MENGANALISIS TINGKAT EFISIENSI STRUKTUR DATA ARRAY DAN LINKED LIST UNTUK MENGELOLA DATA MAHASISWA DARI SEGI KECEPATAN EKSEKUSI DAN PENGGUNAAN MEMORI

Nezza Anggraini Yolandari[1], M. Fikri Zulfi[2], Lastri Elisabet Butarbutar[3], Gloria Rajagukguk[4],
nezzaanggraini0@gmail.com[1], fikrizulfi5@gmail.com[2], elisabetlastri12@gmail.com[3], gloriatata11@gmail.com[4]
Universitas Negeri Medan

ABSTRAK

Dalam pengelolaan data mahasiswa, pemilihan struktur data yang tepat menjadi tantangan penting yang mempengaruhi efisiensi sistem. Penelitian ini bertujuan untuk menganalisis dan membandingkan efisiensi struktur data array dan linked list dalam hal kecepatan eksekusi, penggunaan memori, dan jumlah baris kode. Metode yang digunakan adalah eksperimen komputasi dengan analisis kuantitatif, di mana dua program yang sama dibuat untuk mengelola data mahasiswa menggunakan masing-masing struktur data. Secara keseluruhan, Array lebih cocok digunakan untuk skenario yang membutuhkan efisiensi dalam hal penambahan, pencarian, dan penampilan data, sedangkan Linked List lebih ideal untuk situasi yang memprioritaskan penghapusan data secraa cepat. Namun, keetika data yang dikelola dalam jumlah besar, baik Array maupun Linked List akan membutuhkan lebih banyak memori. Kesimpulannya, pemilihan struktur data yang tepat sangat penting untuk meningkatkan efisiensi dalam aplikasi pengelolaan data akademik.
Kata Kunci: Struktur Data, Array dan Linked List, Efisiensi Pengelolaan Data

ABSTRACT

In managing student data, the selection of the appropriate data structure poses a significant challenge that affects system efficiency. This study aims to analyze and compare the efficiency of array and linked list data structures in terms of execution speed, memory usage, and code complexity. The method employed is computational experimentation with quantitative analysis, where two identical programs are created to manage student data using each data structure. Overall, arrays are more suitable for scenarios that require efficiency in data addition, searching, and display, while linked lists are ideal for situations that prioritize quick data deletion. However, when managing large amounts of data, both arrays and linked lists will require more memory. In conclusion, the choice of the right data structure is crucial for enhancing efficiency in academic data management applications.
***Keyword**: Data Stucture, Array and Linked List, Data Management, Efficiency*

PENDAHULUAN

Struktur data adalah cara untuk menyimpan, mengatur, dan mengelola data secara efisien sehingga dapat digunakan dengan cara yang optimal. Dalam ilmu komputer, struktur data merupakan elemen penting yang memungkinkan pengembang perangkat lunak untuk mengelola data dengan cara yang terorganisir, baik untuk penyimpanan maupun pemrosesan. Pemilihan struktur data yang tepat memengaruhi performa program dalam hal kecepatan, efisiensi memori, dan kemudahan implementasi.

Struktur data menyimpan atau merepresentasikan data di dalam computer agar bisa dipakai secara efisien sedangkan data adalah representasi dari fakta dunia nyata. Fakta atau keterangan tentang kekayaan yang disimpan, direkam atau dipresentasikan dalam bentuk tulisan, suara, gambar, sinyal atau symbol.

Struktur data dan algoritma merupakan bagian yang penting, terintegrasi, dan saling mendukung dalam membangun program aplikasi yang baik dan efisien. Memberikan perhatian lebih terhadap hal ini dapat memberikan hasil yang sangat penting, terutama dalam meningkatkan kecepatan akses dan memberikan wawasan serta panduan tentang cara meningkatkan pemecahan masalah dan menemukan solusi yang lebih baik. Algoritma sangat penting dan merupakan kunci yang menjadi dasar pemograman meskipun algoritma yang berbeda dapat digunakan untuk menyelesaikan masalah yang sama, memilih algoritma dengan strategi yang seusai dan efisiensi akan mempercepat proses eksekusi. Algoritma dan struktur data memiliki hubungan yang kuat dan bekerja sama untuk mencapai hasil optimal dan memungkinkan solusi. (Shandy, dkk, 2024).

