Jurnal Sains Informatika Terapan (JSIT)

E-ISSN: 2828-1659, Volume: 2, Issue: 2, Year: 2025
Page: 441-444, Available online at: https://rcf-indonesia.org/home/

ANALISIS STRUKTUR DATA *LINKED LIST* DALAM PENGOLAHAN DATA MAHASISWA

Maria Triani Mbejo[1)], Ledi Alde Nopa[2)], Jingga Saftina Putri[3)], M. Risky[4)]
Universitas Ibnu Sina
[1] 231055201001@uis.ac.id
[2] 231055201072@uis.ac.id
[3] 231055201040@uis.ac.id
[4] 231055201090@uis.ac.id

Article Info

Article history:

Received: Juny 11, 2025
Revised: July 10, 2025
Accepted: July 30, 2025
Published: Agust 27, 2025

Keywords:

Linked Lis;t
Array;
Node;
Nanosecond;

ABSTRAK

Pemilihan struktur data yang tepat penting untuk proses pengolahan manajemen data mahasiswa. Dalam makalah ini, struktur data baru yang disebut *Linked list* diusulkan untuk mengelola perubahan data yang sering terjadi seperti data siswa. *Linked list* digunakan untuk menyimpan, mengatur, dan mengelola data secara efisien sehingga dapat digunakan dengan cara yang optimal. Untuk itu, penulis ingin mengevaluasi penggunaan struktur data *Linked List* dalam aplikasi manajemen data Siswa sederhana (tambah, hapus, cari, tampilkan) yang dinamis dengan memanfaatkan bahasa pemrograman Java. Uji coba menunjukkan bagaimana *linked list* dapat diterapkan untuk mengelola Perubahan Data yang sering terjadi seperti Data Siswa. Implementasi ini cocok untuk sistem skala kecil hingga menengah seperti aplikasi desktop Manajemen Data Siswa lokal. Untuk itu, implementasi ini dapat memberikan kontribusi dalam pengembangan sistem informasi yang lebih efisien secara akademis, dan menjadi referensi praktis dalam mempelajari struktur data mendalam bidang ilmu yang kita geluti.

ABSTRACT

Data structure is one of the most important data structures for student data management. In this paper, a new data structure called Linked list is proposed for managing frequent data changed change such as student data. Linked lists are used to store, organize, and manage data efficiently so that can be used with the optimal way. For that, the author wants to evaluate use Linked List data structure in application simple Student data management (add, delete, search, display) dynamic with utilizing the Java programming language. The trial show how can linked list applied for managing frequent Data Changed Change such as Student Data. This implementation is suitable for small to medium scale systems such as local Student Data Management desktop applications. For That, this implementation can give contribution in development system information more academic efficient, and become reference practical in learning deep data structure field the science that we keep at it.

1. PENDAHULUAN

Dalam era serba digital saat ini, pengolaan data menjadi satu aspek penting dalam membangun sebuah sistem informasi khususnya dalam Lembaga pendidian tinggi. Setiap perguruan tinggi memiliki entitas mahasiswa yang memiliki berbagai atribut atau identitas yang harus dikelola secara efektif dan efisien. Oleh karenaitu dalam pengolaan datanya terlebih dalam antrian data, pemilihan metode struktur data yang tepat sangat penting dalam proses pengolahan data untuk efisiensi dan efektifitas dalm mengelola data mahasiswa yang memerlukan sistem yang mampu menangani perubahan data secra dinamis oleh karena itu pemilihan metode struktur data yang tepat

dapat menjadi kunci keberhasilan *output* sebuah permrosesan data.

Struktur data adalah cara untuk menyimpan, mengatur, danmengelola data secara efisien sehingga dapat digunakan dengan cara yang optimal.[1]

Salah satu struktur data yang relevan dan cocok dengan kebutuhan pengolaan data mahasiswa tersebut adalah *Linked list* memiliki alokasi dinamis. *Linked list* atau senarai berantai yang digunakan untuk menyimpan sejumlah objek data biasanya secara terurut sehingga memungkinkan penambahan, pengurangan, dan pencarian atas elemen data yang tersimpan dalam daftar dilakukan secara lebih efektif[2]. Singkatny metode *linked list* adalah metode yang fleksibel untuk pengolahan data dinamis.

