LAPORAN APLIKASI E-HEALTH

STRUKTUR DATA DAN ALGORITMA (IF 204)

Dosen Pengampu :

Ir. Ahmad Chusyairi, S.Kom., M.Kom., IPM

Universitas
Siber Asia

Disusun Oleh

CATUR WIBOWO (240401020025)

ALIK SHODIKIN (240401020026)

KELAS IF 204

PROGRAM STUDI PJJ INFORMATIKA

UNIVERSITAS SIBER ASIA

JAKARTA

2025

DAFTAR ISI

BAB 1

PENDAHULUAN

1.1. LATAR BELAKANG

Dalam era digital seperti saat ini, kebutuhan akan sistem informasi yang cepat, akurat, dan efisien menjadi sangat penting, khususnya di sektor kesehatan. Rumah sakit, sebagai salah satu penyedia layanan kesehatan utama, memerlukan sistem yang mampu mengelola informasi secara terstruktur, mulai dari data fasilitas, tenaga medis, hingga informasi geografis dan konektivitas antar fasilitas. Teknologi informasi memberikan peluang besar untuk menciptakan sistem yang terintegrasi dan mudah diakses, baik oleh pengguna internal maupun masyarakat umum.

Salah satu pendekatan yang dapat digunakan adalah penerapan konsep e-Health, yaitu pemanfaatan teknologi informasi dan komunikasi (TIK) untuk mendukung layanan dan manajemen kesehatan. Melalui sistem ini, proses pencatatan, pencarian, dan pengelolaan informasi rumah sakit dapat dilakukan secara otomatis dan sistematis.

Untuk merealisasikan konsep tersebut, diperlukan pemahaman mendalam mengenai struktur data dan algoritma yang mendukung sistem informasi. Dalam hal ini, struktur data graf sangat cocok digunakan untuk merepresentasikan jaringan rumah sakit dan hubungan antar fasilitas. Dengan bantuan algoritma seperti Breadth-First Search (BFS) dan Depth-First Search (DFS), sistem dapat menelusuri jalur atau koneksi antar rumah sakit dengan efisien. Selain itu, algoritma searching dan sorting digunakan untuk mendukung proses pencarian data dan penyajian informasi yang terstruktur.

Makalah ini akan membahas perancangan dan implementasi sistem informasi rumah sakit berbasis C++ dengan pendekatan graf dan algoritma dasar, yang tidak hanya dapat menambah wawasan dalam pengembangan perangkat lunak, tetapi juga menjadi solusi awal dalam pengelolaan data fasilitas kesehatan yang lebih baik.

1.2. RUMUSAN MASALAH

1) Bagaimana membuat aplikasi yang mampu memetakan dan mengelola data rumah sakit beserta hubungan jarak antar rumah sakit?
2) Bagaimana mengimplementasikan algoritma Breadth First Search (BFS), Depth First Search (DFS), searching, dan sorting dalam aplikasi sistem informasi kesehatan?
3) Bagaimana menyajikan hasil aktivitas aplikasi secara otomatis dan sistematis untuk kepentingan dokumentasi?

1.3. TUJUAN

1) Mengembangkan aplikasi sistem informasi rumah sakit sederhana berbasis graph menggunakan bahasa pemrograman C++.
2) Mengimplementasikan dan mendemonstrasikan algoritma BFS, DFS, searching, dan sorting.
3) Membuat sistem pencatatan otomatis seluruh aktivitas aplikasi ke dalam laporan yang dapat diekspor.

1.4. PEMBAGIAN TUGAS KELOMPOK

Pekerjaan dalam proyek pembuatan aplikasi dan penulisan makalah ini dibagi secara adil dan proporsional antara anggota kelompok:

Catur Wibowo bertanggung jawab pada:

1) Perancangan struktur data aplikasi (graph, adjacency matrix).
2) Implementasi fitur algoritma BFS dan DFS pada kode program.
3) Penyusunan bagian latar belakang dan tujuan pada makalah.

Alik Sodikin bertanggung jawab pada:

1) Implementasi fitur searching, sorting, dan ekspor log otomatis pada aplikasi.
2) Penyusunan bagian pembahasan teknis, kesimpulan, dan saran pada makalah.

Selain itu, seluruh proses pengujian aplikasi, dokumentasi log aktivitas, serta penyusunan dan penyuntingan akhir makalah dilakukan bersama-sama secara kolaboratif oleh kedua anggota dengan pembagian beban pekerjaan yang sama.

BAB 2
LANDASAN TEORI

2.1 STRUKTUR DATA GRAPH

Graph adalah struktur data non-linear yang terdiri dari simpul (node) dan sisi (edge) yang menghubungkan pasangan simpul tersebut. Dalam pengembangan sistem e-Health, graph digunakan untuk merepresentasikan jaringan rumah sakit, di mana setiap simpul mewakili rumah sakit dan sisi mewakili koneksi atau jarak antar rumah sakit.

Graph dapat direpresentasikan dalam beberapa cara, salah satunya adalah adjacency matrix, yaitu matriks dua dimensi berukuran n x n (n adalah jumlah simpul) di mana setiap elemen matrix [i] dan [j] menunjukkan ada atau tidaknya koneksi antara simpul i dan j, beserta bobotnya (jarak dalam kilometer).

