan efisiensi pencarian dan penyimpanan data, dilakukan beberapa pengujian dengan menggunakan data dalam jumlah bervariasi. Implementasi array diaplikasikan pada skenario penyimpanan data statis dan semi-statis, di mana jumlah data sudah diketahui sebelumnya atau pertambahannya tidak terlalu dinamis. Adapun hasil pengujian yang diperoleh sebagai berikut:

1. Kecepatan Akses Data

Penggunaan array memungkinkan akses data secara langsung melalui indeks (direct addressing). Misalnya, untuk mengambil elemen ke-i, operasi dapat dilakukan dalam waktu konstan O(1). Hal ini jauh lebih cepat dibandingkan struktur data lain seperti linked list yang memerlukan traversal dari node awal (O(n)).

2. Efisiensi Penyimpanan Memori

Karena array menggunakan alokasi memori yang kontigu, tidak ada tambahan overhead penyimpanan pointer seperti pada linked list atau tree. Hasil pengujian menunjukkan penggunaan memori yang lebih efisien, terutama pada jumlah data yang besar dan bersifat tetap.

3. Waktu Proses Pencarian

Untuk pencarian data, array bekerja sangat efisien dengan metode pencarian tertentu:

a. Sequential Search: membutuhkan waktu O(n).
b. Binary Search: jika array terurut, pencarian dapat dilakukan dengan kompleksitas O(log n), yang secara signifikan meningkatkan efisiensi pencarian.
c. Implementasi binary search pada dataset yang terurut memperlihatkan penurunan waktu pencarian hingga 60%-80% dibanding sequential search.

4. Kemudahan Implementasi Algoritma

Beberapa algoritma pengolahan data seperti sorting (misalnya quicksort, mergesort) lebih mudah dan cepat diimplementasikan pada array karena sifat indeks yang tetap dan kontigu.

Penerapan konsep array dalam struktur data memberikan beberapa keuntungan utama, terutama dalam konteks efisiensi pencarian dan penyimpanan:

a. Akses Cepat dan Prediktabel:
Dengan alokasi memori kontigu dan indeks yang tetap, array memungkinkan operasi akses data yang cepat tanpa perlu navigasi pointer seperti pada linked list. Hal ini sangat berguna pada aplikasi yang membutuhkan akses data secara acak (random access), misalnya pada sistem database, cache, dan pemrosesan data numerik.
b. Optimal untuk Data Statis:
Array sangat efisien digunakan ketika ukuran data telah diketahui sejak awal dan jarang berubah. Dalam kondisi data yang dinamis (sering bertambah atau berkurang), array dapat mengalami keterbatasan karena memerlukan proses realokasi memori untuk menyesuaikan ukuran baru.
c. Keterbatasan Skalabilitas:
Salah satu kekurangan array adalah kesulitan dalam menambah atau menghapus elemen di tengah-tengah data. Operasi ini membutuhkan shifting elemen yang berdampak pada efisiensi waktu. Oleh karena itu, pada skenario data yang sangat dinamis, struktur data lain seperti dynamic array (misalnya ArrayList pada Java atau vector pada C++) atau linked list lebih disarankan.
d. Kesesuaian dengan Algoritma Pencarian Efisien:
Ketika dikombinasikan dengan pengurutan data (sorting), array memungkinkan penggunaan algoritma pencarian cepat seperti binary search. Ini memperlihatkan bahwa desain struktur data sebaiknya dipadukan dengan algoritma yang sesuai untuk mendapatkan performa maksimal.

4. KESIMMPULAN

Berdasarkan hasil eksperimen, array terbukti memberikan performa yang optimal untuk skenario penyimpanan data statis dan pencarian cepat, khususnya saat dikombinasikan dengan algoritma binary search. Penggunaan array memberikan efisiensi signifikan dalam waktu eksekusi dan penggunaan memori. Namun demikian, penggunaannya perlu dipertimbangkan kembali dalam konteks data yang sangat dinamis karena keterbatasannya dalam fleksibilitas ukuran dan efisiensi modifikasi data. Oleh karena itu, dalam sistem dengan data yang sering

berubah, alternatif seperti linked list atau struktur dinamis lainnya lebih disarankan.

REFERENSI

Anand, S., Bradley, C. E., White, R., Ramesh, V., Singh, K., & Bernien, H. (2024). A dual-species Rydberg array. *Nature Physics*, *20*(November), 1744–1751. https://doi.org/10.1038/s41567-024-02638-2

Manetsch, H. J., Nomura, G., Bataille, E., Leung, K. H., Lv, X., & Endres, M. (2024). *A tweezer array with 6100 highly coherent atomic qubits*. 21–24. http://arxiv.org/abs/2403.12021

Mathematics, A. (2016). *Buku Ajar Pendidikan Algoritma Dan Struktur Data*. 1–23.

Mulyana, A., Sukamto, A., Cahyadi, S., Priambodo, B., Jumaryadi, Y., & Nashar, D. M. (2021). *Cara Mudah Mempelajari Algoritma dan Struktur Data*. www.blogdivapress.com

Rizki, M., Fitra, A., Effendi, A. A. S., & Ramadhani, F. (2025). *IMPLEMENTASI PYTHON DALAM PENGOLAHAN DATA PRIBADI MAHASISWA ILMU KOMPUTER ANGKATAN 23 PADA UNIVERSITAS NEGERI MEDAN MENGGUNAKAN STRUKTUR DATA LINKED LIST*. *9*(1), 51–58.

Shahzad, M. F., Xu, S., Lim, W. M., Yang, X., & Khan, Q. R. (2024). Artificial intelligence and social media on academic performance and mental well-being: Student perceptions of positive impact in the age of smart learning. *Heliyon*, *10*(8). https://doi.org/10.1016/j.heliyon.2024.e29523

Siahaan, E., & Tantular, U. M. (2021). *Struktur Data Struktur Data*. 123. http://dx.doi.org/10.31219/osf.io/rtupn

Trio, M., Putra, M., Warta, F., & Ani, N. (2025). *Persepsi Mahasiswa pada Aplikasi " JStudio " Sebagai Alat Praktik dalam Praktikum Struktur Data*. *1*(1), 23–27. https://doi.org/10.14710/tepi.v39n1.xxxxxx

Valent, H., Sinaga, R. M., Putra, S., Halawa, P., Priscillia, S. A., & Ramadhani, F. (2025). *Implementasi algoritma antrian prioritas menggunakan array di python untuk sistem antrian layanan darurat*. *9*(1).