MAKALAH

IMPLEMENTASI ALGORITMA SORTING, SEARCHING, DAN GRAPH DALAM STUDI KASUS NYATA

Disusun Untuk Memenuhi Tugas Mata Kuliah Struktur Data

Dosen Pengampu : Insidini Fawwaz M.Kom

Disusun Oleh Kelompok 5 :

Risky Jonalson Silaen	**241401010**
Nayla Syifa Tanjung	**241410019**
Doni Rivaldo Simamora	**241401037**
Muhammad Diaz WilliamBevan	**241401088**
Syahid Lukman	**241401112**
Farid Sani Ahdaputra	**241401127**

UNIVERSITAS SUMATERA UTARA

2025

KATA PENGANTAR

Puji dan syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa yang telah melimpahkan rahmat dan karunia-Nya sehingga kami dapat menyelesaikan makalah yang berjudul "Implementasi Algoritma Sorting, Searching, dan Graph dalam Studi Kasus Nyata". Makalah ini disusun guna memenuhi tugas kelompok untuk mata kuliah Struktur Data dengan dosen pengampu Ibu Insidini Fawwaz M.Kom.

Dalam penyusunan makalah ini, kami menyadari bahwa keberhasilan ini tidak terlepas dari bimbingan dan bantuan dari berbagai pihak. Untuk itu, kami ingin menyampaikan terima kasih kepada Ibu Insidini Fawwaz M.Kom, selaku dosen pengampu mata kuliah Struktur Data yang telah memberikan bimbingan dan ilmunya. Rekan-rekan Kelompok 5 yang telah berkontribusi dan bekerja sama dengan baik dalam penyelesaian program dan makalah ini.

Kami menyadari bahwa makalah ini masih jauh dari sempurna dikarenakan terbatasnya pengalaman dan pengetahuan yang kami miliki. Oleh karena itu, kami mengharapkan segala bentuk saran serta masukan bahkan kritik yang membangun dari berbagai pihak.

Akhir kata, kami berharap semoga makalah ini dapat memberikan manfaat dan menambah wawasan bagi pembaca, khususnya dalam pemahaman implementasi struktur data dan algoritma di dunia nyata.

Medan, 12 November 2025

Kelompok 2

DAFTAR ISI

BAB I
PENDAHULUAN

1.1. LATAR BELAKANG

Mikie Funland merupakan salah satu destinasi wisata taman hiburan terbesar dan terpopuler di Brastagi, Sumatera Utara. Dengan area yang sangat luas, taman ini menampung puluhan wahana, fasilitas, dan berbagai zona tematik yang berbeda, seperti Dream Zone, Adventure Zone, dan Atlantis Zone.

Bagi pengunjung, terutama yang baru pertama kali datang, navigasi di dalam area yang begitu kompleks dapat menjadi tantangan. Menemukan lokasi wahana tertentu, fasilitas umum seperti toilet atau musholla, atau sekadar menentukan rute terdekat dari satu tempat ke tempat lain dapat membuang waktu dan mengurangi kenyamanan berwisata. Pengunjung juga sering kali kesulitan untuk memutuskan wahana mana yang akan dicoba terlebih dahulu, atau mencari wahana yang sesuai dengan kriteria tertentu (misalnya, yang paling populer atau yang paling murah poin tiketnya).

Melihat permasalahan ini, dibutuhkan sebuah sistem informasi terpusat yang tidak hanya menyajikan peta, tetapi juga dapat memberikan solusi navigasi yang cerdas. Oleh karena itu, kami merancang sebuah program aplikasi konsol (CLI) berbasis C++ yang mengimplementasikan berbagai algoritma dari mata kuliah Struktur Data. Program ini berfungsi sebagai pemandu digital, yang mampu menghitung rute terpendek, mengurutkan wahana berdasarkan popularitas, dan mencari lokasi berdasarkan kriteria spesifik untuk membantu pengunjung merencanakan kunjungan mereka secara efisien.

