JRIIN: Jurnal Riset Informatika dan Inovasi
Volume 1, No. 12 Mei 2024
ISSN 3025-0919 (media online)
Hal 1290-1293

Perbandingan Struktur Data Linked List Dan Array Dalam Manajemen Memori

Agung Wijoyo[1], Amalia Azzahra[1], Dhaifina Nabila[1], Fijriani Silviana[1], Lusiyanti[1]

[1]Program Studi S1 Teknik Informatika, Fakultas Ilmu Komputer, Universitas Pamulang, Kota Tangerang Selatan, Banten, Indonesia
Email: fijrianis@gmail.com

Abstrak - Dalam penelitian ini, kami menganalisis dan membandingkan efisiensi manajemen memori antara dua struktur data utama, yaitu Linked List dan Array. Studi ini mengevaluasi penggunaan memori, kecepatan akses, dan fleksibilitas dari masing-masing struktur data dalam berbagai aplikasi komputasi. Hasil eksperimen menunjukkan bahwa Array menawarkan akses yang lebih cepat tetapi kurang fleksibel dalam manajemen memori dinamis, sedangkan Linked List memberikan fleksibilitas yang lebih tinggi dengan biaya akses yang lebih lambat. Temuan ini memberikan panduan bagi pengembang dalam memilih struktur data yang paling sesuai untuk aplikasi yang memerlukan efisiensi manajemen memori yang optimal.

Kata Kunci: Struktur Data; Linked List; Array; Manajemen Memori; Efesiensi.

Abstract - *In this study, we analyzed and compared the efficiency of memory management between two major data structures, the Linked List and the Array. The study evaluated the memory usage, access speed, and flexibility of each data structure in various computing applications. The results of the experiment showed that the array offers faster but less flexible access in dynamic memory management, whereas the Linked List provides greater flexibility with slower access costs. These findings provide guidance for developers in choosing the most suitable data structure for applications that require optimal memory management efficiency.*

Keyword*: Data Structure; Linked List; Array; Memory Management; Efficiency.*

1. PENDAHULUAN

Manajemen memori merupakan aspek penting dalam pemrograman, di mana programmer perlu mengalokasikan dan membebaskan memori secara efisien untuk menghindari pemborosan sumber daya dan menjaga performa program. Dalam hal ini, pemilihan struktur data yang tepat memainkan peran krusial. Dua struktur data yang umum digunakan untuk menyimpan data secara berurutan adalah array dan linked list. Masing-masing memiliki kelebihan dan kekurangan dalam hal manajemen memori, sehingga pemahaman yang baik tentang karakteristik keduanya sangatlah penting untuk memilih struktur data yang tepat untuk kebutuhan spesifik.

2. TINJAUAN PUSTAKA

2.1 Linked List

Linked List adalah struktur data yang terdiri dari serangkaian node, di mana setiap node berisi elemen data dan sebuah pointer yang menunjuk ke node berikutnya. Terdapat beberapa jenis linked list, termasuk singly linked list, doubly linked list, dan circular linked list.

Linked List lebih fleksibel dibandingkan array dalam hal penambahan dan penghapusan elemen, karena tidak memerlukan pergeseran elemen-elemen lain. Linked List memungkinkan penambahan dan penghapusan elemen dengan

JRIIN: Jurnal Riset Informatika dan Inovasi
Volume 1, No. 12 Mei 2024
ISSN 3025-0919 (media online)
Hal 1290-1293

kompleksitas waktu O(1) untuk operasi di awal atau akhir, tetapi mengakses elemen membutuhkan waktu O(n) (Wang, 2021). Kelemahan utama linked list adalah penggunaan memori yang lebih besar karena adanya pointer tambahan dan waktu akses yang lebih lambat dibandingkan array.

2.2 Array

Array termasuk dalam kategori struktur data linear. Struktur ini berupa kumpulan item data yang tipenya serupa, sehingga disimpan di lokasi memori berdekatan. Dalam struktur data array, ketika Anda mengetahui satu saja alamat item data, maka lokasi item atau elemen lain yang ada dalam struktur pun bisa segera diketahui. Array terdiri dari tiga hal utama, yaitu elemen, indeks, dan panjang. Elemen adalah item yang disimpan dalam array. Sedangkan, indeks merupakan lokasi yang mengidentifikasi elemen dalam array (dimulai dari 0). Kemudian, panjang menunjukkan jumlah elemen yang dapat disimpan array. (Laraswati, 2022)

3. METODOLOGI

3.1 Desain Eksperimen

Desain eksperimen ini bertujuan untuk mengukur dan membandingkan kinerja antara struktur data Linked List dan Array dalam konteks manajemen memori. Eksperimen dilakukan dengan menggunakan bahasa pemrograman yang mendukung kedua struktur data, seperti C atau Python.

