Jurnal Teknologi Dan Sistem Informasi Bisnis
Vol. 6 No. 2 April 2024 Hal. 286-296
http://jurnal.unidha.ac.id/index.php/jteksis

E-ISSN : **2655-8238**
P-ISSN : **2964-2132**

Implementasi *Stack* dan *Array* pada Pengurutan Lagu dengan Metode *Selection Sort*

Ghina Mawarni Putri[a], Kimberly Alfa Di Pradja[b], Muhammad Bilal Mardhiyyani Azizi[c], Prihatma Nurwahid[d], Abdi Surya Perdana[e], Munawir[f]

[a]Teknik Komputer, Kampus UPI di Cibiru, Universitas Pendidikan Indonesia, ghinamawarniputri@upi.edu
[b]Teknik Komputer, Kampus UPI di Cibiru, Universitas Pendidikan Indonesia, kimberlyalfa@upi.edu
[c]Teknik Komputer, Kampus UPI di Cibiru, Universitas Pendidikan Indonesia, muhammadbilalma@upi.edu
[d]Teknik Komputer, Kampus UPI di Cibiru, Universitas Pendidikan Indonesia, prihatma21@upi.edu
[e]Teknik Komputer, Kampus UPI di Cibiru, Universitas Pendidikan Indonesia, abdisury4@upi.edu
[f]Teknik Komputer, Kampus UPI di Cibiru, Universitas Pendidikan Indonesia, munawir@upi.edu

Submitted: 25-12-2023, Reviewed: 08-01-2024, Accepted 12-02-2024
https://doi.org/10.47233/jteksis.v6i2.1192

Abstract
A song is a work of art combined between sound art in which there is a melody and color of the singer's voice with language art. The number of songs nowadays requires a program that can sort songs to make it easier for users to find songs. This article aims to apply stack and array data structures to sort songs with the selection sort algorithm method in the hope that it can make it easier for users to find songs. In addition, this article aims to compare whether the selection sort algorithm is faster than insertion sort or vice versa. In making this journal, a literature study research method was carried out to deepen the material to be used in the project. The results of the program that has been made are that users can choose between adding songs, searching for songs, displaying songs that are already in the list, sorting songs, and exiting the program. With this, users can access and add their favorite songs anywhere and anytime. The conclusion of the results obtained is that selection sort is faster than insertion sort.
Keyword: *Stack, Array, Selection Sort, C++, Song*

Abstrak
Lagu merupakan karya seni gabungan antara seni suara yang di dalamnya terdapat melodi dan warna suara penyanyi dengan seni bahasa. Banyaknya lagu sekarang ini maka diperlukan sebuah program yang bisa mengurutkan lagu agar memudahkan pengguna dalam pencarian lagu. Artikel ini bertujuan untuk menerapkan struktur data *stack* dan *array* pada pengurutan lagu dengan metode algoritma *selection sort* dengan harapan dapat mempermudah pengguna dalam menemukan lagu. Selain itu, artikel ini bertujuan untuk membandingkan apakah algoritma *selection sort* lebih cepat daripada *insertion sort* ataukah sebaliknya. Dalam pembuatan jurnal ini dilakukan metode penelitian studi literatur untuk memperdalam materi yang akan digunakan pada proyek. Adapun hasil program yang telah dibuat yaitu pengguna dapat memilih antara menambahkan lagu, mencari lagu, menampilkan lagu yang sudah ada dalam daftar, mengurutkan lagu, dan keluar dari program. Dengan ini pengguna dapat mengakses dan menambahkan lagu favorit mereka dimanapun dan kapanpun. Simpulan dari hasil yang didapatkan yaitu bahwa *selection sort* lebih cepat daripada *insertion sort*.
Kata Kunci: *Stack, Array, Selection Sort, C++,* Lagu

PENDAHULUAN

Di era gen Z atau generasi yang dikenal dengan generasi internet ini merupakan generasi yang lahir dalam keadaan teknologi yang sudah maju. Generasi ini tumbuh dengan teknologi seperti *smartphone*, media sosial, internet, dan lain sebagainya yang mendampingi mereka dalam kehidupan sehari-hari. Era dengan kemajuan teknologi internet ini tentunya banyak industri yang berkembang pesat, salah satunya adalah lagu.

Lagu termasuk ke dalam karya sastra (puisi) [Moeliono(Peny.), 2003: 624]. Lagu merupakan ekspresi perasaan manusia, baik yang diucapkan maupun yang dirasakan [Koentjaraningrat, 1986]. Lagu merupakan suatu medium untuk menyampaikan pesan dan perasaan secara simbolis, sedangkan menurut KBBI lagu memiliki arti sebuah ragam suara yang berirama (dalam bercakap, bernyanyi, membaca, dan sebagainya). Lagu juga merupakan hasil karya seni gabungan antara seni suara dan seni bahasa yang juga pastinya melibatkan melodi dan warna suara penyanyi [18].

Lagu merupakan suatu rangkaian dari nada yang dipadukan dengan irama yang harmonis dan dilengkapi oleh syair yang membentuk sebuah harmonisasi indah. Lagu sering kali dijadikan salah satu media yang efektif untuk menyampaikan pesan kepada orang lain karena lagu bisa menangkap dan membangkitkan pola perasaan seperti pengharapan, keinginan, kegembiraan bahkan kegilaan [1]. Hal ini memberikan banyak manfaat dan kegunaan bagi pendengarnya. Lagu juga dapat diputar kapan saja dan dimana saja untuk menemani aktivitas

Jurnal Teknologi Dan Sistem Informasi Bisnis
Vol. 6 No. 2 April 2024 Hal. 286-296
http://jurnal.unidha.ac.id/index.php/jteksis

E-ISSN : **2655-8238**
P-ISSN : **2964-2132**

masyarakat, mulai dari belajar, bekerja, bersantai, dan masih banyak lagi.

Saat ini, banyak cara untuk mengakses lagu dengan mudah. Banyak aplikasi yang mendukung kumpulan lagu yang dapat didengar kapan saja. Aplikasi yang biasanya digunakan untuk mendengarkan lagu di antaranya *Spotify*, *Joox*, *Apple Music*, dan lainnya. Namun, banyaknya jumlah lagu yang ada terkadang membuat masyarakat kebingungan dan kesulitan untuk mencari lagu-lagu favoritnya. Oleh karena itu, diperlukan suatu sistem yang dapat membantu penggunanya untuk menyortir dan mengelola lagu-lagu favorit yang dipilih oleh pendengar. Pada artikel yang berjudul "Implementasi *Stack* dan *Array* pada Pengurutan Lagu Menggunakan Metode *Selection Sort*" ini, akan menerapkan hubungan antara struktur data dan algoritma dengan pengurutan lagu. Adapun struktur data dan algoritma yang dimaksud yaitu *stack* dan *array*.