Kelebihan dari adjacency matrix adalah kemudahan dalam mengakses dan memodifikasi hubungan antar node, namun kelemahannya adalah penggunaan memori yang besar jika jumlah node sangat banyak dan koneksi sedikit (sparse graph).

2.2 IMPLEMENTASI BREADTH FIRST SEARCH (BFS)

Breadth First Search adalah algoritma penelusuran graf yang dimulai dari satu simpul dan menjelajahi semua simpul tetangganya terlebih dahulu sebelum berpindah ke simpul berikutnya. BFS menggunakan struktur data antrian (queue) untuk menyimpan node yang akan dikunjungi.

Pada aplikasi ini, BFS digunakan untuk menelusuri jaringan rumah sakit secara melebar dari satu rumah sakit ke rumah sakit lain yang saling terhubung. BFS cocok digunakan untuk mencari jalur terpendek dalam graf tidak berbobot dan untuk penelusuran yang membutuhkan hasil secara urut dari tingkat terdekat ke tingkat berikutnya.

2.3 IMPLEMENTASI DEPTH FIRST SEARCH (DFS)

Depth First Search adalah algoritma penelusuran graf yang menjelajahi simpul-simpul dalam satu jalur sedalam mungkin sebelum kembali dan menjelajahi

jalur lainnya. DFS menggunakan struktur data tumpukan (stack) atau bisa juga menggunakan rekursi.

DFS berguna untuk mengeksplorasi semua node dan jalur dalam graf, sangat efektif dalam menemukan komponen yang saling terhubung, serta cocok untuk kasus seperti deteksi siklus dalam graf atau memeriksa jalur tertentu.

2.4 ALGORITMA PENCARIAN (SEARCHING)

Searching merupakan proses untuk menemukan data tertentu dari sekumpulan data. Aplikasi ini menggunakan linear search, yaitu metode pencarian yang memeriksa elemen satu per satu dari awal hingga akhir.

Linear search mudah diimplementasikan dan tidak memerlukan data yang terurut, namun tidak efisien untuk jumlah data yang besar karena memiliki kompleksitas waktu O(n). Alternatif lain seperti binary search lebih cepat namun memerlukan data yang sudah terurut.

2.5 ALGORITMA PENGURUTAN (SORTING)

Sorting adalah proses menyusun data dalam urutan tertentu, seperti dari yang terkecil ke terbesar atau sebaliknya. Dalam aplikasi ini, sorting digunakan untuk mengurutkan rumah sakit berdasarkan rating.

Algoritma yang digunakan adalah Bubble Sort, algoritma sederhana yang bekerja dengan cara membandingkan dua elemen bersebelahan dan menukarnya jika urutannya salah. Bubble Sort memiliki kelebihan mudah diimplementasikan, namun memiliki kelemahan dari sisi efisiensi (kompleksitas waktu O(n^2)), sehingga tidak cocok untuk data dalam jumlah besar.

2.6 KOMPLEKSITAS ALGORITMA (BIG-O NOTATION)

Big-O Notation adalah cara untuk mengukur efisiensi algoritma dalam hal waktu dan penggunaan memori terhadap input yang diberikan. Notasi ini digunakan untuk membandingkan algoritma satu dengan yang lain berdasarkan skala besar. Berikut adalah kompleksitas algoritma yang digunakan dalam aplikasi e-Health:

- Penambahan Rumah Sakit: O(1)
- Penambahan Jarak (Edge): O(1)
- Menampilkan Daftar Rumah Sakit: O(n)
- Menampilkan Matriks Graf: O(n^2)

- Pencarian (Linear Search): O(n)
- Pengurutan (Bubble Sort): O(n^2)
- Penelusuran Graf (BFS dan DFS): O(V + E), dengan V adalah jumlah node dan E adalah jumlah edge.

Dengan pemahaman kompleksitas ini, kita dapat memperkirakan kinerja aplikasi seiring bertambahnya data rumah saki

BAB 3
PEMBAHASAN APLIKASI

3.1 BAHASA DAN TOOLS

1) **Bahasa Pemrograman C++**

C++ merupakan bahasa pemrograman yang banyak digunakan untuk membangun aplikasi yang membutuhkan efisiensi dan performa tinggi. Bahasa ini mendukung paradigma berorientasi objek, prosedural, dan generik, sehingga fleksibel untuk berbagai jenis aplikasi. Dalam konteks pengembangan aplikasi e-Health, C++ sangat sesuai karena memungkinkan pengelolaan memori yang presisi, kontrol penuh terhadap struktur data, serta efisiensi eksekusi yang dibutuhkan untuk memproses data graf dan algoritma seperti BFS dan DFS.

2) **Compiler : G++**

G++ adalah bagian dari GNU Compiler Collection (GCC) yang digunakan untuk mengompilasi kode C++. Compiler ini mendukung standar modern C++, termasuk C++11, C++14, C++17, dan seterusnya. Dalam proyek ini, G++ digunakan untuk menerjemahkan source code menjadi program yang dapat dijalankan di berbagai sistem operasi. G++ terkenal karena keandalannya, kecepatan proses kompilasi, serta kompatibilitas lintas platform.