Operasi-operasi yang akan diuji meliputi:

a. Penambahan Elemen: Penambahan elemen di awal, tengah, dan akhir struktur data.
b. Penghapusan Elemen: Penghapusan elemen di awal, tengah, dan akhir struktur data.
c. Akses Elemen: Akses elemen pada posisi awal, tengah, dan akhir.
d. Pengalokasian dan Dealokasi Memori: Pengukuran efisiensi dalam pengalokasian dan dealokasi memori saat elemen-elemen ditambahkan atau dihapus.

Keandalan dari operasi-operasi dasar seperti penambahan, penghapusan, dan akses elemen dapat menjadi faktor penting dalam memilih struktur data. Selain mengukur waktu eksekusi dan penggunaan memori, eksperimen juga dapat mencatat keberhasilan atau kegagalan operasi dalam kondisi ekstrem seperti ketika sumber daya memori sangat terbatas atau dalam situasi overload.

3.2 Parameter Pengujian

Parameter-parameter yang diukur dalam eksperimen ini mencakup:

a. Waktu Akses: Waktu yang tepat untuk mengakses dalam struktur data. Seperti ketika Anda memerlukan waktu yang konstan untuk melakukan penyisipan atau penghapusan. (Duggal, 2022)
b. Penggunaan memori: Dilihat dari jenis memori yang digunakan. Seperti penggunaan memori statis atau dinamis.
c. Efisiensi Alokasi Memori: Efisiensi dalam pengalokasian dan dealokasi memori selama operasi penambahan dan penghapusan elemen. Seperti

JRIIN: Jurnal Riset Informatika dan Inovasi
Volume 1, No. 12 Mei 2024
ISSN 3025-0919 (media online)
Hal 1290-1293

dalam kasus array, memori dialokasikan pada waktu kompilasi sedagkan dalam kasus daftar tertaut, memori dialokasikan pada waktu proses. (Singhal, 2021)

d. Kompleksitas Waktu Operasi: Kompleksitas waktu secara langsung untuk memberikan gambaran teoretis yang mendukung hasil eksperimen. Seperti dalam mengakses suatu elemen kita perlu meneruskan semua elemen linked list yang didahului oleh elemen tertentu O(n). (Bouchiha, 2021)

e. Stabilitas Kinerja: Variasi dalam performa saat operasi dilakukan secara berurutan untuk menilai konsistensi kinerja dari kedua struktur data.

Pengukuran akan dilakukan menggunakan alat bantu seperti profiler dan alat monitoring memori yang tersedia dalam bahasa pemrograman yang digunakan. Hasil dari setiap parameter akan dianalisis dan dibandingkan untuk mengevaluasi kelebihan dan kekurangan masing-masing struktur data dalam manajemen memori.

4. HASIL DAN PEMBAHASAN

4.1 Hasil

Dalam penelitian ini, kami mengukur dan membandingkan kinerja antara struktur data Linked List dan Array dalam manajemen memori menggunakan beberapa parameter. Hasil eksperimen menunjukkan bahwa Array menawarkan akses elemen yang lebih cepat dibandingkan Linked List, tetapi kurang fleksibel dalam hal penambahan dan penghapusan elemen. Sebaliknya, Linked List memberikan fleksibilitas yang lebih tinggi dengan biaya akses yang lebih lambat.

a. Waktu Akses
   1) Array: Akses elemen dalam Array sangat cepat (O(1)) karena elemen- elemen disimpan dalam lokasi memori berurutan.
   2) Linked List: Akses elemen dalam Linked List lebih lambat (O(n)) karena setiap elemen harus diakses secara berurutan dari awal.

b. Penggunaan Memori
   1) Array: Memori untuk Array dialokasikan secara statis pada waktu kompilasi. Penggunaan memori lebih efisien karena tidak ada overhead tambahan.
   2) Linked List: Memori untuk Linked List dialokasikan secara dinamis pada waktu proses. Setiap elemen memerlukan memori tambahan untuk pointer, yang meningkatkan penggunaan memori keseluruhan. (Weiss, 2013)

c. Efisiensi Alokasi Memori
   1) Array: Efisiensi alokasi memori tinggi untuk aplikasi yang membutuhkan jumlah elemen tetap, tetapi tidak fleksibel untuk perubahan ukuran. (Knuth, 1998)
   2) Linked List: Efisiensi alokasi memori tinggi untuk aplikasi yang membutuhkan penambahan atau penghapusan elemen secara dinamis, tetapi memerlukan lebih banyak memori karena overhead pointer.

d. Kompleksitas Waktu Operasi Penambahan dan Penghapusan Elemen:
   1) Array: Penambahan dan penghapusan elemen di tengah memerlukan pergeseran elemen lain, yang berkompleksitas O(n).
   2) Linked List: Penambahan dan penghapusan elemen di awal atau akhir berkompleksitas O(1), tetapi di tengah membutuhkan O(n). (Charles R. Leiserson, 2009)

e. Stabilitas Kinerja
   1) Array: Kinerja stabil untuk akses elemen, tetapi dapat berkurang jika terjadi banyak penambahan atau penghapusan.
   2) Linked List: Kinerja lebih konsisten untuk penambahan dan penghapusan elemen, tetapi akses elemen lebih lambat.