Secara umum, struktur data dibagi menjadi dua kategori utama yaitu struktur data linear dan struktur data non-linear. Masing-masing kategori ini memiliki berbagai jenis struktur data yang berbeda. Struktur data linear adalah struktur di mana elemen data diatur secara berurutan satu demi satu. Beberapa contoh umum dari struktur data linear adalah array yang menyimpan elemen-elemen yang memiliki tipe data yang sama dalam blok memori yang berdekatan, linked list yang setiap elemen (disebut node) menyimpan data dan sebuah referensi (atau pointer) ke elemen berikutnya

Dalam pengelolaan data, pemilihan struktur data yang tepat memainkan peran penting dalam menentukan efisiensi sistem Struktur data linear, seperti Array dan Linked List sering digunakan dalam berbagai aplikasi pemograman untuk memanipulasi dan menyimpan data. Masing-masing struktur memiliki karakteristik dan kelebihan tersendiri yang dapat mempengaruhi kinerja sistem dalam hal kecepatan ases data, penggunaan memori, dan kompleksitas implementasi kode.

Pada konteks pengelolaan data mahasiswa, di mana data sering diakses, diperbarui, dan dianalisis, efisiensi struktur data yang digunakan menjadi krusial. Kecepatan akses menjadi faktor penting ketika melakukan pencarian nilai mahasiswa tertentu, penambahan atau penghapusan data baru, serta penyusunan data secara dinamis. Selain itu, dengan terbatasnya sumber daya perangkat keras, seperti kapasitas memori, pemilihan struktur data yang hemat memori menjadi pertimbangan signifikan. Tidak kalah penting, jumlah baris program atau kompleksitas kode dalam mengimplementasikan struntur data juga menjadi perhatian.

Penelitian ini bertujuan untuk menganalisis dan membandingkan tingkat efisiensi dari struktur data array dan linked list dalam konteks pengelolaan data mahasiswa. Penelitian ini akan menilai efisiensi dari tiga aspek utama, yaitu kecepatan eksekusi, penggunaan memori, dan jumlah baris program yang diperlukan untuk implementasi. Dengan demikian, penelitian ini diharapkan dapat memberikan panduan yang jelas bagi pengemembang perangkat lunak, khususnya dalam memilih struktur data yang paling efisien untuk aplikasi akademik dan sistem pengelolaan data mahasiswa.

METODE PENELITIAN

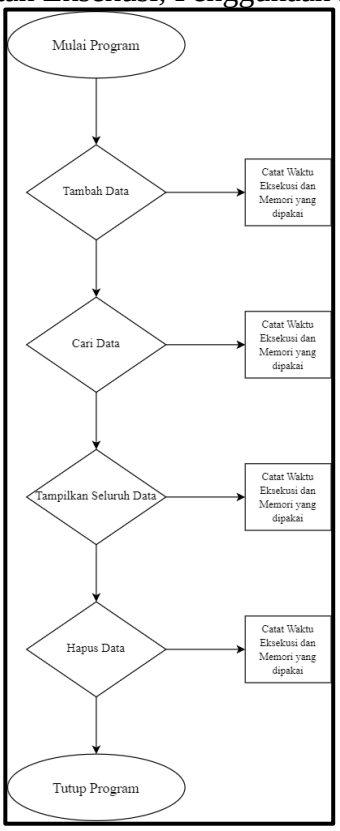
Penelitian ini menggunakan metode penelitian eksperimen komputasi dengan analisis komparatif melalui pengukuran kuantitatif untuk mengevaluasi efisiensi waktu dan penggunaan memori, dengan tujuan membandingkan efisiensi struktur data array dan linked list berdasarkan kriteria efisiensi tersebut.