Dalam penerapannya *Linked list* memiliki 3 jenis diantaranya ada *Single Linked list* yang terdiri dari elemen-elemen individu, dimana masing-masing dihubungkan dengan pointer tunggal[3]. Setiap *node* dalam *single linked list* memiliki dua bagian: data yang disimpan dan referensi ke *node* berikutnya dalam urutan. Jenis yan gkedua ada doubly linked list yang adalah suatu *linked list* yang mempunyai 2 penunjuk yaitu penunjuk ke simpul sebelumnya dan ke simpul berikutnya[4]. *Linked list double* merupakan perluasan dari *single linked list*. Sedangkan *circular linked list* merupakan model *linked list* yang terdiri dari *node* yang disusun sedemikian rupa sehingga saling berkaitan satu sama lain membentuk sebuah untaian data[5].

Berbeda dengan *Array* dimana daftar ini tidak dapat bertambah (atau) menyusut, hal ini menjadi kerugian besar karena pengguna perlu mengetahui ukuran *array* terlebih dahulu untuk mengimplementasikan dan mendapatkan nilai yang akan dialokasikan.[6] Karena sifat *array* yg statis atau tidak dapat diubah, tidaklah efektif untuk menggunakannya dalam pengolahan data mahasiswa. Namun meskipun *linked list* unggul dalam fleksibilitas, struktur ini memiliki kelemahan dalam hal akses langsung terhadap elemen tertentu, karena setiap pencarian harus dilakukan secara berurutan dari *node* awal.

Dalam penelitian tentang perbandingan kecepatan *Linked list* dan *array* yang dilakukan oleh [6], Linked list hanya membutuhkan O(1) untuk penyisipan sedangkan *array* list membutuhkan O(n). Akibatnya, lebih baik menggunakan linked list untuk mengalokasikan data secara lebih efisien dan menghapus data. Penelitian lain mengenai pemilihan pendekatan Linked list dan *array* yang dilakukan oleh [7] menyimpulkan bahwa pemilihan kedua metode ini bergantung pada spesifikasi dan kebutuhan aplikasi yang ingin dibuat, jika memerlukan ukuran tumbupan statis dan akses elemen cepat, *array* merupakan pilihan yang baik sebaliknya jika ukuran batch bersifat dinamis maka *linked list* akan lebih efisien.

Literatur lain tentang pengolahan data pribadi mahasiswa menggunakan Struktur data *Linked List* oleh [8] menyimpulkan implementasi *Linked List* dalam *Phyton* untuk pengolahan data mahasiswa berhasil dijalankan dengan baik, penambahan dan penghapusan data dapat dilakukan sehingga menghasilkan konfigurasi program yang berfungsi dengan baik dan efisien. Dalam jurnal referensi [9] yang membahas penggunaan *linked list* dalam aplikasi AR utnuk menyimpan urutan objek (marker) secara dinamis memperlihatkan pendekatan Linked list dalam menyimpan dan memanipulasi data secara real-time, yang mana konsep ini dapat diterapkan ke data mahasiswa misalnya saat pengurutan atau penyaringan.

Berdasarkan beberapa penelitian terdahulu diatas, penulis ingin mengevaluasi penggunaan struktur Data *Linked list* dalam pengelolaan data mahasiswa. Dengan demikian, hasil dari penelitian ini diharapkan dapat memberikan kontribusi dalam pengembangan sistem informasi akademik yang lebh efisien, serta menjadi referensi praktis dalam pembelajaran struktur data dalam bidang ilmu yang kita tekuni ini.