Stack merupakan struktur data yang bertumpuk dengan konsep urutan yang disebut dengan LIFO (*Last In First Out*) [2][4][5]. Konsep LIFO tersebut memiliki beberapa proses. Pertama, proses *push* yang diibaratkan jika kita menyimpan buku di atas tumpukan buku. Kedua yaitu proses *pop* yang diibaratkan dengan pengambilan tumpukan buku pada tumpukan yang paling atas atau yang dapat disebut dengan *top of the stack*. Namun, pada percobaan ini hanya menggunakan konsep *push* saja yang artinya hanya ada proses penambahan data yang dalam hal ini yaitu penambahan lagu.

Array adalah struktur data variabel untuk menyimpan sekumpulan data dengan tipe data yang sama. Data akan menempati alamat yang berbeda-beda atau bisa disebut dengan elemen *array* yang dapat diakses melalui indeks yang berada di dalamnya. *Array* memiliki beberapa bentuk, seperti *array* 2 dimensi dan *array* 3 dimensi [3].

Stack dan *Array*, keduanya akan digunakan untuk menambah lagu, melihat lagu, dan mengurutkan lagu dengan metode *Selection Sort*. Jadi, metode *sorting* yang digunakan pada percobaan ini yaitu metode *sorting selection sort*. Alasan digunakannya *selection sort* pada percobaan ini yaitu karena konsepnya mudah untuk dipahami.

Selection sort sendiri merupakan sebuah algoritma yang digunakan untuk mengurutkan sebuah data dari yang terkecil ke yang terbesar (*ascending*) ataupun dari yang terbesar ke yang terkecil (*descending*). *Selection sort* bekerja dengan cara menyisipkan data pada posisi yang tepat yang dimulai jika menemukan data yang lebih kecil. Pencarian posisi yang tepat dilakukan dengan melakukan pencarian berurutan di dalam barisan elemen, selama pencarian posisi yang tepat dilakukan pergeseran elemen [3].

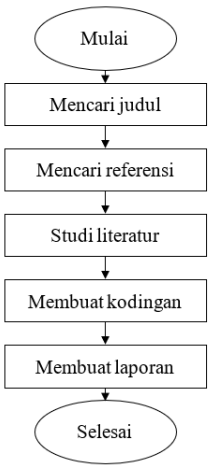
Adapun tujuan dari percobaan ini yaitu untuk mempermudah pengguna untuk mencari lagu yang mereka inginkan dengan hanya mengetik sebuah kata kunci yang berhubungan dengan lagu yang dicari. Selanjutnya, hasil pengurutan lagu dengan metode *selection sort* tersebut akan dibandingkan dengan metode *insertion sort* dengan tujuan untuk mengetahui apakah *selection sort* lebih cepat dibanding *insertion sort* ataukah sebaliknya. Selain itu, biasanya selection sort banyak digunakan pada data yang berupa angka atau nilai tertentu. Namun, pada percobaan ini, data yang diurutkan yaitu berupa kata yang selanjutnya akan diurutkan dari huruf A-Z.

METODE PENELITIAN

2.1. Metode Pengumpulan Data

A. Pendekatan Studi Literatur

Dalam pembuatan Jurnal dengan judul "Implementasi *Stack* dan *Array* pada Pengurutan Lagu Menggunakan Metode *Selection Sort*" ini tentunya memerlukan pencarian mendalam tentang materi yang bersangkutan. Hal ini dilakukan agar kita dapat memahami lebih dalam tentang materi proyek yang akan kita kerjakan. Pada pembuatan jurnal ini dilakukan penelitian dengan metode studi literatur, Metode studi literatur adalah serangkaian kegiatan yang berkenaan dengan metode pengumpulan data pustaka, membaca dan mencatat, serta mengelola bahan penelitian [Zed, 2008:3].



Gambar 2.1. Diagram alur pengerjaan proyek

Mencari Judul

Langkah awal dalam membuat jurnal ini yaitu mencari judul terlebih dahulu atau lebih tepatnya menentukan judul yang akan menjadi pokok utama dalam pembahasan jurnal ini. Penentuan judul dilakukan kurang lebih dalam jangka waktu 2 minggu dengan melihat permasalahan yang ada di lingkungan sekitar yang kemudian ditentukanlah judul yaitu "Implementasi *Stack* dan *Array* pada

Jurnal Teknologi Dan Sistem Informasi Bisnis
Vol. 6 No. 2 April 2024 Hal. 286-296
http://jurnal.unidha.ac.id/index.php/jteksis

E-ISSN : **2655-8238**
P-ISSN : **2964-2132**

Pengurutan Lagu Menggunakan Metode *Selection Sort*".

Mencari Referensi
Setelah judul ditentukan, langkah selanjutnya yaitu mencari referensi-referensi yang berkaitan dengan judul tersebut, entah itu dari segi *stack*-nya, *array*-nya, *selection sort*-nya, maupun dari segi sistematika jurnalnya yang sesuai dengan kaidah jurnal ilmiah. Pada tahap ini, dicari juga referensi kodingan sebagai kodingan dasar untuk penyelesaian program yang sesuai dengan judul.

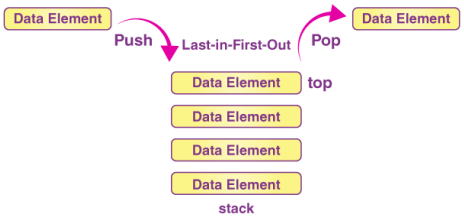
Studi Literatur
Tahap studi literatur ini dilakukan untuk mengkaji lebih dalam berkenaan dengan judul yang telah ditentukan. Dari referensi-referensi yang telah dicari tersebut, dikaji lebih dalam untuk melakukan pengembangan dari kodingan atau program yang telah ada tersebut. Selain itu, mencari juga referensi lain yang selanjutnya dikaji lebih dalam lagi seperti sebelumnya.

Membuat Kodingan
Setelah mencari referensi dan juga melakukan studi literatur yang mendalam, selanjutnya yaitu masuk ke dalam tahap membuat kodingan. Pada tahap ini, sedikit demi sedikit kodingan program dirancang sedemikian rupa sehingga didapatkan hasil/program yang diinginkan berdasarkan judul jurnal ini yaitu "Implementasi *Stack* dan *Array* pada Pengurutan Lagu Menggunakan Metode *Selection Sort*".

Membuat Laporan
Setelah program selesai dibuat, selanjutnya masuk ke dalam tahap membuat laporan. Laporan ini disusun dan dimuat dalam bentuk artikel ilmiah atau jurnal seperti pada jurnal yang telah dibuat dengan panduan *template* yang sesuai.

Penerapan *stack* dan *array* pada pengurutan *selection sort* yaitu bahwa *array* untuk menyimpan sekumpulan data lagu untuk program, sedangkan fungsi *stack* dapat digunakan untuk menyimpan indeks yang telah ditemukan pada setiap iterasi (salah satu sifat yang ada pada algoritma untuk melakukan perulangan pada suatu urutan). Dengan hal ini, kita dapat memahami lebih dalam tentang konsep *selection sort* mulai dari dasarnya.