Populasi penelitian berupa program komputer yang mengelola data mahasiswa menggunakan struktur data dengan sampel yang digunakan terdiri dari 2 program yang masing-masing menggunakan Array dan Linked List untuk mengelola data mahasiswa. Penelitian ini menggunakan Bahasa pemrograman Python dan implementasi tkinter sebagai GUI dengan Time Complexity dan Memory Profiler untuk mengukur waktu eksekusi dan

penggunaan memori.

Prosedur pengumpulan data pada penelitian meliputi membuat 2 program yang sama untuk mengelola data mahasiswa menggunakan masing- masing struktur data Array dan Linked List, melakukan pengujian pada setiap program dengan berbagai ukuran data yang sama, mencatat waktu eksekusi setiap program (operasi menambah data, menghapus data, mencari data, dan menampilkan seluruh data), mengukur penggunaan memori saat program dijalankan yang paling efisien.

Penelitian mencakup beberapa teknik analisis data yaitu Analisis kecepatan menggunakan runtime atau time complexity, Analisis penggunaan memori menggunakan profiler, Analisis baris program untuk melihat seberapa kompleks kode yang diperlukan, dan melakukan pembandingan hasil tersebut untuk setiap program struktur data dan menentukan yang paling efisien dalam mengelola data nilai mahasiswa. Variabel pada penelitian berupa variable Independen yaitu jenis struktur data Array dan Linked List dan variable dependen yaitu Kecepatan Eksekusi, Penggunaan memori, jumlah baris kode.



Gambar 1. Alur Pengumpulan Data

Dalam penelitian ini peneliti menggunakan laptop dengan spesifikasi perangkat keras prosesor dari Intel yaitu prosesor Intel Core i5-1235U yang merupakan salah satu prosesor Intel generasi 12 untuk mobile processor. Dengan spesifikasi lengkapnya adalah dari Inti CPU 10, jumlah thread 12 dengan base speed 1,3 Ghz, cache L1 928KB, cache L2 6,5MB, dan cache L3 12 MB.

HASIL DAN PEMBAHASAN

Pengujian dilakukan terhadap dua struktur data yaitu Array dan Linked List pada dua skala data yang berbeda, yaitu 10.000 dan 20.000 data, dengan lima kali percobaan untuk setiap operasi dasar pada kedua data tersebut meliputi penambahan, penghapusan, pencarian, dan penampilan data. Analisis dilakukan berdasarkan waktu eksekusi dan penggunaan memori.

Tabel 1. Hasil Pengujian Array

PROGRAM STRUKTUR DATA **ARRAY**												
Skala	Operasi Dasar	Percobaan 1		Percobaan 2		Percobaan 3		Percobaan 4		Percobaan 5		Jumlah Baris
		Waktu (detik)	Memori (KB)	Waktu (detik)	Memori (KB)	Waktu (detik)	Memori (KB)	Waktu (detik)	Memori (KB)	Waktu (detik)	Memori (KB)	
10.000 data	Menambah	0,015523	942,5039	0,015742	942,5039	0,015527	942,5039	0,018112	942,5039	0,015743	942,5039	204
	Menghapus	0,094267	46,83593	0,102011	46,83593	0,109993	46,83593	0,109851	46,83593	0,10981	46,83593	
	Mencari	0	0,101562	0	0,101562	0,015453	0,101562	0,006268	0,101562	0,002517	0,101562	
	Menampilkan	0,024722	1,125	0,015437	0,155273	0,023	1,117188	0,020099	1,115234	0,031176	1,115234	
20.000 data	Menambah	0,024359	1887,66	0,031369	1887,66	0,015899	1887,66	0,010603	1887,66	0,015429	1887,66	
	Menghapus	0,393183	95,08593	0,400686	95,08593	0,361709	95,08593	0,392466	95,08593	0,408163	95,08593	
	Mencari	0,01554	0,101562	0,031054	0,101562	0,014516	0,101562	0,015439	0,101562	0,015752	0,101562	
	Menampilkan	0,046683	1,125	0,046909	1,126953	0,031258	1,116211	0,04725	1,120117	0,046899	1,115234	