3) **IDE (Integrated Development Environment): Visual Studio Code / Code::Blocks**

Visual Studio Code (VS Code) adalah text editor serbaguna yang mendukung berbagai bahasa pemrograman, termasuk C++. Dengan tambahan ekstensi seperti C/C++ dari Microsoft, pengguna dapat mengakses fitur seperti IntelliSense, debugging, dan integrasi terminal langsung. Code::Blocks adalah IDE khusus untuk bahasa C dan C++ yang ringan dan mudah digunakan, serta memiliki debugger dan kompiler terintegrasi. Keduanya memfasilitasi proses penulisan, debugging, dan kompilasi program dengan cara yang efisien dan terstruktur.

3.2 BAHASA PEMROGRAMAN

Program dirancang menggunakan paradigma pemrograman berorientasi objek (Object-Oriented Programming/OOP) untuk meningkatkan modularitas dan kemudahan pemeliharaan. Struktur program berpusat pada kelas EHealthSystem, yang bertindak sebagai pengelola utama data dan fungsi-fungsi terkait rumah sakit. Di dalam kelas ini, terdapat dua struktur data utama:

1. vector<RumahSakit> rsList: Menyimpan daftar rumah sakit sebagai objek, di mana setiap rumah sakit memiliki atribut nama, alamat, dan rating.
2. vector<vector<int>> graph: Menyimpan representasi jarak antar rumah sakit dalam bentuk adjacency matrix.

Setiap kali rumah sakit ditambahkan, program secara otomatis memperluas matriks graph agar node baru memiliki koneksi ke node-node sebelumnya. Aplikasi ini juga mencatat seluruh aktivitas dalam stringstream log, yang nantinya diekspor ke dalam file laporan_ehealth.txt saat aplikasi ditutup.

Fungsi-fungsi utama diimplementasikan sebagai metode dalam kelas, seperti tambahRS(), tambahJarak(), bfs(), dfs(), urutkanRS(), dan cariRS(). Hal ini memungkinkan pemisahan logika dan menjaga keterbacaan kode serta meningkatkan skalabilitas untuk pengembangan fitur selanjutnya. Program juga mengimplementasikan kontrol menu interaktif berbasis terminal yang memungkinkan pengguna memilih berbagai operasi sesuai kebutuhan