1. *Stack*
Seperti yang sudah dijelaskan sebelumnya bahwa *stack* merupakan sebuah tumpukan dengan metode LIFO (*Last In First Out*), yaitu data terakhir yang masuk ke dalam tumpukan merupakan data pertama yang akan dikeluarkan [4][5]. Namun, pada percobaan ini, metode yang digunakan hanyalah metode *push* saja. Artinya, hanya melakukan proses pemasukkan data saja tanpa menghapus data.



Gambar 2.2. *Stack* (*Sumber:* https://byjus.com/gate/stack-and-its-applications/)

Gambar 2.2. menggambarkan bagaimana *stack* bekerja. Istilah *push* data merupakan penambahan data pada tumpukan, sedangkan istilah *pop* data merupakan menghapus data pada tumpukan data teratas.

2. *Array*
Array atau larik merupakan jenis struktur data yang terdiri dari sejumlah komponen yang memiliki jenis atau tipe data yang sama. Dalam definisinya, *array* merupakan suatu variabel yang mampu menyimpan lebih dari satu nilai sejenis dengan tipe data yang serupa. Sebuah *array* memiliki jumlah komponen yang tetap dan jumlah ini ditunjukkan oleh indeks yang disebut sebagai tipe indeks. Proses pengolahan data *array* dilakukan berdasarkan indeksnya. *Array* terbagi menjadi *array* satu dimensi (*array* linier), dua dimensi (matriks), dan multidimensi. *Array* dapat memiliki tipe data sederhana (seperti *byte*, *word*, *integer*, *real*, *boolean*, *char*, ataupun *string*) dan tipe data skalar, yang berarti komponen-komponen atau elemen-elemen *array* memiliki nilai dengan tipe data yang sama [7].

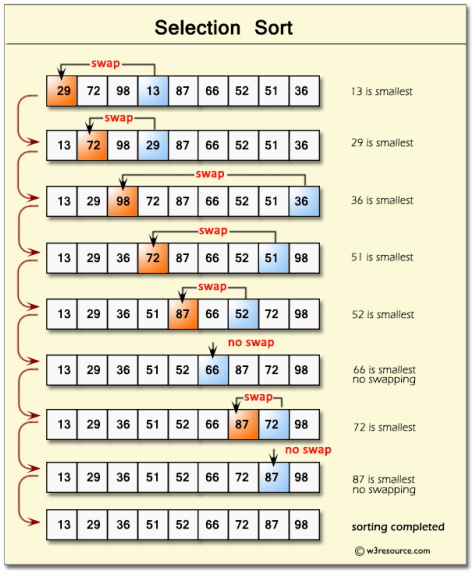
Gambar 2.3. *Array* (*Sumber:* https://blog.algorit.ma/struktur-data-array/)

3. **Algoritma *Selection Sort***
Algoritma *selection sort* ini merupakan salah satu algoritma untuk mengurutkan data, baik secara *ascending* (pengurutan data dari yang terkecil ke yang terbesar) maupun *descending* (pengurutan data dari yang terbesar ke yang terkecil). Algoritma *selection sort* sering disebut sebagai metode maksimum atau minimum. Istilah "metode maksimum" merujuk pada pendekatan algoritma

Jurnal Teknologi Dan Sistem Informasi Bisnis
Vol. 6 No. 2 April 2024 Hal. 286-296
http://jurnal.unidha.ac.id/index.php/jteksis

E-ISSN : **2655-8238**
P-ISSN : **2964-2132**

ini yang berfokus pada pemilihan data atau elemen maksimum sebagai dasar pengurutan. Konsepnya melibatkan pemilihan elemen maksimum yang kemudian ditukar dengan elemen terakhir untuk pengurutan secara menaik, dan dengan elemen pertama untuk pengurutan secara menurun. Sebagai metode minimum, algoritma *selection sort* juga berlandaskan pada pemilihan elemen minimum sebagai dasar pengurutan. Ide dasarnya yaitu memilih elemen minimum dan menukarnya dengan elemen terakhir untuk pengurutan menaik, serta dengan elemen pertama untuk pengurutan menurun [3] [19].



Gambar 2.4. *Selection sort* (*Sumber: https://images.app.goo.gl/WdHhmaSbU8h5cK3MA*)

Dapat dilihat dari gambar diatas cara kerja dari algoritma *selection sort* yaitu apabila ditemukan data yang lebih kecil pada *array* maka akan ditukar dengan data yang lebih besar di depannya, dan akan terus berulang sampai data terurut dengan benar [3].

B. Studi Pustaka

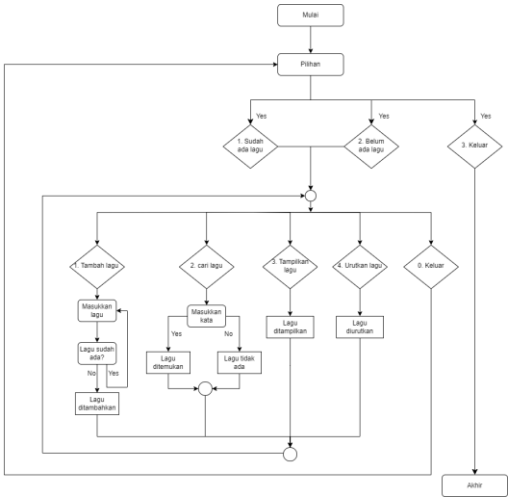
Metode studi pustaka melibatkan pengumpulan data dengan menggali informasi dari berbagai sumber seperti buku, literatur, catatan, dan laporan yang terkait dengan permasalahan yang hendak diatasi atau berupa penelitian yang pernah dilakukan sebelumnya [6].

1. Penelitian yang dilakukan oleh Hindriyani, Narwen, dan Yozza yaitu mencoba untuk mengimplementasikan antrian (*queue*) pada *array*. *Queue* ini memiliki konsep FIFO (*First In First Out*). Konsep tersebut memiliki arti bahwa data yang pertama kali masuk merupakan data yang akan keluar atau selesai pertama kali [7].
2. Penelitian yang dilakukan oleh Benardo, Mesterjon, dan Zulita yaitu mengimplementasikan metode *selection sort* untuk menentukan nilai prestasi siswa kelas 3 SD dan kelas 4 SD di SD Negeri 107 Seluma. Pada metode *selection sort* yang diterapkan, pendekatan yang digunakan yaitu metode *descending*. Hal tersebut dilakukan dengan mengurutkan nilai-nilai hasil penilaian siswa dari nilai terbesar hingga nilai terkecil. Hasil yang didapatkan dari penelitian tersebut disusun dalam suatu laporan penilaian yang telah diurutkan nilainya berdasarkan nilai tertinggi hingga nilai terendah [8].
3. Penelitian yang dilakukan oleh Putra, J. D. et al. yaitu menggunakan algoritma pengurutan *selection sort* pada sistem informasi lokasi kontrakan di Kota Medan. Pada penelitian tersebut, data yang diurutkan yaitu harga sewa kontrakan dari harga terendah ke harga tertinggi atau sebaliknya. Adapun hasil yang didapatkan yaitu sistem informasi lokasi kontrakan tersebut tidak hanya menampilkan harganya saja namun juga dapat menampilkan pemilik kontrakan, titik koordinat kontrakan, serta fasilitas yang terdapat pada kontrakan tersebut [9].
4. Penelitian yang dilakukan oleh Hakim ini yaitu implementasi *selection sort* untuk menentukan barang yang harus distok ulang dalam sebuah penjualan. Pada sistem tersebut, barang perlu diurutkan jumlah ketersediaannya dan juga menentukan barang mana yang sekiranya harus distok ulang. Adapun output yang dihasilkan dari sistem informasi penjualan tersebut berupa laporan data barang yang telah diurutkan dari stok paling sedikit hingga paling banyak [10].
5. Penelitian yang dilakukan oleh Harahap dan Effendy yaitu mengimplementasikan algoritma *selection sort* dalam membangun aplikasi pemesanan jasa *make-up*. Pada sistem tersebut akan mengurutkan harga jasa pemesanan *make-up* sebagai kunci dalam penerapan algoritma *selection sort*. Adapun hasil yang didapatkan yaitu berupa urutan harga sedemikian rupa sehingga memudahkan pelanggan dalam memilih paket berdasarkan biaya yang dimilikinya [11].
6. Penelitian yang dilakukan oleh Farah dan Kurniawati yaitu menerapkan *selection sort* sebagai algoritma untuk menentukan nilai tertinggi siswa. Adapun yang akan diurutkan yaitu nilai masing-masing siswa yang kemudian diurutkan dari yang tertinggi hingga nilai terendah sehingga akan tampil nilai tertinggi siswa [12].