Pada skala 10.000 data, pada operasi menambah data, waktu eksekusi konsisten di 0,0155 detik. Penggunaan memori juga stabil di angka 942,5039 KB. Operasi penambahan data pada array untuk 10.000 data cukup efisien dan stabil dari segi waktu dan memori. Penambahan data tidak mengalami overhead signifikan saat beroperasi dengan jumlah data yang relatif. Pada operasi menghapus data, waktu eksekusi berada diantara 0,094 hingga 0,109 detik, dengan penggunaan memori konstan di 46,83593 KB. Waktu penghapusan data lebih lama dibandingkan penambahan data disebabkan pergantian data data array setelah penghapusan. Tetapi penggunaan memori yang rendah menunjukan efisiensi memori. Pada operasi mencari data, waktu eksekusi percobaan pertama dan ke-2 0 detik namun percobaan seterusnya berkisar 0,001562 detik. Penggunaan memori rendah konstan di 0,101562 KB. Hasil pencarian cepat terutama percobaan awal, mungkin disebabkan pencarian melalui indeks yang efisien. Pada operasi menampilkan data, waktu eksekusi stabil antara 0,023 hingga 0,024 detik, dengan penggunaan memori rata – rata 0,9255858 KB.

Pada skala 20.000 data, pada operasi menambah data konstan di 1887,66 KB penggunaan memori dan waktu eksekusi rata – rata 0,016 detik. Karena ukuran array lebih besar memerlukan waktu lebih lama untuk alokasi penambahan data. Meskipun ada peningkatan waktu ketika jumlah data meningkat, kinerja array masih efisien dengan penggunaan memori yang juga semakin meningkat. Pada operasi menghapus data, penggunaan memori konstan di 95,08593 KB dan waktu yang diperlukan berkisar 0,4 detik. Ukuran data yang semakin besar memerlukan waktu lebih lama ini menjadi kelemahan array dalam operasi menghapus data. Pada operasi mencari data, penggunaan memori konstan di 0,101562 KB dengan waktu eksekusi rata - rata 0,01822 detik. Pencarian data di array tetap efisien dan tidak terpengaruh dari meningkatnya data. Pada operasi menampilkan data, membutuhkan waktu di 0,4 detik dan penggunaan memori yang stabil sekitar 1,120703 KB. Semakin meningkat data maka waktu eksekusi dan penggunaan memori saat menampilkan data juga meningkat. Walaupun waktu eksekusi bertambah kinerja penggunaan memori dalam menampilkan data tetap efisien.

Pada skala 10.000 data, semua operasi berjalan dengan waktu eksekusi yang cepat dan penggunaan memori yang efisien serta stabil. Namun, ketika skala data meningkat menjadi 20.000, operasi tetap cukup efisien, meskipun terjadi peningkatan waktu eksekusi. Sebaliknya, operasi menghapus data menunjukkan penurunan efisiensi yang signifikan, dengan waktu eksekusi yang jauh lebih lama akibat penambahan data. Dari segi penggunaan memori, terjadi peningkatan yang cukup signifikan pada skala yang lebih besar, namun tetap stabil pada setiap percobaan. Perbandingan ini menunjukkan bahwa struktur data array efisien untuk operasi penambahan, pencarian, dan penampilan, namun kurang optimal dalam penghapusan data pada data yang besar.

Tabel 2. Hasil Pengujian Linked List

PROGRAM STRUKTUR DATA **LINKED LIST**												
Skala	Operasi Dasar	Percobaan 1		Percobaan 2		Percobaan 3		Percobaan 4		Percobaan 5		Jumlah Baris
		Waktu (detik)	Memori (KB)	Waktu (detik)	Memori (KB)	Waktu (detik)	Memori (KB)	Waktu (detik)	Memori (KB)	Waktu (detik)	Memori (KB)	
10.000 data	Menambah	1,428282	1484,714	1,47705	1484,714	1,470206	1484,714	1,452591	1484,714	1,613446	1484,714	257
	Menghapus	0,015648	0,484375	0,015669	0,484375	0,01352	0,484375	0,015648	0,484375	0,015452	0,484375	
	Mencari	0	0,476562	0,1554	0,476562	0,01996	0,476562	0,009748	0,054688	0,014139	0,476562	
	Menampilkan	0,031169	1158,927	0,015739	1158,927	0,031965	1158,927	0,015624	1158,927	0,024899	1178,46	
20.000 data	Menambah	6,300822	2969,089	6,217787	2963,808	6,014008	2969,089	5,824661	2963,808	5,97327	2963,808	
	Menghapus	0,015732	0,484375	0,015015	0,054688	0,014148	0,484375	0,015664	0,054688	0,014109	0,484375	
	Mencari	0,015514	0,476562	0,015751	0,054688	0,015647	0,054688	0,003527	0,00956	0,015122	0,054688	
	Menampilkan	0,062403	2377,524	0,004736	2297,966	0,046983	1947,815	0,03136	1926,854	0,003123	2337,031	