1.2. RUMUSAN MASALAH

Berdasarkan latar belakang yang telah diuraikan, rumusan masalah dalam penulisan makalah ini adalah sebagai berikut:

1. Bagaimana merepresentasikan peta Mikie Funland yang kompleks, termasuk semua lokasi dan jarak antar lokasi, ke dalam struktur data komputasional yang logis?
2. Bagaimana algoritma *Graph* (graf) dapat diimplementasikan untuk menemukan rute terpendek dan paling efisien antara dua lokasi yang dipilih pengunjung?
3. Bagaimana algoritma *Sorting* (pengurutan) dapat diterapkan untuk membantu pengunjung memilih wahana dengan mengurutkan daftar lokasi berdasarkan kriteria seperti rating tertinggi atau poin termurah?
4. Bagaimana algoritma *Searching* (pencarian) dapat digunakan untuk memfilter dan menampilkan lokasi berdasarkan kategori, rentang rating, atau rentang poin tertentu?

1.3. TUJUAN PENULISAN

Adapun tujuan dari penulisan makalah dan perancangan program ini adalah:

1. Mengimplementasikan struktur data Graph Berbobot (Weighted Graph) untuk memetakan Mikie Funland, di mana lokasi menjadi vertex (simpul) dan jarak antar lokasi menjadi edge (sisi) dengan bobot tertentu.
2. Menerapkan Algoritma Dijkstra untuk menjalankan fitur pencarian rute terpendek (shortest path), sehingga program dapat memberikan rekomendasi navigasi yang efisien kepada pengguna.
3. Menerapkan Algoritma Merge Sort untuk menyediakan fitur pengurutan lokasi berdasarkan rating (secara descending) dan poin (secara ascending).
4. Menerapkan Algoritma Pencarian Linear (Linear Search) sebagai mekanisme filtering untuk menyaring dan menampilkan lokasi berdasarkan kategori, rating, atau poin yang diinginkan pengguna.
5. Membangun sebuah program C++ fungsional yang mengintegrasikan ketiga algoritma tersebut (Dijkstra, Merge Sort, dan Searching) ke dalam satu aplikasi pemandu yang koheren.

BAB II
TINJAUAN TEORI

2.1. SORTING (PENGURUTAN)

Sorting adalah proses fundamental dalam ilmu komputer untuk menyusun sekumpulan data atau objek ke dalam urutan tertentu (seperti numerik atau alfabetis), baik secara ascending (terkecil ke terbesar) maupun descending (terbesar ke terkecil). Tujuan utama dari pengurutan adalah untuk mengoptimalkan efisiensi pencarian data dan menyajikan data agar lebih mudah dibaca dan dianalisis oleh pengguna.

Terdapat banyak algoritma sorting, mulai dari yang sederhana seperti Bubble Sort dan Insertion Sort (dengan kompleksitas waktu rata-rata $O(n^2)$), hingga yang lebih kompleks dan efisien seperti Quick Sort dan Merge Sort (dengan kompleksitas waktu rata-rata O(n log n)). Beberapa algoritma pengurutan yang umum digunakan antara lain:

2.1.1. MERGE SORT

Dalam program ini, algoritma yang dipilih untuk fitur pengurutan adalah Merge Sort. Merge Sort adalah algoritma pengurutan yang menggunakan pendekatan *Divide and Conquer* (Bagi dan Kuasai).

Cara kerja Merge Sort dapat dibagi menjadi tiga langkah utama:

1. Divide (Membagi): Memecah array (atau vector) data utama menjadi dua sub-array yang berukuran sama secara rekursif, hingga setiap sub-array hanya berisi satu elemen (yang secara definisi sudah terurut).
2. Conquer (Menaklukkan): Mengurutkan setiap sub-array tersebut. Dalam implementasi mergeSort.cpp, langkah ini terjadi saat proses penggabungan.
3. Combine (Menggabungkan): Menggabungkan kembali sub-array yang sudah terurut menjadi satu array utuh yang terurut. Proses penggabungan ini membandingkan elemen dari kedua sub-array dan menempatkannya ke dalam array utama sesuai urutan yang benar.

Algoritma ini dipilih karena memiliki kompleksitas waktu O(n log n) yang stabil dan efisien, bahkan dalam skenario kasus terburuk. Ini jauh lebih cepat daripada $O(n^2)$, untuk mengurutkan 60 lokasi yang ada di dataset. Dalam implementasinya (mergeSort.cpp), algoritma ini digunakan untuk mengurutkan vector<string> berisi nama lokasi, dengan logika perbandingan (*compare*) yang diarahkan ke data di data_peta_lengkap untuk mengurutkan berdasarkan rating (tertinggi ke terendah) atau poin (termurah ke termahal).