2.2. Metode Perancangan Program

Jurnal Teknologi Dan Sistem Informasi Bisnis
Vol. 6 No. 2 April 2024 Hal. 286-296
http://jurnal.unidha.ac.id/index.php/jteksis

E-ISSN : **2655-8238**
P-ISSN : **2964-2132**



Gambar 2.5. *Flowchart* program pengurutan lagu

Gambar 2.5 merupakan gambaran alur program yang akan dibuat. Jadi, pengguna akan diminta untuk memilih apakah ingin mengolah data yang sudah tersedia daftar lagunya ataukah ingin mengolah data yang daftar lagunya belum tersedia sama sekali sehingga perlu menambah data sendiri. Adapun alur program tersebut akan diterapkan pada bahasa C++.

HASIL DAN PEMBAHASAN

3.1. Fungsi-fungsi yang Digunakan dalam Program

Fungsi Pembuatan *File* Eksternal

```
void baca_FileEksternal_AdaLagu(vector<string>& data) {
   ifstream fileInput("sudah_ada_lagu.txt");
   if (fileInput.is_open()) {
      string namaLagu;
      while (getline(fileInput, namaLagu)) {
         data.push_back(namaLagu);
      }
      fileInput.close();
   } else {
      cout << "Gagal membuka file." << endl;
   }
}
```

File eksternal digunakan sebagai tempat penyimpanan lagu dengan tujuan ketika lagu ditambahkan akan tersimpan ke dalam sebuah *file* dengan ekstensi ".txt" sehingga ketika program dihentikan maka lagu yang telah ditambahkan tersebut tidak akan menghilang. Jadi, sebelum program utama dijalankan, fungsi ini akan terlebih dulu dieksekusi agar *file* atau bisa kita sebut wadah telah ada (telah dibuat).

Fungsi Penambahan Lagu

```
void tulis_FileEksternal_AdaLagu(const vector<string>& data) {
   ofstream fileOutput("sudah_ada_lagu.txt");
   if (fileOutput.is_open()) {
      for (const string& namaLagu : data) {
         fileOutput << namaLagu << endl;
      }
      fileOutput.close();
   } else {
      cout << "Gagal membuka file." << endl;
   }
   cout << endl;
}
```

Fungsi penambahan lagu ini berfungsi untuk memasukkan lagu ke dalam sebuah vektor dengan tipe data *string* dan juga memasukkan lagu ke dalam *file* eksternal yang telah dibuat sebelumnya.

Fungsi Pencarian Lagu

```
void cariLagu_denganKata_Ada(const vector<string>& data, const string& kata) {
   vector<string> hasil_pencarian;
   for (const string& namaLagu : data) {
      if (namaLagu.find(kata) != string::npos) {
         hasil_pencarian.push_back(namaLagu);
      }
   }

   // Menampilkan hasil pencarian
   if (!hasil_pencarian.empty()) {
      cout << "\nHasil Pencarian untuk \"" << kata << "\"." << endl;
      int z=1;
      for (const string& lagu : hasil_pencarian) {
         cout << z << ". " << lagu << endl;
         z++;
      }
      cout << "\n(Total Lagu Ditemukan: " << z-1 << " Lagu)\n\n";
   } else {
      cout << "\nTidak ada lagu yang cocok dengan kata \"" << kata << "\"." << endl;
   }
   getch();
}
```

Fungsi di atas merupakan fungsi yang digunakan untuk mencari lagu dengan kata kunci yang dimasukkan dengan memanfaatkan *file* eksternal sebagai tempat untuk mencari lagu baik yang sudah tersedia maupun yang belum tersedia.

Fungsi Menampilkan Seluruh Lagu

```
void TampilkanLagu_Ada(const vector<string>& data) {
   cout << "Seluruh Data Lagu:" << endl;
   int z=1;
```

Jurnal Teknologi Dan Sistem Informasi Bisnis
Vol. 6 No. 2 April 2024 Hal. 286-296
http://jurnal.unidha.ac.id/index.php/jteksis

E-ISSN : **2655-8238**
P-ISSN : **2964-2132**

```
    for (const string& namaLagu : data) {
        cout << z << ". "<< namaLagu << endl;
        z++;
    }
    cout << endl;
}
```

Fungsi tersebut yaitu fungsi untuk menampilkan seluruh lagu yang ada dalam vektor *string*. Vektor memiliki cara kerja yang mirip dengan *array*, yaitu sama-sama menyimpan data dengan urutan indeks. Namun, vektor lebih efisien dibandingkan dengan *array* karena ukurannya yang tidak harus dideklarasikan (bersifat dinamis) [13]. Adapun sistem penambahan data dalam vektor mirip dengan *stack* yaitu setiap penambahan lagu akan terus bertumpuk seperti metode *push*.

Fungsi Pengurutan Lagu (Selection Sort)

```
void selectionSort_Ada(vector<string>& data) {
    size_t n = data.size();

    for (size_t i = 0; i < n - 1; i++) {
        // Temukan indeks minimum di antara
data[i+1] sampai data[n-1]
        size_t minIndex = i;
        for (size_t j = i + 1; j < n; j++) {
            if (data[j] < data[minIndex]) {
                minIndex = j;
            }
        }

        // Tukar elemen minimum dengan elemen
pertama yang belum diurutkan
        if (minIndex != i) {
            swap(data[i], data[minIndex]);
        }
    }

    cout << "\n(Data telah diurutkan!)\n";
    getch();
}
```

Fungsi tersebut di atas merupakan algoritma *selection sort* untuk mengurutkan kata dengan metode *ascending* (menaik dari A ke Z). Jika biasanya yang dibandingkan untuk pengurutan data yaitu berupa angka atau suatu nilai maka pada fungsi ini yang dibandingkan yaitu berupa kata.