2. METODOLOGI

Adapun metode pengumpulan data dalam penelitian ini adalah degan melakukan studi literatur dari berbagai jurnal dan penelitian terdahulu yang membahsa topik yang relevan serta menggunakan ekperimen kode dengan membangun dan menguji pengimplementasian Linked list dalam aplikasi sederhana pengelolaan data mahasiswa (tambah, hapus, cari, tampil) dinamis dengan memanfaatkan Bahasa pemrograman java.

Tools yang digunakan dalam penelitian ini yakni Netbeans IDE 21 dengan sistem Operasi *Windows*.

Terdapat teknik pengumpulan data primer dan sekunder, dimana data primer diambil dari hasil eksekusi dan pengukuran waktu saat menjalankan program dan data sekunder diperoleh dari studi Pustaka dan literatur dari jurnal terdahulu untuk mendukung konsep teori dan perbandingan metode.

Data akan dianalisis berdasarkan Kecepatan eksekusi (dalam nanodetik),serta kemudahan implementasi dankompleksitas kode.

Brikut alur proses pengimplementasian Linked list dalam pengolaan data mahasiswa

3. IMPLEMENTASI DAN HASIL

3.1 Implementasi Kode

Implementasi kode dilakukan pada Java Netbeans IDE 21. Terdapat 3 buah class diantaranya Class Mahasiswa, *class Linked List* Mahasiswa dan class Main untuk pengolahan 1000 data mahasiswa untuk nantinya di uji kecepatan eksekusinya dan efisiensinya dalam mengelola data dinamis.

3.1.1 Struktur Data

Dalam Kelas Mahasiswa, dilakukan pendeklarasian atribut dengan menerapkan Struktur Data *Linked list*

```
public class Mahasiswa {
    String nim, nama, jurusan;
    double ipk;
    Mahasiswa next;

    public Mahasiswa(String nim, String nama, String jurusan, double ipk) {
        this.nim = nim;
        this.nama = nama;
        this.jurusan = jurusan;
        this.ipk = ipk;
        this.next = null;
    }
}
```

Gambar 1. Kode kelas Mahasiwa

3.1.2 Operasi-operasi Linked list

Menambahkan Mahasiswa

Kelas *LinkedList*Mahasiswa menyimpan kode untuk eksekusi data mahasiswa yaitu menmbah data:

```
public class LinkedListMahasiswa {
    Mahasiswa head;

    public void tambahMahasiswa(String nim, String nama, String jurusan, double ipk) {
        Mahasiswa baru = new Mahasiswa(nim, nama, jurusan, ipk);
        if (head == null) {
            head = baru;
        } else {
            Mahasiswa temp = head;
            while (temp.next != null) {
                temp = temp.next;
            }
            temp.next = baru;
        }
    }
```

Gambar 2. Kode menambah Data Mahasiswa

Mencari Data Mahasiswa

```
public Mahasiswa cariMahasiswa(String nim) {
    long start = System.nanoTime();

    Mahasiswa temp = head;
    while (temp != null) {
        if (temp.nim.equals(nim)) {
            long end = System.nanoTime();
            System.out.println("Durasi cari: " + (end - start) + " ns");
            return temp;
        }
        temp = temp.next;
    }

    long end = System.nanoTime();
    System.out.println("Durasi cari (tidak ditemukan): " + (end - start) + " ns");
    return null;
}
```

Gambar 3. Kode Mencari Data Mahasiswa

Menghapus Data Mahasiswa

```
public void hapusMahasiswa(String nim) {
    long start = System.nanoTime();

    if (head == null) {
        System.out.println("List kosong.");
        return;
    }

    if (head.nim.equals(nim)) {
        head = head.next;
    } else {
        Mahasiswa temp = head;
        while (temp.next != null && !temp.next.nim.equals(nim)) {
            temp = temp.next;
        }
        if (temp.next != null) {
            temp.next = temp.next.next;
        } else {
            System.out.println("Data dengan NIM " + nim + " tidak ditemukan.");
        }
    }

    long end = System.nanoTime();
    System.out.println("Durasi hapus: " + (end - start) + " ns");
}
```

]