2.2. SEARCHING

Searching adalah proses untuk menemukan keberadaan atau lokasi suatu data spesifik di dalam sekumpulan data. Dalam konteks program ini, "searching" diimplementasikan sebagai fitur Filtering (Penyaringan), yaitu mencari dan menampilkan semua data yang memenuhi kriteria tertentu.

2.2.1. LINEAR SEARCHING

Algoritma yang digunakan untuk fitur penyaringan (searchingAlgorithm.cpp) adalah Linear Search (atau Sequential Search). Ini adalah algoritma pencarian paling dasar yang bekerja dengan cara memeriksa setiap elemen dalam kumpulan data satu per satu, dari awal hingga akhir, sampai semua elemen telah diperiksa.

Algoritma ini digunakan karena:

1. Sederhana: Mudah diimplementasikan untuk berbagai kriteria.
2. Fleksibel: Dapat bekerja pada data yang tidak terurut. Misalnya, untuk mencari berdasarkan "rating > 4.5", data tidak perlu diurutkan berdasarkan rating terlebih dahulu.

Dalam implementasinya, program akan melakukan iterasi (menggunakan loop) pada seluruh map<string, Lokasi> data_peta_lengkap. Untuk setiap lokasi, program akan memeriksa apakah lokasi tersebut memenuhi kriteria yang dipilih pengguna:

- Berdasarkan Kategori: if (lokasi.tipe == "Atlantis Zone") ...
- Berdasarkan Rating: if (lokasi.rating > 4.5) ...
- Berdasarkan Poin: if (lokasi.poin >= 5 && lokasi.poin <= 10) ...

Lokasi yang memenuhi kriteria akan dikumpulkan dan ditampilkan kepada pengguna.

2.3. GRAPH

Graph (Graf) adalah struktur data non-linear yang digunakan untuk merepresentasikan hubungan (relasi) antar objek. Graf terdiri dari dua komponen utama:

1. Vertex (Simpul): Merepresentasikan objek atau entitas. Dalam program ini, vertex adalah semua lokasi di Mikie Funland (contoh: "Gerbang Masuk", "Tsunami", "Jungle Resto").
2. Edge (Sisi): Merepresentasikan koneksi atau jalur antar vertex.

Dalam program ini, edge adalah jalur/jalan yang menghubungkan dua lokasi.

- Weighted (Berbobot): Setiap edge memiliki nilai atau bobot yang merepresentasikan jarak (dalam meter) antara dua lokasi. Contoh: koneksi dari "Gerbang Masuk" ke "Ticketing" memiliki bobot 10.

- Undirected (Tak Berarah): Jalur antar lokasi dapat dilalui dari kedua arah. Jika ada jalur dari "Ticketing" ke "Gerbang Masuk", maka otomatis ada jalur dari "Gerbang Masuk" ke "Ticketing".

Representasi graf ini disimpan dalam dataset.hpp menggunakan map<string, Lokasi>, di mana string adalah nama *vertex* dan Lokasi berisi vector<Koneksi> yang merepresentasikan *edge* berbobot.

2.3.1. ALGORITMA DJIKSTRA

Untuk menemukan rute terpendek pada graf berbobot, algoritma yang paling umum dan efisien adalah Algoritma Dijkstra. Algoritma ini digunakan untuk menemukan jalur terpendek dari satu vertex awal (titik A) ke semua vertex lain dalam graf.

Cara kerja Algoritma Dijkstra (seperti dalam djikstraAlgorithm.cpp) adalah sebagai berikut:

1. Inisialisasi jarak dari lokasiAwal ke dirinya sendiri sebagai 0 dan ke semua lokasi lain sebagai tak terhingga (INF).
2. Membuat sebuah Priority Queue (Antrian Prioritas) untuk menyimpan pasangan {jarak, namaLokasi}. Priority queue ini sangat penting karena memastikan algoritma selalu memproses vertex terdekat yang belum dikunjungi.
3. Masukkan lokasiAwal ke priority queue dengan jarak 0.
4. Selama priority queue tidak kosong: a. Ambil (pop) vertex dengan jarak terkecil (u) dari queue. b. Jika u adalah lokasiTujuan, algoritma dapat berhenti (optimasi). c. Untuk setiap tetangga (v) dari u, hitung jarak baru: jarak(u) + bobot(u, v). d. Jika jarak baru ini lebih kecil dari jarak v yang tersimpan saat ini, perbarui jarak v dan masukkan v ke priority queue.
5. Setelah algoritma selesai, program akan memiliki jarak terpendek dari lokasiAwal ke lokasiTujuan serta data ruteSebelumnya untuk merekonstruksi jalurnya.