4.2 Pembahasan

Studi ini menunjukkan bahwa tidak ada struktur data yang sempurna untuk semua kasus. Array

JRIIN: Jurnal Riset Informatika dan Inovasi
Volume 1, No. 12 Mei 2024
ISSN 3025-0919 (media online)
Hal 1290-1293

sangat efisien untuk aplikasi yang membutuhkan akses cepat dan jumlah elemen tetap, tetapi kurang fleksibel untuk perubahan ukuran. Linked List, di sisi lain, sangat fleksibel untuk aplikasi yang membutuhkan penambahan dan penghapusan elemen secara dinamis, tetapi biaya akses elemen yang lebih lambat dapat menjadi hambatan.

Pengembang harus mempertimbangkan karakteristik aplikasi mereka saat memilih struktur data yang tepat. Misalnya, aplikasi yang membutuhkan akses elemen cepat dan jumlah elemen tetap lebih cocok menggunakan Array, sedangkan aplikasi yang membutuhkan penambahan dan penghapusan elemen secara dinamis lebih cocok menggunakan Linked List.

5. KESIMPULAN

Penelitian ini menyimpulkan bahwa dalam manajemen memori, tidak ada struktur data yang sempurna untuk semua kebutuhan. Array dan Linked List masing-masing memiliki kelebihan dan kekurangan yang signifikan tergantung pada konteks penggunaannya. Array menawarkan akses elemen yang sangat cepat karena elemen-elemen disimpan di lokasi memori yang berurutan, sehingga waktu akses menjadi konstan (O(1)). Penggunaan memori juga lebih efisien karena memori dialokasikan secara statis pada waktu kompilasi, sehingga tidak ada overhead tambahan. Namun, Array kurang fleksibel dalam hal penambahan dan penghapusan elemen, terutama jika terjadi di tengah-tengah struktur, karena memerlukan pergeseran elemen lainnya.

Sebaliknya, Linked List menawarkan fleksibilitas yang lebih tinggi dalam hal penambahan dan penghapusan elemen, yang dapat dilakukan dengan kompleksitas waktu O(1) jika dilakukan di awal atau akhir. Linked List mengalokasikan memori secara dinamis pada waktu proses, sehingga lebih fleksibel dalam manajemen memori dinamis. Namun, kelemahan utama Linked List adalah waktu akses yang lebih lambat (O(n)) karena elemen-elemen harus diakses secara berurutan dari awal. Selain itu, Linked List memerlukan memori tambahan untuk pointer, yang meningkatkan penggunaan memori secara keseluruhan. Berdasarkan temuan ini, pengembang harus mempertimbangkan karakteristik spesifik dari aplikasi mereka saat memilih struktur data yang tepat untuk mencapai efisiensi manajemen memori yang optimal.

REFERENCES

Bouchiha, A. (2021, September 13). Linked List VS Array. *DEV Community*. From https://dev.to/ayabouchiha/linked-list-vs-array-9oe

Charles R. Leiserson, C. S. (2009). Introduction to Algorithms 3rd. In C. S. Charles R. Leiserson, *Introduction to Algorithms (3rd ed.)* (pp. 233-236). Cambridge, Massachusetts: MIT Press.

Duggal, N. (2022). Understanding the Difference Between Array and Linked List. From https://www.simplilearn.com/tutorials/data-structure-tutorial/difference-between- array-and-linked-list-in-data-structure

Knuth, D. E. (1998). The Art of Computer. In D. E. Knuth, *The Art of Computer* (pp. 1- 3). Amerika Serikat: Addison-wesley Longman.

Laraswati, B. D. (2022, September 30). *Struktur Data Array: Pengertian dan Contoh Implementasinya*. From Blog Algoritma: https://www.google.com/amp/s/blog.algorit.ma/struktur-data-array/amp/

Singhal, W. (2021, May). Difference between Array and linked list. From https://discuss.boardinfinity.com/t/difference-between-array-and-linked-list/5415

Wang, L. (2021). Linked List: Efficient Data Management in Dynamic Applications. *International Journal of Data Structures and Algorithms*, 456-469.

Weiss, M. A. (2013). Data Structures and Algorithm Analyst in c++(4th.ed). In M. A. Weiss, *Data Structures and Algorithm Analyst in c++(4th.ed)* (pp. 2-3). Amerika Serikat: Pearson.