Gambar 4. Kode Menghapus Data

Menampilkan Data

```
public void tampilkanData() {
    long start = System.nanoTime();

    Mahasiswa temp = head;
    if (temp == null) {
        System.out.println("List kosong.");
    }

    while (temp != null) {
        System.out.println("NIM: " + temp.nim + ", Nama: " + temp.nama + ", Jurusan: " + temp.jurusan + ", IPK: " + temp.ipk);
        temp = temp.next;
    }

    long end = System.nanoTime();
    System.out.println("Durasi tampil: " + (end - start) + " ns");
}
}
```

Gambar 5. Kode Tampil Data

Kelas Main

Pada kelas main, menyimpan *main method* atau titik masuk utama dari program yang bertugas membuat objek *linked list*, menguji 1000 data mahasiswa ke dalam *linked list,* mengukur rata-rata waktu *insert,* menguji pencarian, menghapus data serta menampilkan seluruh data mahasiswa.

```
public class Main {
    public static void main(String[] args) {
        LinkedListMahasiswa list = new LinkedListMahasiswa();

        // Insert 1000 data dan ukur waktu rata-rata
        int jumlahData = 1000;
        long totalWaktuInsert = 0;

        for (int i = 1; i <= jumlahData; i++) {
            String nim = "NIM" + i;
            String nama = "Mahasiswa " + i;
            String jurusan = "Informatika";
            double ipk = 2.5 + (i % 3);

            long start = System.nanoTime();
            list.tambahMahasiswa(nim, nama, jurusan, ipk);
            long end = System.nanoTime();

            totalWaktuInsert += (end - start);
        }

        double rataRata = (double) totalWaktuInsert / jumlahData;
        System.out.println("Rata-rata waktu insert 1000 data: " + rataRata + " ns");

        // Cari mahasiswa
        Mahasiswa hasilCari = list.cariMahasiswa("NIM500");
        if (hasilCari != null) {
            System.out.println("Ditemukan: " + hasilCari.nama);
        }

        // Hapus mahasiswa
        list.hapusMahasiswa("NIM10");

        // Tampilkan beberapa data
        list.tampilkanData();
    }
}
```

Gambar 6. Kelas Main

3.2 Hasil Uji Coba

Setelah kode program diatas di jalankan, menghasilkan output seperti yg diinginkan dengan hasil dari waktu eksekusi program yang beragam.

Pengujian dilakukan terhadap 4 operasi utama dengan data yang berjumlah 1000 elemen menggunakan system.nanoTime() untuk mengekur durasi waktu eksekusi dari masing-masing operasinya.

Table 1. Tabel waktu eksekusi

Operasi Olah Data	Data (1000 elemen)
Tambah(rata-rata)	4549.2 ns
Cari	27500 ns
Tampil	161372100 ns
Hapus	2300 ns

4. ANALISIS HASIL UJI COBA

Hasil dari uji coba menunjukkan bagaimana *Linked list* bisa diterapkan untuk mengelola data yang sering berubah ubah seperti data mahaiswa. Dengan *Linked list*, data mahasiswa dapat ditambahakan dan dihapus tanpa harus memindhkn elemen yg lain. Hal ini yang membuat *Linked list* unggul dalam mengelola data dinamis.

Dalam pengolahan data mahasiswa yg berjumlah 1000 elemen ini, Tambah data cenderung lebih lambat karena traversal ke akhir *list* ini menyebabkan waktu *insert* bertambah seiring jumlah data, sehingga rata-rata waktu akan naik jika jumlah data meningkat drastis karena kompleksitasnya (O(n)). Berdasarkan Tabel 1., untuk nilai rata-rata tambah data, artinya setiap mahasiswa dimasukkan ke *linked list* dalam waktu ± 4549,2 nanodetik atau sekitar 4,5492 milidetik. Performanya masih stabil untuk jumlah kecil hingga menengah (≤1000 elemen).