Penjelasan coding ehealth3new.cpp (diberikan komentar keterangan) sebagai berikut :

```
#include <iostream>
#include <vector>
#include <queue>
#include <stack>
#include <iomanip>
#include <fstream>
#include <sstream>
using namespace std;
struct RumahSakit {
    string nama;
    string alamat;
    int rating;
};
class EHealthSystem {
private:
    vector<RumahSakit> rsList;
    vector<vector<int>> graph;
    stringstream log;
public:
    void tambahRS(const string& nama, const string& alamat, int rating) {
        RumahSakit rs = {nama, alamat, rating};
        rsList.push_back(rs);
        for (auto& row : graph) row.push_back(0);
        graph.push_back(vector<int>(rsList.size(), 0));
        log << "[Tambah RS] Nama: " << nama << ", Alamat: " << alamat << ", Rating: " << rating << "\n";
        log << "Konsep: Menambah vertex pada graph (rumah sakit baru sebagai node).\n";
        log << "Big O: O(1) untuk tambah node dan update adjacency matrix.\n\n";
    }

    void tambahJarak(int dari, int ke, int jarak) {
```

```
        if (dari >= rsList.size() || ke >= rsList.size()) return;
        graph[dari][ke] = jarak;
        graph[ke][dari] = jarak;

        log << "[Tambah Edge] Jarak antar RS " << rsList[dari].nama << " <-> " << rsList[ke].nama
            << " = " << jarak << " KM\n";
        log << "Konsep: Menambah edge pada graph.\n";
        log << "Big O: O(1) untuk update adjacency matrix.\n\n";
    }

    void tampilkanRS() {
        log << "[Tampilkan RS] Daftar seluruh rumah sakit saat ini:\n";
        for (int i = 0; i < rsList.size(); i++) {
            cout << i << ". " << rsList[i].nama << " - " << rsList[i].alamat << " | Rating: " << rsList[i].rating << endl;
            log << i << ". " << rsList[i].nama << " - " << rsList[i].alamat << " | Rating: " << rsList[i].rating << endl;
        }
        log << "Konsep: Menampilkan seluruh node pada graph (list RS).\n";
        log << "Big O: O(n)\n\n";
    }

    void tampilkanMatriksGraf() {
        int n = rsList.size();
        cout << "\nMatriks Graf (Jarak antar Rumah Sakit dalam KM):\n\n";
        log << "[Tampilkan Matriks Graph] Matriks adjacency RS:\n";
        cout << setw(20) << " ";
        log << setw(20) << " ";
        for (int i = 0; i < n; i++) {
            cout << setw(20) << rsList[i].nama;
            log << setw(20) << rsList[i].nama;
        }
        cout << "\n";
        log << "\n";
        for (int i = 0; i < n; i++) {
            cout << setw(20) << rsList[i].nama;
            log << setw(20) << rsList[i].nama;
            for (int j = 0; j < n; j++) {
                cout << setw(20) << graph[i][j];
                log << setw(20) << graph[i][j];
            }
            cout << "\n";
            log << "\n";
        }
        log << "Konsep: Representasi graf menggunakan adjacency matrix (konsep graph).\n";
        log << "Big O: O(n^2)\n\n";
    }

    void bfs(int start) {
        vector<bool> visited(rsList.size(), false);
        queue<int> q;
        q.push(start);
        visited[start] = true;
        cout << "Hasil BFS:\n";
        log << "[BFS] Breadth First Search mulai dari " << rsList[start].nama << ":\n";
        while (!q.empty()) {
            int curr = q.front(); q.pop();
            cout << rsList[curr].nama << endl;
            log << rsList[curr].nama << "\n";
```

```
            for (int i = 0; i < rsList.size(); i++) {
                if (graph[curr][i] != 0 && !visited[i]) {
                    visited[i] = true;
                    q.push(i);
                }
            }
        }
        log << "Konsep: BFS pada graph untuk menjelajahi node secara melebar (menggunakan queue).\n";
        log << "Big O: O(V+E)\n\n";
    }

    void dfs(int start) {
        vector<bool> visited(rsList.size(), false);
        stack<int> s;
        s.push(start);
        cout << "Hasil DFS:\n";
        log << "[DFS] Depth First Search mulai dari " << rsList[start].nama << ":\n";
        while (!s.empty()) {
            int curr = s.top(); s.pop();
            if (!visited[curr]) {
                visited[curr] = true;
                cout << rsList[curr].nama << endl;
                log << rsList[curr].nama << "\n";
                for (int i = rsList.size() - 1; i >= 0; i--) {
                    if (graph[curr][i] != 0 && !visited[i]) {
                        s.push(i);
                    }
                }
            }
        }
        log << "Konsep: DFS pada graph untuk menjelajahi node secara mendalam (menggunakan stack).\n";
        log << "Big O: O(V+E)\n\n";
    }

    void cariRS(const string& nama) {
        log << "[Searching] Cari RS dengan nama '" << nama << "':\n";
        for (int i = 0; i < rsList.size(); i++) {
            if (rsList[i].nama == nama) {
                cout << "Ditemukan: " << rsList[i].nama << " - " << rsList[i].alamat
                     << " | Rating: " << rsList[i].rating << endl;
                log << "Ditemukan: " << rsList[i].nama << " - " << rsList[i].alamat
                    << " | Rating: " << rsList[i].rating << "\n";
                log << "Konsep: Searching menggunakan Linear Search.\n";
                log << "Big O: O(n)\n\n";
                return;
            }
        }
        cout << "Rumah sakit tidak ditemukan.\n";
        log << "Tidak ditemukan.\n";
        log << "Konsep: Searching menggunakan Linear Search.\n";
        log << "Big O: O(n)\n\n";
    }

    void urutkanRS(bool naik) {
        for (int i = 0; i < rsList.size() - 1; i++) {
            for (int j = 0; j < rsList.size() - i - 1; j++) {
                if ((naik && rsList[j].rating > rsList[j + 1].rating) ||
                    (!naik && rsList[j].rating < rsList[j + 1].rating)) {
```

```
                    swap(rsList[j], rsList[j + 1]);
                }
            }
        }
        log << "[Sorting] Pengurutan berdasarkan rating (" << (naik ? "naik" : "turun") << "):\n";
        cout << "\nHasil setelah diurutkan berdasarkan rating "
             << (naik ? "TERENDAH ke TERTINGGI:\n" : "TERTINGGI ke TERENDAH:\n");
        for (int i = 0; i < rsList.size(); i++) {
            cout << i << ". " << rsList[i].nama << " - " << rsList[i].alamat
                 << " | Rating: " << rsList[i].rating << endl;
            log << i << ". " << rsList[i].nama << " - " << rsList[i].alamat
                << " | Rating: " << rsList[i].rating << endl;
        }
        log << "Konsep: Sorting menggunakan Bubble Sort.\n";
        log << "Big O: O(n^2)\n\n";
    }

    void exportLaporan() {
        ofstream file("laporan_ehealth.txt");
        if (!file) {
            cout << "Gagal menulis laporan!\n";
            return;
        }
        file << "========== LAPORAN AKTIVITAS APLIKASI E-HEALTH ==========\n\n";
        file << "Konsep Graph: Node = Rumah Sakit, Edge = Jarak RS. Representasi: adjacency matrix.\n";
        file << "Breadth First Search (BFS): Penelusuran node graph secara melebar.\n";
        file << "Depth First Search (DFS): Penelusuran node graph secara mendalam.\n";
        file << "Searching: Linear search pada daftar node.\n";
        file << "Sorting: Bubble sort pada data RS.\n";
        file << "Big O Notation: Sudah dijelaskan pada setiap aktivitas berikut.\n\n";
        file << log.str();
        file.close();
        cout << "Laporan aktivitas telah diexport ke 'laporan_ehealth.txt'.\n";
    }
};

int main() {
    EHealthSystem system;

    // Data awal
    system.tambahRS("RS Sehat Selalu", "Jl. Merdeka 10", 95);
    system.tambahRS("RS Harapan Bunda", "Jl. Bambu 5", 90);
    system.tambahRS("RS Medika", "Jl. Kenanga 12", 92);
    system.tambahRS("RS Kasih Ibu", "Jl. Dahlia 2", 98);
    system.tambahRS("RS Permata", "Jl. Melati 9", 94);

    system.tambahJarak(0, 1, 3);
    system.tambahJarak(1, 2, 2);
    system.tambahJarak(2, 3, 4);
    system.tambahJarak(3, 4, 5);
    system.tambahJarak(0, 4, 6);

    int pilihan;
    do {
        cout << "\n=== Menu E-Health by Catur & Alik===\n";
        cout << "1. Tampilkan Rumah Sakit\n";
        cout << "2. Tambah Rumah Sakit\n";
        cout << "3. Cari Rumah Sakit\n";
```

```
            cout << "4. Urutkan Berdasarkan Rating\n";
            cout << "5. BFS Traversal\n";
            cout << "6. DFS Traversal\n";
            cout << "7. Tampilkan Matriks Graf\n";
            cout << "0. Keluar\n";
            cout << "Pilih: ";
            cin >> pilihan;
            cin.ignore();

            if (pilihan == 1) {
                system.tampilkanRS();
            } else if (pilihan == 2) {
                string nama, alamat;
                int rating;
                cout << "Nama RS   : "; getline(cin, nama);
                cout << "Alamat RS : "; getline(cin, alamat);
                cout << "Rating (90-99): "; cin >> rating;
                system.tambahRS(nama, alamat, rating);
            } else if (pilihan == 3) {
                string cari;
                cout << "Cari Nama RS: "; getline(cin, cari);
                system.cariRS(cari);
            } else if (pilihan == 4) {
                int urut;
                cout << "Urutkan berdasarkan rating:\n1. Terendah ke Tertinggi\n2. Tertinggi ke
Terendah\nPilih: ";
                cin >> urut;
                system.urutkanRS(urut == 1);
            } else if (pilihan == 5) {
                int start;
                cout << "Mulai dari indeks RS: ";
                cin >> start;
                system.bfs(start);
            } else if (pilihan == 6) {
                int start;
                cout << "Mulai dari indeks RS: ";
                cin >> start;
                system.dfs(start);
            } else if (pilihan == 7) {
                system.tampilkanMatriksGraf();
            }
        } while (pilihan != 0);

        system.exportLaporan(); // Panggil export otomatis di akhir program

        return 0;
    }
```