3.2. Implementasi *Source Code* pada Program dan Hasilnya

Berikut merupakan hasil implementasi *source code* dan hasil output program yang telah dibuat.

```
---------- PROGRAM PENGURUTAN DAFTAR LAGU ----------

      1. Jalankan Program (Daftar lagu sudah ada)
      2. Jalankan Program (Daftar lagu belum ada)
      0. Hentikan Program

      Masukkan pilihan (1/2/0): _
```

Gambar 3.1. Tampilan awal program

Ketika program dijalankan maka akan muncul tampilan seperti pada gambar 3.1. Pada tampilan tersebut, terdapat beberapa pilihan yaitu menjalankan program dengan daftar lagu yang telah tersedia atau yang belum tersedia sama sekali.

Jalankan Program (Daftar lagu sudah ada)

Pada program ini, daftar lagu sudah disediakan sehingga pengguna tinggal mengolah data saja namun tetap bisa menambah lagu.

```
Menu (Daftar lagu sudah ada):
1. Tambah Lagu
2. Cari Lagu
3. Tampilkan Seluruh Lagu
4. Urutkan Lagu (Selection Sort)
0. Keluar

Pilih menu (1/2/3/4/0):
```

Gambar 3.2. Menu pilihan (Lagu sudah ada)

Terdapat beberapa pilihan program pada gambar 3.2. Untuk membuktikan bahwa daftar lagu sudah tersedia, kita dapat memilih pilihan nomor 3 yaitu menampilkan seluruh lagu.

```
336. Why Don't We - What Am I
337. Boyz II Men - On Bended Knee
338. Etham - 12:45
339. Brian McKnight - Back At One
340. Elijah Woods - 24/7,365
341. Radiohead - Creep
342. Why Don't We - Trust Fun Baby
343. Juicy Lucy - Sialan
344. Lyodra - Tak Dianggap
345. Juicy Lucy - Asing
346. Juicy Lucy - Tanpa Tergesa
347. Yahya - Keepyousafe
348. Dewa 19 - Kangen
349. Girl In Red - We Fell In Love In October
350. D4vd - Here With Me


Menu (Daftar lagu sudah ada):
1. Tambah Lagu
2. Cari Lagu
3. Tampilkan Seluruh Lagu
4. Urutkan Lagu (Selection Sort)
0. Keluar

Pilih menu (1/2/3/4/0): _
```

Gambar 3.3. Jumlah lagu yang tersedia

Terlihat pada gambar 3.3, jumlah lagu yang tersedia yaitu sebanyak 350 lagu. Terlihat juga bahwa lagu-lagu tersebut masih belum terurut sesuai dengan urutan hurufnya. Kemudian program kembali lagi ke tampilan menu awal.

Jurnal Teknologi Dan Sistem Informasi Bisnis
Vol. 6 No. 2 April 2024 Hal. 286-296
http://jurnal.unidha.ac.id/index.php/jteksis

E-ISSN : **2655-8238**
P-ISSN : **2964-2132**

```
(Data telah diurutkan!)

Menu (Daftar lagu sudah ada):
1. Tambah Lagu
2. Cari Lagu
3. Tampilkan Seluruh Lagu
4. Urutkan Lagu (Selection Sort)
0. Keluar

Pilih menu (1/2/3/4/0):
```

Gambar 3.4. Urutkan lagu

Ketika memilih program "Urutkan Lagu (*Selection Sort*)" maka akan muncul tampilan seperti pada gambar 3.4 yang artinya data sudah diurutkan. Untuk membuktikannya kita dapat memilih tampilkan seluruh lagu untuk melihat apakah data telah terurut ataukah belum.

```
334. Voodoo - Katakan
335. Weezer - Back to the Shack
336. Weezer - Buddy Holly
337. Whizzkid - Lepaskanlah
338. Why Don't We - 8 Letters
339. Why Don't We - How Do You Love Sombody
340. Why Don't We - Let Me Down Easy (Lie)
341. Why Don't We - Love Back
342. Why Don't We - Trust Fun Baby
343. Why Don't We - What Am I
344. Yahya - Keepyousafe
345. Yeah Yeah Yeahs - Maps
346. Yovie & Nuno - Janci Suci
347. Yura Yunita - Buka Hati
348. Yura Yunita - Intuisi
349. Yura Yunita - Tenang
350. Yura Yunita - Tutur Batin


Menu (Daftar lagu sudah ada):
1. Tambah Lagu
2. Cari Lagu
3. Tampilkan Seluruh Lagu
4. Urutkan Lagu (Selection Sort)
0. Keluar

Pilih menu (1/2/3/4/0):
```

Gambar 3.5. Hasil pengurutan lagu

Gambar 3.5 merupakan tampilan seluruh lagu ketika telah diurutkan. Pengurutan lagu menggunakan algoritma selection sort dengan metode *ascending* tersebut bisa dilakukan meski data yang dibandingkan yaitu bukan nilai atau angka melainkan kata. Kita juga dapat menambah lagu dengan memilih program "Tambah Lagu".

```
Format Penambahan Lagu: Artis/Band - Judul Lagu, CONTOH: Linkin Park - In The End
Masukkan lagu: Dewa 19 - Kangen

Lagu sudah ada dalam daftar. Tidak dapat menambahkan data yang sama dua kali.

Menu (Daftar lagu sudah ada):
1. Tambah Lagu
2. Cari Lagu
3. Tampilkan Seluruh Lagu
4. Urutkan Lagu (Selection Sort)
0. Keluar

Pilih menu (1/2/3/4/0):
```

Gambar 3.6. Validasi penambahan lagu

Ketika lagu yang ditambahkan sudah ada dalam daftar maka akan muncul tampilan seperti pada gambar 3.6. Adapun tampilan ketika lagu yang berbeda ditambahkan yaitu sebagai berikut.

```
Format Penambahan Lagu: Artis/Band - Judul Lagu, CONTOH: Linkin Park - In The End
Masukkan lagu: Alan Walker - Lily

(Data berhasil ditambahkan!)

Menu (Daftar lagu sudah ada):
1. Tambah Lagu
2. Cari Lagu
3. Tampilkan Seluruh Lagu
4. Urutkan Lagu (Selection Sort)
0. Keluar

Pilih menu (1/2/3/4/0):
```

Gambar 3.7. Penambahan lagu

Kemudian untuk membuktikan bahwa lagu tersebut sudah bertambah atau tidak bisa kita gunakan program tampilkan lagu.

```
350. D4vd - Here With Me
351. Alan Walker - Lily


Menu (Daftar lagu sudah ada):
1. Tambah Lagu
2. Cari Lagu
3. Tampilkan Seluruh Lagu
4. Urutkan Lagu (Selection Sort)
0. Keluar

Pilih menu (1/2/3/4/0):
```