Tampil data secara penuh memakan waktu proporsional terhadap jumlah *node.*

Untuk pencarian data mahasiswa, pencarian tergantung posisi data kalau NIM di awal maka eksekusinya aka cepat sebaliknya jika NIM ada di akhir data eksekusinya akan lambat. Mirip dengan pencarian, penghapusan data tergantung posisi datanya.

5. KESIMPULAN

Dalam penerapan *Linked list* untuk mengelola data mahasiswa dengan operasi tambah, cari, tampil dan hapus, memiliki berbagai tingkat efiensi tergantung operasinya, untuk Tambah data masih perlu *traverse* ke akhir dan bisa dioptimasi dengan *tail pointer*, untuk operasi cari, *linked list* tidak efisien untuk pencarian harus *traverse* satu persatu, operasi hapus bergantung posisi data seperti pencarian, sedangkan operasi tampil bisa cepat atau lambat tergantung banyaknya data. Untuk rekomendasi pengmbangan lanjutan agar bisa menyempurnakan kekurangan yang ada yaitu dengan menambahkan *pointer* ke *tail* atau ujung data hal ini dilakukan untuk mengubah waktu *insert* menjadi O(1).

Untuk itu, Implementasi ini cocok digunakan untuk **sistem skala kecil hingga menengah** seperti aplikasi desktop manajemen data mahasiswa lokal. *Linked list* memang cocok untuk data yang berubah-ubah dan tidak memerlukan pencarian cepat.

Untuk 1000 data, performa masih baik dengan rata-rata sekitar 4549,2 ns per *insert.*

DAFTAR RUJUKAN

[1] *J. Kajian and I. Multidisipliner, "MENGANALISIS TINGKAT EFISIENSI STRUKTUR DATA ARRAY DAN LINKED LIST UNTUK MENGELOLA DATA MAHASISWA," vol. 8, no. 10, pp. 222–227, 2024.*

[2] *P. Setialana, "Analisis Kecepatan Struktur Data Linked List Dan Tree," no. February, 2016, doi: 10.13140/RG.2.1.1243.8801.*

[3] *K. N. WAHID, "Senarai Berantai (Linked List) Struktur Data," pp. 7–20, 2018, [Online]. Available: https://softsein.000webhostapp.com/2018/04/kuliah-senarai-berantai-linked-list-struktur-data*

[4] *D. L. List, S. Simpul, N. Endrecord, T. Point, D. Data, and P. Jadi, "Type nama_pointer = ↑Simpul".*

[5] *P. T. Congklak, LAPORAN PENELITIAN Judul: IMPLEMENTASI CIRCULAR LINKED LIST PADA PEMBUATAN GAME PERMAINAN TRADISIONAL CONGKLAK. 2014.*

[6] *K. R. Devi, "Analysis of Arraylist and Linked list," Int. J. Comput. Sci. Eng., 2019, doi: 10.26438/ijcse/v7i5.15661570.*

[7] *R. D. Setiyawan, D. Hermawan, A. F. Abdillah, A. Mujayanah, and R. Vindua, "PENGGUNAAN STRUKTUR DATA STACK DALAM PEMROGRAMAN C ++ DENGAN PENDEKATAN ARRAY DAN LINKED LIST," vol. 5, pp. 484–498, 2024.*

[8] *M. Rizki, A. Fitra, A. A. S. Effendi, and F. Ramadhani, "IMPLEMENTASI PYTHON DALAM PENGOLAHAN DATA PRIBADI MAHASISWA ILMU KOMPUTER ANGKATAN 23 PADA UNIVERSITAS NEGERI MEDAN MENGGUNAKAN STRUKTUR DATA LINKED LIST," vol. 9, no. 1, pp. 51–58, 2025.*

[9] *H. Wijaya, W. S. Wardhono, and I. Arwani, "Implementasi Linked List pada Interaksi Antar Marker Augmented Reality untuk Operand dan Operator Aritmetika," J. Pengemb. Teknol. Inf. dan Ilmu Komput. Univ. Brawijaya, vol. 2, no. 9, pp. 3328–3332, 2018.*