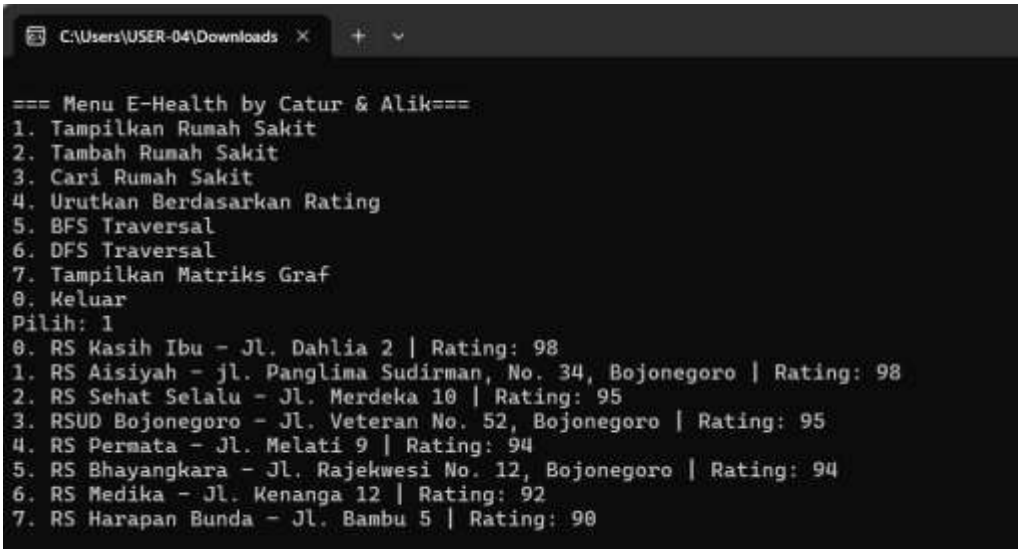
3.3 FITUR APLIKASI

1) Tambah Rumah Sakit

Fitur ini memungkinkan pengguna menambahkan data rumah sakit baru ke dalam sistem, mencakup nama, alamat, dan nilai rating. Setiap rumah sakit yang ditambahkan akan menjadi node baru dalam graf dan secara otomatis memperluas matriks adjacency.