Gambar 3.8. Tampilan hasil penambahan lagu

Pada program ini, setiap penambahan lagu baru, lagu tersebut akan berada di urutan paling bawah (belum terurut). Kemudian fungsi terakhir yaitu pencarian lagu.

```
Masukkan lagu yang ingin dicari: Yura
(Lagu ditemukan!)

Hasil Pencarian untuk "Yura".
1. Yura Yunita - Buka Hati
2. Yura Yunita - Intuisi
3. Yura Yunita - Tenang
4. Yura Yunita - Tutur Batin

(Total Lagu Ditemukan: 4 Lagu)


Menu (Daftar lagu sudah ada):
1. Tambah Lagu
2. Cari Lagu
3. Tampilkan Seluruh Lagu
4. Urutkan Lagu (Selection Sort)
0. Keluar

Pilih menu (1/2/3/4/0):
```

Gambar 3.9. Hasil pencarian lagu

Untuk mencari lagu seperti pada gambar 3.9, diperlukan sebuah kata kunci baik itu dari nama artis/band-nya maupun judul lagunya. Program tersebut akan mencari nama artis atau judul lagu yang memiliki kata kunci yang diinginkan oleh pengguna. Dengan memanfaatkan *file* eksternal, dapat dengan mudah untuk mencari kata kunci yang diinginkan oleh pengguna. Selain itu, akan muncul juga total lagu yang ditemukan seperti pada gambar 3.9.

Jalankan Program (Daftar lagu belum ada)

Sama dengan program yang daftar lagunya telah tersedia, namun bedanya pada program ini daftar lagu belum tersedia sehingga meminta pengguna untuk memasukkan lagu yang diinginkan. Ini cocok ketika pengguna ingin menambahkan lagu favoritnya saja.

```
Menu (Daftar lagu belum ada):
1. Tambah Lagu
2. Cari Lagu
3. Tampilkan Seluruh Lagu
4. Urutkan Lagu (Selection Sort)
0. Keluar

Pilih menu (1/2/3/4/0):
```

Gambar 3.10. Menu program (daftar lagu belum ada)

Tampilan menu pada program ini sama dengan program sebelumnya. Hanya saja yang membedakannya yaitu belum ada daftar lagunya.

Jurnal Teknologi Dan Sistem Informasi Bisnis
Vol. 6 No. 2 April 2024 Hal. 286-296
http://jurnal.unidha.ac.id/index.php/jteksis

E-ISSN : **2655-8238**
P-ISSN : **2964-2132**

```
Seluruh Data Lagu:


Menu (Daftar lagu belum ada):
1. Tambah Lagu
2. Cari Lagu
3. Tampilkan Seluruh Lagu
4. Urutkan Lagu (Selection Sort)
0. Keluar

Pilih menu (1/2/3/4/0): _
```

Gambar 3.11. Tampilan seluruh lagu (lagu belum ada)

Terlihat bahwa data lagu masih belum tersedia sehingga perlu ditambahkan lagu dengan menggunakan fungsi tambah lagu. Adapun cara kerja dari seluruh program pada program lagu yang belum tersedia daftar lagunya sama seperti program yang daftar lagunya telah tersedia.

3.3. Analisis Perbandingan antara *Selection Sort* dan *Insertion Sort*

Penelitian yang dilakukan oleh Maulana yaitu membandingkan kompleksitas antara *selection sort* dan *insertion sort*. Hasil penelitian yang dilakukan yaitu bahwa algoritma *insertion sort* lebih cepat daripada algoritma *selection sort* [14].

Hasil penelitian yang dilakukan oleh Retnoningsih yaitu bahwa *insertion sort* lebih unggul (lebih cepat) ketika data yang diuji berjumlah lebih sedikit. Namun, ketika data yang diuji berjumlah lebih banyak maka algoritma *selection sort* lebih unggul (lebih cepat) [3].

Penelitian yang dilakukan oleh Sunandar juga yaitu membandingkan algoritma *selection sort* dan *insertion sort*. Sunandar menyimpulkan bahwa tidak ada perbedaan yang signifikan antara keduanya. Namun, Sunandar menyebutkan bahwa algoritma *selection sort* terlihat lebih sederhana karena hanya membandingkan data yang satu dengan data lainnya yang ketika nilai terkecil ditemukan posisinya akan saling bertukar. Sedangkan, algoritma *insertion sort* sedikit kompleks karena memerlukan sebuah variabel penampung data sementara yang kemudian melakukan proses geser (bukan pindah atau bertukar) sehingga waktu proses relatif lebih lama daripada *selection sort* [15].

Hasil penelitian yang dilakukan oleh Kusumaningrum juga menyebutkan bahwa algoritma *insertion sort* lebih cepat ketimbang *bubble sort* dan *selection sort* [16].

```
Jumlah Data: 500 Nama

(Klik enter untuk lanjut)
```

Gambar 3.12. Jumlah data yang akan diurutkan

Gambar 3.13. Data yang belum diurutkan (*selection sort*)

Gambar 3.14. Data yang sudah diurutkan (*selection sort*)

Gambar 3.15. Data yang belum diurutkan (*insertion sort*)

Gambar 3.16. Data yang sudah diurutkan (*insertion sort*)

```
---------------------------------------------
Selection = 0.375 seconds
Insertion = 0.375 seconds
---------------------------------------------
```

Gambar 3.17. Pengujian 1

```
---------------------------------------------
Selection = 0.39 seconds
Insertion = 0.379 seconds
---------------------------------------------
```

Gambar 3.18. Pengujian 2

```
---------------------------------------------
Selection = 0.36 seconds
Insertion = 0.375 seconds
---------------------------------------------
```

Gambar 3.19. Pengujian 3

```
---------------------------------------------
Selection = 0.328 seconds
Insertion = 0.359 seconds
---------------------------------------------
```

Gambar 3.20. Pengujian 4

```
---------------------------------------------
Selection = 0.391 seconds
Insertion = 0.391 seconds
---------------------------------------------
```

Jurnal Teknologi Dan Sistem Informasi Bisnis
Vol. 6 No. 2 April 2024 Hal. 286-296
http://jurnal.unidha.ac.id/index.php/jteksis

E-ISSN : **2655-8238**
P-ISSN : **2964-2132**

Gambar 3.21. Pengujian 5

Pada percobaan perbandingan algoritma *selection sort* dan *insertion sort* ini, disediakan sebanyak 500 data dalam bentuk string (kata) seperti pada gambar 3.12. Kemudian 500 data tersebut akan di-*sorting* dengan menggunakan *selection sort* dan *insertion sort*. Pada percobaan ini, dilakukan sebanyak 5 kali yang kemudian akan dirata-ratakan sehingga didapatkan algoritma yang paling cepat dalam hal pengurutan data. Hasil pengujian akan dibuat dalam bentuk tabel sebagai berikut.