Pada skala 10.000 data, pada operasi penambahan data, penggunaan memori konstan di 1484,71 KB dan memerlukan waktu cukup lama berkisar 1,4 detik karena harus mengalokasikan memori baru untuk setiap data yang ditambahkan. Pada operasi penghapusan data, penggunaan memori konstan di 0,48438 KB dengan waktu eksekusi sangat cepat sekitar 0,01565 hingga 0,01567 detik. Linked list unggul dalam penghapusan karena hanya perlu mengubah pointer. Pada operasi mencari data, penggunaan memori konstan di 0,47656 KB dengan waktu yang digunakan pada percobaan pertama 0 detik dan percobaan seterusnya meningkat hingga 0,1554 detik. Menampilkan data pada linked list relative efisien meskipun lebih lambat dengan struktur data lainnya. Pada operasi menampilkan data, penggunaan memori konstan di 1158,93 KB. Semua operasi menggunakan memori yang konstan disetiap percobaan.

Pada skala 20.000 data, Pada operasi penambahan waktu eksekusi meningkat menjadi sekitar 6 detik dengan penggunaan memori bervariasi 2969,09 dan 2963,81 KB. Penambahan data pada linked list membutuhkan waktu jauh lebih lama ketika jumlah data meningkat karena memerlukan waktu untuk mengalokasikan memori. Pada operasi menghapus data, penggunaan memori berkisar di 0,05 hingga 0,48 KB dan waktu penghapusan stabil di 0,01 detik. Pada operasi mencari data, waktu pencarian stabil di 0,01 detik dan penggunaan memori lebih tinggi di 0,00956 hingga 0,47656 KB. Pada operasi menampilkan data, waktu yang dibutuhkan meningkat menajdi 0,03136 hingga 0,0624 detik dengan penggunaan memori yang lebih tinggi di 1926,85 hingga 2377,52 KB. Penampilan data semakin lambat ketika data meningkat meskipun tetap efisien dalam penggunaan memori.

Secara keseluruhan, linked list bekerja sangat baik untuk operasi penghapusan dan pencarian data di kedua skala data, tetapi performa menurun dalam operasi penambahan dan menampilkan data seiring bertambahnya jumlah data yang dikelola, terutama pada skala data yang lebih besar.

Pada struktur data Array, operasi penambahan data menggunakan waktu relative lebih cepat dibandingkan Linked List. Pada operasi penghapusan data, array lebih lambat dibandingkan Linked List karena memiliki struktur yang lebih dinamis. Untuk operasi pencarian, Array cenderung lebih cepat sedangkan linked list membutuhkan waktu yang lebih siginifikan. Operasi menampilkan data pada Linked List lebih lambat dibandingkan Array, dimana Linked List membutuhkan waktu lebih lama karena perlu menelusuri seluruh data.

Pada penggunaan memori, Array menggunakan memori yang lebih efisien dan konsisten, sedangkan Linked List menggunakan lebih banyak memori yang semakin meningkat ketika data bertambah karena Linked List menyimpan data beserta pointer untuk setiap data sehingga menggunakan lebih banyak ruang penyimpanan.