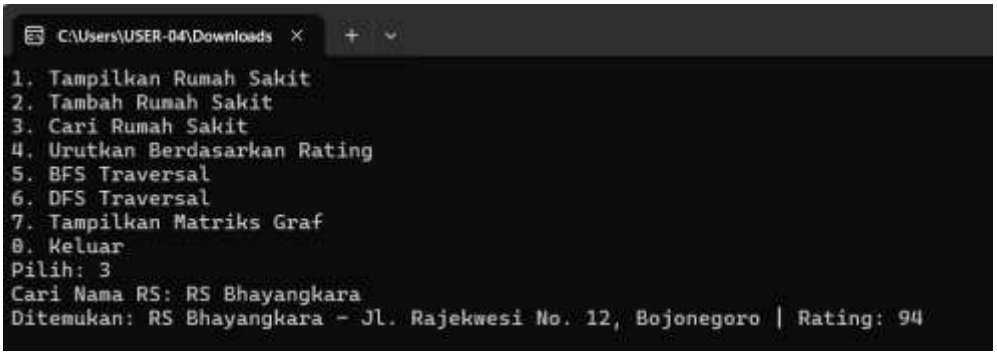
2) Tampilkan Daftar Rumah Sakit

Menampilkan seluruh daftar rumah sakit yang telah tersimpan dalam sistem. Data yang ditampilkan mencakup indeks, nama, alamat, dan rating. Fitur ini membantu pengguna memverifikasi data rumah sakit yang tersedia.



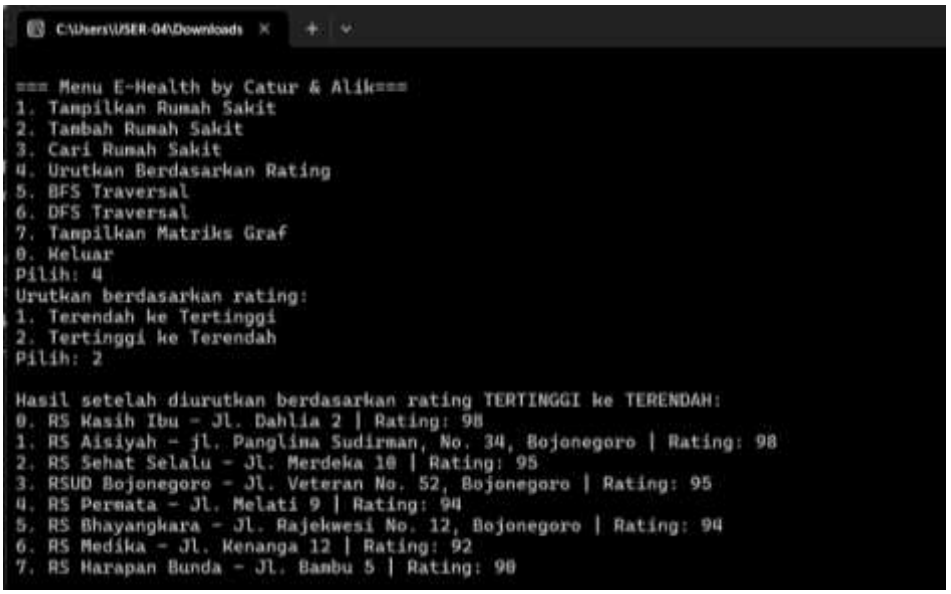
3) Pencarian Rumah Sakit (Linear Search)

Fitur ini memungkinkan pencarian rumah sakit berdasarkan nama. Sistem akan melakukan pencarian secara linear dari elemen pertama hingga akhir. Jika ditemukan, informasi lengkap rumah sakit akan ditampilkan.



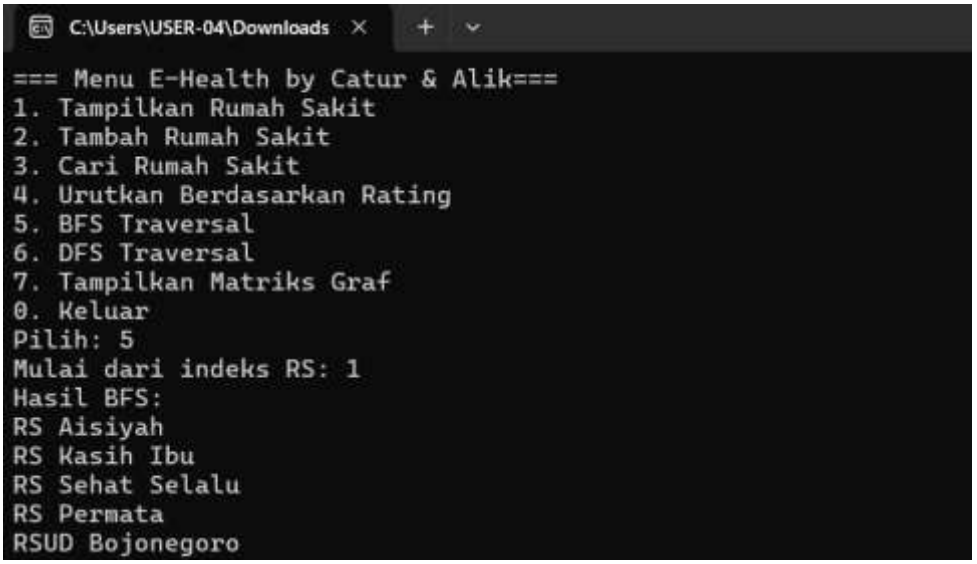
4) Pengurutan Rumah Sakit (Bubble Short)

Digunakan untuk mengurutkan rumah sakit berdasarkan nilai rating, baik dari nilai terendah ke tertinggi maupun sebaliknya. Pengurutan ini memudahkan pengguna dalam mengevaluasi kualitas rumah sakit secara cepat.