Tabel 3.1. Hasil Perbandingan

Pengujian ke-	*Selection Sort (seconds)*	*Insertion Sort (seconds)*
1	0,375	0,375
2	0,39	0,379
3	0,36	0,375
4	0,328	0,359
5	0,391	0,391
Rata-rata	0,3688	0,3758

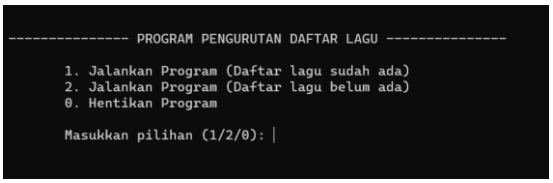
Dari tabel 3.1 didapatkan hasil bahwa algoritma *selection sort* sedikit lebih cepat dibandingkan algoritma *insertion sort*. Ini berbanding terbalik dengan apa yang disebutkan oleh Maulana, Retnoningsih, Sunandar, dan Kusumaningrum yaitu bahwa algoritma *insertion sort* lebih cepat dibandingkan algoritma *selection sort*. Perbedaan hasil tersebut mungkin dipengaruhi oleh beberapa faktor seperti yang disebutkan oleh Sunandar yaitu salah satunya bahwa algoritma *insertion sort* lebih kompleks sehingga berpengaruh terhadap kecepatan waktu eksekusi [15].

3.4. Pembahasan Hasil Program

Hasil pengujian menunjukkan bahwa penggunaan *stack* dan *array* memungkinkan pengendalian yang lebih efisien terhadap daftar putar lagu yang terus berkembang, sementara *array* memungkinkan akses langsung ke elemen-elemen lagu tertentu yang mengurangi waktu pencarian lagu spesifik. Namun, karena kompleksitas algoritma yang meningkat, penggunaan *stack* dan *array* dalam implementasi x*selection sort* mungkin memiliki batasan. Berikut merupakan cuplikan kode yang mengilustrasikan bagaimana *selection sort* diimplementasikan dengan menggunakan *stack* dan *array* pada daftar lagu.

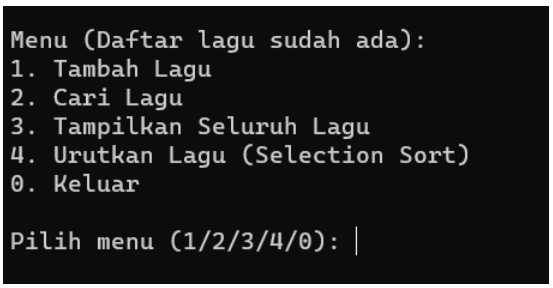
```
void selectionSort_Ada(vector<string>& data) {
   size_t n = data.size();

   for (size_t i = 0; i < n - 1; i++) {
      // Temukan indeks minimum di antara
data[i+1] sampai data[n-1]
      size_t minIndex = i;
      for (size_t j = i + 1; j < n; j++) {
         if (data[j] < data[minIndex]) {
            minIndex = j;
         }
      }

      // Tukar elemen minimum dengan elemen
pertama yang belum diurutkan
      if (minIndex != i) {
         swap(data[i], data[minIndex]);
      }
   }

   cout << "\n(Data telah diurutkan!)\n";
   getch();
}
```

Setelah melakukan implementasi *selection sort* menggunakan *stack* dan *array* pada daftar lagu, contoh output yang dihasilkan adalah sebagai berikut.



Gambar 3.22. Halaman awal

Pada halaman awal akan muncul tampilan seperti pada gambar 3.22. Pilihan untuk menjalankan program dengan lagu yang sudah ada atau dengan lagu yang belum ada atau menghentikan program yang sedang dijalankan. Setelah memilih, kita akan dilanjutkan ke halaman berikutnya dengan 5 pilihan, yaitu menambahkan lagu, mencari lagu, menampilkan seluruh lagu, mengurutkan lagu, dan keluar dari program seperti pada gambar 3.23.



Gambar 3.23. Pilihan menu

Untuk pengurutan lagu sendiri, dikarenakan kita fokus pada *selection sort* maka kodingan secara otomatis akan mengurutkan dengan metode tersebut. Setelah melakukan pengujian, dapat diketahui bahwa penggunaan *selection sort* dengan *stack* yaitu untuk manajemen sementara dan *array* untuk akses cepat sehingga dapat melakukan pengurutan daftar lagu dengan baik. Program ini

Jurnal Teknologi Dan Sistem Informasi Bisnis
Vol. 6 No. 2 April 2024 Hal. 286-296
http://jurnal.unidha.ac.id/index.php/jteksis

E-ISSN : **2655-8238**
P-ISSN : **2964-2132**

memiliki kemampuan untuk mengurutkan 500 data dalam waktu 0,3688 detik.

Hasil analisis pengujian menunjukkan bahwa metode yang menggunakan *stack* dan *array* untuk pengurutan lagu memiliki keunggulan dan kekurangan. Penggunaan *stack* untuk manajemen sementara dan *array* untuk akses cepat memungkinkan pengurutan yang efektif pada skala kecil hingga menengah. Contoh output menunjukkan kemampuan program untuk mengurutkan daftar lagu dalam waktu eksekusi yang singkat. Namun, waktu eksekusi meningkat secara signifikan saat jumlah lagu meningkat secara signifikan. Hal ini menunjukkan bahwa meskipun metode ini efisien pada skala kecil, kompleksitas waktu pemilihan *sort* menjadi penting saat berurusan dengan skala data yang besar.

Pengujian ini menunjukkan, bahwa penggunaan *stack* dan *array* dapat menawarkan keunggulan dalam manajemen dan akses data dalam konteks pengurutan lagu. Namun, sehubungan dengan banyak algoritma pengurutan lainnya, kelemahannya menjadi lebih jelas saat skala data meningkat. Hasil yang mengejutkan yaitu kompleksitas waktu yang signifikan saat jumlah lagu mencapai ribuan. Hasil ini menunjukkan bahwa penggunaan algoritma pengurutan atau pengembangan metode lain yang dapat mengatasi kompleksitas waktu yang meningkat harus dipertimbangkan ulang untuk skala data yang sangat besar.

Tujuan dari penelitian ini yaitu untuk mengetahui seberapa efektif penggunaan *stack* dan *array* dalam pengurutan lagu. Penemuan ini memberikan wawasan tentang keunggulan dan kekurangan implementasi algoritma pengurutan *selection sort* dengan struktur data tersebut. Pentingnya penelitian ini terletak pada kemungkinan penggunaan *stack* dan *array* dalam pengurutan, namun penemuan ini juga menekankan bahwa diperlukan alternatif yang lebih efisien saat menghadapi skala data yang besar. Penelitian masa depan mungkin melihat metode pengurutan lainnya atau pengoptimalan dalam penggunaan algoritma tertentu untuk pengurutan daftar lagu yang besar dengan lebih efisien, mempertimbangkan faktor kompleksitas waktu yang terungkap dalam penelitian ini.