Array lebih baik dalam hal efisiensi waktu untuk operasi penambahan, pencarian, dan penampilan data, namun pada penghapusan data lebih lambat. Linked List unggul dalam

operasi penghapusan, tetapi memakan lebih banyak waktu pada operasi penambahan dan menampilkan serta menggunakan memori yang lebih banyak. Untuk data dalam skala besar, penggunaan Linked List dan Array dapat meningkatkan konsumsi penggunaan memori.

KESIMPULAN

Dari hasil penelitian yang membandingkan efisiensi struktur data Array dan Linked List dalam pengelolaan data mahasiswa, dapat disimpulkan bahwa dalam hal efisiensi waktu, struktur data Array lebih unggul dalam hal penambahan, pencarian, dan penampilan data, dengan waktu eksekusi yang lebih cepat dan konsisten pada skala data yang lebih kecil dan besar. Sedangkan struktur data Linked List memiliki performa yang lebih baik pada operasi penghapusan data, karena hanya memerlukan perubahan pointer. Namun, waktu eksekusi Linked List lebih lambat untuk operasi penambahan dan menampilkan data dibandingkan Array. Dalam hal penggunaan memori, Array lebih efisien dalam penggunaan memori, dengan konsumsi memori yangg stabil lebih rendah dibandingkan Linked List. Sebaliknya, Linked List menggunakan lebih banyak memori karena setiap elemen menyimpan pointer tambahan. Ketika skala data meningkat, penggunaan memori Linked List juga bertambah secara signifikan. Dalam hal kompleksitas kode, implementasi Array membutuhkan lebih sedikit baris kode, membuatnya lebih sederhana dan edisien dari segi kompleksitas kode. Linked List memerlukan lebih banyak baris kode karena struktur yang lebih dinamis dan kompleksitas dalam pengelolaan pointer. Secara keseluruhan, Array lebih cocok digunakan untuk skenario yang membutuhkan efisiensi dalam hal penambahan, pencarian, dan penampilan data, sedangkan Linked List lebih ideal untuk situasi yang memprioritaskan penghapusan data secraa cepat. Namun, keetika data yang dikelola dalam jumlah besar, baik Array maupun Linked List akan membutuhkan lebih banyak memori.

Saran

Untuk penelitian selanjutnya, disarankan untuk menggunakan jumlah data yang lebih besar, seperti 50.000 atau 100.000 data, agar dapat menguji bagaimana efisiensi struktur data Array dan Linked List pada skala yang lebih tinggi. Selain itu, peneliti selanjutnya juga dapat mempertimbangkan penggunaan data yang lebih kompleks, seperti objek dengan banyak atribut atau data yang saling terkait, untuk memahami bagaimana kedua struktur data ini berfungsi dalam skenario yang lebih mendekati aplikasi dunia nyata. Penelitian ini juga dapat diperluas dengan menerapkan variasi dari Array dan Linked List, seperti Dynamic Array atau Doubly Linked List, untuk melihat potensi pengingkatan efisiensi. Dengan menggunakan data yang lebih besar dan kompleks, penelitian di masa depan akan memberikan wawasan yang lebih luas dan mendalam terkait kinerja optimal dalam pengelolaan data.

DAFTAR PUSTAKA

Ayam Bakar Pak Mul Berbasis Mobile Android. Jurnal Riset dan Aplikasi Mahasiswa Informatika (JRAMI)

Charisma, T, S, dkk. (2023). Buku Ajar Pemrograman Struktur Data. Sumedang: CV. Mega Press Nusantara

Computer Sciences and Engineering. Vol. 7 No. 5

Nuranggang, L., & Tri Y. A. (2020). Perancangan Sistem Aplikasi Pemesanan Makanan Di

Pastima, S, dkk. (2020). Pengantar Konsep Struktur Data. Padang: Pustaka Galeri Mandiri

Renuka, D, K, (2019). Analysis of Arraylist and Linked list. Internasional Journal of

Rita, K, dkk. (2023). Pengantar Ilmu Komputer. Jambi: Pt. Sonpedia Publishing Indonesia.

Shandy, L, dkk. (2024). Pelatihan Mengenalkan Struktur Data Pada SMK Negeri 2 Cimahi. Jurnal Puan Indonesia. Vol. 6 No. 133