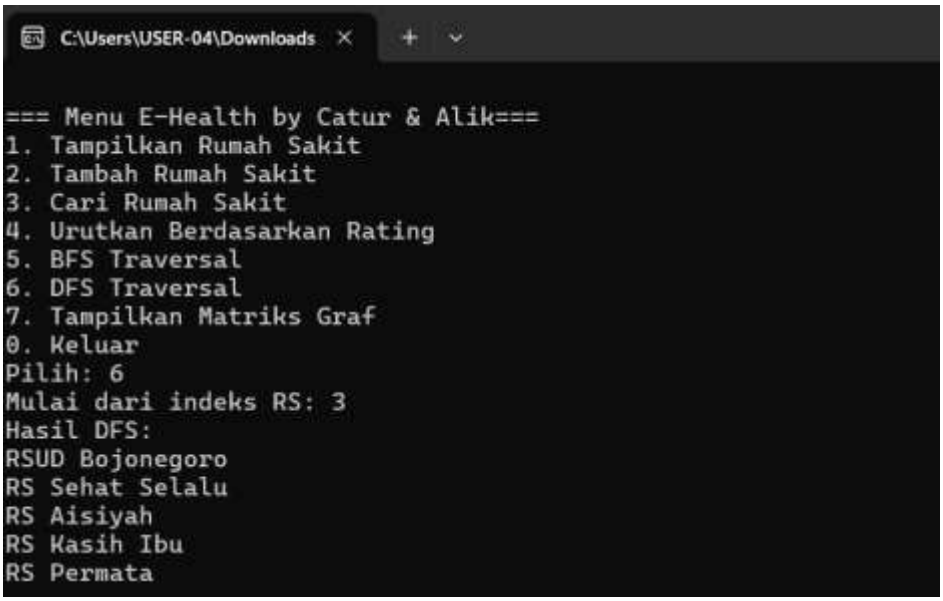
5) BFS Traversal

Breadth First Search digunakan untuk menelusuri rumah sakit dari satu node awal ke node lainnya yang terhubung. Traversal ini bersifat melebar dan cocok untuk eksplorasi data berdasarkan tingkat konektivitas



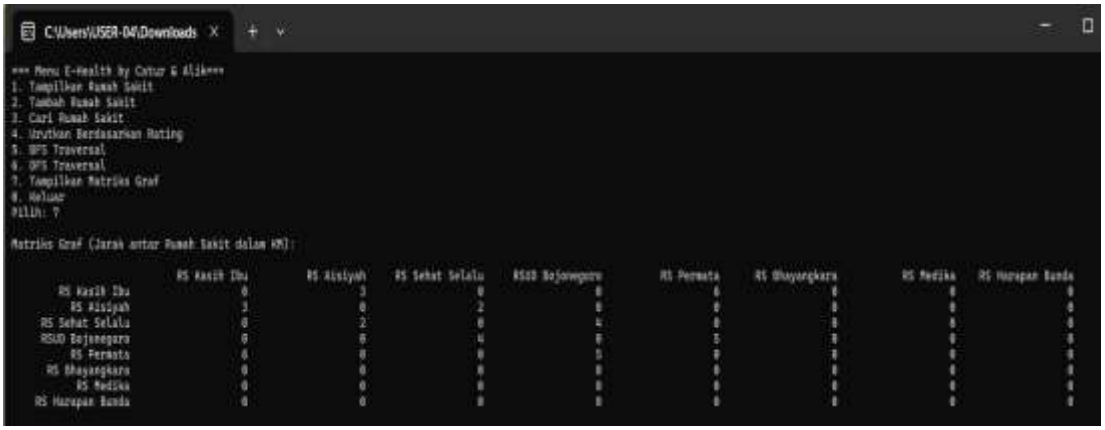
6) DFS Traversal

Depth First Search menjelajahi node secara mendalam dari titik awal. Algoritma ini berguna untuk menemukan jalur potensial atau koneksi yang lebih dalam antar rumah sakit.