SIMPULAN

Simpulan yang didapatkan dari percobaan ini yaitu bahwa *selection sort* juga mampu mengurutkan data yang bertipe *string* (kata/huruf). Dari program yang telah dibuat, hasilnya yaitu memudahkan pengguna dalam pencarian lagu hanya dengan memasukkan kata kunci saja, apakah itu dari nama artis/band-nya ataukah dari judul lagu itu sendiri. Selain itu, penampungan data lagu dalam proses seperti *stack* dan *array* juga memiliki keuntungan yaitu dapat dengan mudah mengakses lagu tertentu hanya dengan nilai indeks dari *array* tersebut. Selain itu, dapat disimpulkan juga bahwa algoritma *selection sort* sedikit lebih cepat daripada algoritma *insertion sort* berdasarkan penelitian yang telah dilakukan yaitu sebanyak 500 data dengan 5 kali pengujian.

UCAPAN TERIMAKASIH

Terima kasih kepada Bapak Dr. Eng. Munawir, MT. selaku Dosen pada Mata Kuliah Struktur Data dan Algoritma dan terima kasih juga kepada Kang Abdi Surya Perdana selaku Asisten Laboratorium yang telah mendampingi dan membimbing kami dalam penyusunan artikel ilmiah ini.

DAFTAR PUSTAKA

[1] Octaviani, S. & Nurfauziah, N., "Menelaah makna tersembunyi dalam lirik lagu 'Istirahat' Nosstress," *JURRIBAH*, *vol. 2*(1), 146-157, April, 2023, https://doi.org/10.55606/jurribah.v2i1.1152

[2] Mulyana, A. et al. (2021). *Cara mudah mempelajari algoritma dan struktur data* (Cetakan Pertama). DIVA Press. https://www.researchgate.net/publication/361248827_Cara_Mudah_Mempelajari_Algoritma_dan_Struktur_Data

[3] Retnoningsih, E., "Algoritma pengurutan data (sorting) dengan metode insertion sort dan selection sort," *INFORMATION MANAGEMENT FOR EDUCATORS AND PROFESSIONALS*, *vol. 3*(1), 95-106, Desember, 2018, http://101.255.92.196/index.php/IMBI/article/view/1060

[4] Sihombing, J., "Penerapan stack dan queue pada array dan linked list dalam java," *INFOKOM*, *vol. 7*(2), 15-24, Desember, 2019, https://www.journal.piksi.ac.id/index.php/INFOKOM/article/view/160

[5] Selamet, R., "Implementasi struktur data list, queue dan stack dalam java," *Media Informatika*, *vol. 15*(3), 18-25, 2016, https://jurnal.likmi.ac.id/Jurnal/11_2016/112016_03_Rachmat.pdf

[6] Santosa, R., Sari, P. A., & Sasongko, A. T., "Sistem monitoring suhu dan kelembaban berbasis IoT (Internet of thing) pada gudang penyimpanan PT Sakafarma Laboratories," *JTEKSIS*, *vol. 5*(4), 391-400, Oktober, 2023, https://doi.org/10.47233/jteksis.v5i4.943

[7] Hindriani, N., Narwen, & Yozza, H., "Implementasi antrian dengan menggunakan array," *Jurnal Matematika UNAND*, *vol. 3*(4), 147-151, 2014, https://doi.org/10.25077/jmu.3.4.147-151.2014

[8] Benardo, Mesterjon, & Zulita, L. N., "Implementasi metode selection sort untuk menentukan nilai prestasi siswa kelas 3 dan kelas 4 SD Negeri 107 Seluma", *Jurnal Media Infotama*, *vol. 11*(1), 91-100, Februari, 2015, https://doi.org/10.37676/jmi.v11i1.256

[9] Putra, J. D., Ananda, M. R., Syahputra, A., Nurshabillah, & Sinaga, S. P., "Pengurutan menggunakan metode selection sort pada sistem informasi lokasi kontrakan di kota Medan berbasis Android," *Blend Sains Jurnal Teknik*, *vol. 1*(4), 259-266, Januari, 2023, https://doi.org/10.56211/blendsains.v1i4.184

E-ISSN : **2655-8238**
P-ISSN : **2964-2132**

[10] Hakim, F. Z., "Implementasi metode selection sort untuk menentukan barang yang harus di stok ulang dalam sistem informasi penjualan," *Journal Information Engineering and Educational Technology, vol. 1*(1), 18-26, 2017, https://journal.unesa.ac.id/index.php/jieet/article/download/668/542

[11] Harahap, F. & Effendy, I., "Implementasi algoritma selection sort dalam membangun aplikasi android pemesanan jasa make-up Palembang," *Jurnal JUPITER, vol. 15*(1), 61-72, April, 2023, https://jurnal.polsri.ac.id/index.php/jupiter/article/download/5208/2586

[12] Farah A. D. E. & Kurniawati, D. O., "Penggunaan algoritma selection sort untuk menentukan nilai tertinggi siswa," *JURNAL SISTEM & TEKNOLOGI INFORMASI KOMUNIKASI, vol. 6*(2), 23-26, 2023, https://doi.org/10.32524/jusitik.v6i2.961

[13] Zailani, A. U., Apriyanto, B., & Zakaria, H. (2020). *Struktur data* (Cetakan Pertama). Unpam Press. https://repository.unpam.ac.id/8871/1/TPL0113_STRUKTUR%20DATA.pdf

[14] Maulana, R., "Analisa perbandingan kompleksitas algoritma selectionsort dan insertionsort," *INFORMATIKA, vol. 3*, 208-218, September, 2016, https://ejournal.bsi.ac.id/ejurnal/index.php/ji/article/view/810

[15] Sunandar, E., "Perbandingan metode selection sort dan insertion sort dalam pengurutan data menggunakan bahasa program java," *PETIR, vol. 12*(2), 172-178, September, 2019, https://doi.org/10.33322/petir.v12i2.485

[16] Kusumaningrum, A., "Perbandingan kecepatan antara selection sort, insertion sort, dan bubble sort," *TEKNOMATIKA, vol. 3*(1), 63-70, Juli, 2020, https://ejournal.unjaya.ac.id/index.php/teknomatika/article/download/363/311/

[17] Gultom, D. I., "Analisis perbandingan penggunaan algoritma pengurutan data dengan metoda bubble sort, metoda selection sort, metoda insertion sort," *Jurnal Tekinkom, vol. 1*(1), 8-13, Juni, 2018, https://doi.org/10.37600/tekinkom.v1i1.47

[18] Loebis, R. A. A., "Lagu, kaum muda dan budaya demokrasi," *PUSTAKA, vol. 18*(2), 81-85, Juni, 2020, https://doi.org/10.24843/PJIIB.2018.v18.i02.p02

[19] Hartanti, N. T., & Astuti, Y. (2018) *Algoritma struktur data.* Universitas AMIKOM Yogyakarta. http://materi.amikom.ac.id/2020/10/25102020_Modul%20Teori%20Algoritma%20Struktur%20Data.pdf

[20] Jadoon, S., Solehria, S. F., & Qayum, M., "Optimized selection sort slgorithm is faster than insertion Sort algorithm: a comparative study," *International Journal of Electrical & Computer Sciences IJECS-IJENS, vol. 11*(2), 18-23, 2011, https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=20960a569bacc4ef9d7a3266fc5e73dfa93c265c