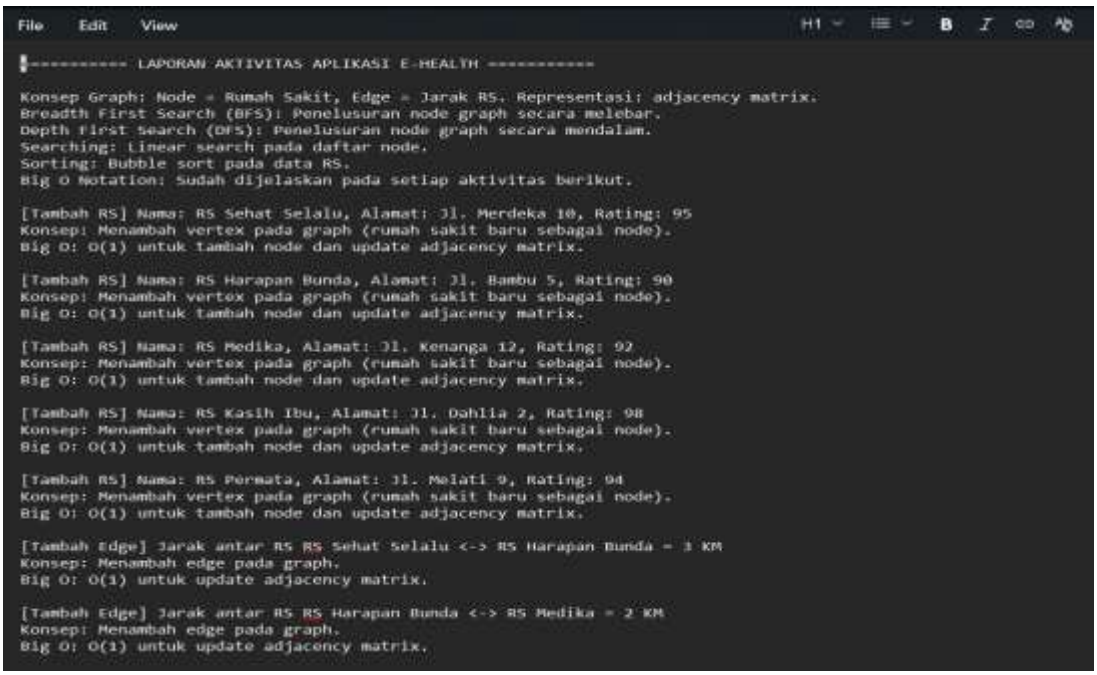
7) Tampilkan Matriks Graf

Menampilkan representasi visual dari matriks adjacency yang menggambarkan hubungan jarak antar rumah sakit. Matriks ini mencerminkan graf tidak berarah karena jarak antara RS A ke RS B dianggap sama dengan RS B ke RS A.



8) Ekspor Laporan

Semua aktivitas pengguna selama penggunaan aplikasi dicatat dalam stringstream log, dan secara otomatis diekspor ke file teks bernama laporan_ehealth.txt saat program diakhiri. Laporan ini berguna sebagai dokumentasi atau arsip proses pengolahan data.



Setiap fitur saling terkait dan mendukung satu sama lain dalam membangun sistem informasi yang menyeluruh. Pengguna dapat mengkombinasikan fitur-fitur tersebut untuk melakukan analisis, pencatatan, dan pemetaan koneksi rumah sakit secara digital.

BAB 4
PENUTUP

4.1 KESIMPULAN

Aplikasi e-Health berbasis C++ ini berhasil mengimplementasikan konsep graf, pencarian, pengurutan, dan traversal dengan efektif. Program ini dapat digunakan untuk pengelolaan rumah sakit dalam sistem informasi sederhana. Dengan pendekatan struktur data berbasis graf, aplikasi mampu memodelkan hubungan antar rumah sakit secara visual dan logis, sehingga memudahkan dalam analisis jaringan fasilitas kesehatan.

Implementasi algoritma BFS dan DFS menjadi landasan penting untuk menunjukkan bagaimana sistem dapat menjelajahi dan memetakan koneksi antar fasilitas secara efisien. Selain itu, pemanfaatan algoritma linear search dan bubble sort memberikan gambaran konkrit bagaimana operasi dasar seperti pencarian dan pengurutan dilakukan dalam struktur data. Seluruh aktivitas juga dicatat secara logis dalam bentuk laporan terstruktur yang dapat dijadikan dokumentasi penting.

Aplikasi ini memberikan fondasi yang kuat bagi pembelajaran struktur data dan pemrograman berbasis objek, sekaligus menjadi prototipe awal dari sistem informasi yang lebih kompleks dan profesional.

4.2 SARAN

Agar aplikasi ini lebih baik ke depannya, berikut beberapa saran pengembangan yang sederhana dan efektif:

1) Tambahkan fitur penyimpanan data menggunakan database agar informasi tetap tersimpan setelah program ditutup.
2) Buat tampilan antarmuka yang lebih mudah digunakan dengan GUI.
3) Gunakan algoritma sorting dan searching yang lebih cepat untuk meningkatkan kinerja aplikasi.

Dengan perbaikan tersebut, aplikasi e-Health ini bisa menjadi lebih praktis dan nyaman digunakan

Link Video Presentasi : https://youtu.be/G_Odx-q9QqY

Link Google Drive :

https://drive.google.com/drive/folders/17y4S4QQx6fXRw92pfiieG3wZaIF4jSGT

DAFTAR PUSTAKA

- Sedgewick, R., & Wayne, K. (2011). Algorithms (4th Edition). Addison-Wesley.
- Knuth, D. E. (1997). The Art of Computer Programming: Volume 1: Fundamental Algorithms. Addison-Wesley.
- Kurniawan, B. (2018). Pemrograman C++ untuk Pemula. Elex Media Komputindo.
- Documentation: cplusplus.com, GeeksforGeeks.org
- Materi Kuliah Struktur Data, Universitas.