BAB III
PERANCANGAN DAN HASIL UJI COBA

3.1. FLOWCHART

3.2. KODE SUMBER DAN HASIL UJI COBA

3.1.1. KODE SUMBER

1. Main()

```
#include "djikstraAlgorithm.hpp"
#include "mergeSort.hpp"
#include "searchingAlgorithm.hpp"

void tampilkanSemuaLokasi();
void inputLokasi();

vector<string> namaLokasi;

MergeSort sorting;
DjikstraAlgorithm djikstra;
SearchingAlgorithm searching;

int main(){
   string pilih;
   int pilihanFitur;

   do{
      clearScreen();
      tampilkanSemuaLokasi();

      cout << "Pilihan fitur " << endl;
      cout << "[1]. Urutkan lokasi" << endl;
      cout << "[2]. Mulai pencarian" << endl;
      cout << "[3]. Cari rute terpendek" << endl;

      do{
         cout << "\nPilih fitur (1-3) : "; cin >> pilihanFitur;

         if (pilihanFitur == 1){
            sorting.jalankanSorting();
```

```
            }
            else if(pilihanFitur == 2){
                searching.jalankanSearching();
            }
            else if(pilihanFitur == 3){
                inputLokasi();
            }

            cout << "\n\nKembali ke Utama?[Y/N] "; cin >> pilih;
        } while (!validasiPilihan(1, 3, pilihanFitur));
    } while (pilih == "Y" || pilih == "y");
}

void inputLokasi(){
    int posisiAwal, tujuan;
    do{
        cout << "\nMasukkan angka sesuai dengan nomor kota di atas" << endl;
        cout << "Masukkan posisi anda sekarang : "; cin >> posisiAwal;
        cout << "Masukkan lokasi tujuan anda   : "; cin >> tujuan;

        } while (!validasiPilihan(1, namaLokasi.size(), posisiAwal) || !validasiPilihan(1,
namaLokasi.size(), tujuan));

    djikstra.cariRuteTerpendek(namaLokasi[posisiAwal - 1], namaLokasi[tujuan - 1]);
}

void tampilkanSemuaLokasi(){
    string garis(130, '-');

    gotoxy(0, 0); cout << garis;
    gotoxy(40, 1); cout << "--- Daftar Lokasi di Mikie Funland ---";
    gotoxy(0, 2); cout << garis;

    int i = 0, j = 1;
    int col = 0;
    int startY = 4;
    const int maxHeight = 14;
    const int maxWidth = 35;

    for(const auto& pasangan : data_peta_lengkap){
        gotoxy(col * maxWidth, i + startY);  cout << j << ". " << pasangan.first;
        namaLokasi.push_back(pasangan.first);

        if (i == maxHeight){
            i = -1;
            col++;
        }

        i++;
        j++;
```

```
    }
    gotoxy(0, maxHeight + 5); cout << endl;
}
```

2. Class DjikstraAlgorithm

```
#ifndef DJIKSTRAALGORITHM_HPP
#define DJIKSTRAALGORITHM_HPP

#include "dataSet.hpp"

typedef pair<int, string> jarakPair;

class DjikstraAlgorithm{
    private:
            static void cetakRute(const string& lokasiAwal, const string& lokasiTujuan,
map<string, string>& ruteSebelumnya);

    public:
        DjikstraAlgorithm();
        static void cariRuteTerpendek(const string& lokasiAwal, const string& lokasiTujuan);
};

#endif
```

```
#include "djikstraAlgorithm.hpp"

const int INF = numeric_limits<int>::max();

DjikstraAlgorithm::DjikstraAlgorithm(){}

void DjikstraAlgorithm::cariRuteTerpendek(const string& lokasiAwal, const string&
lokasiTujuan){
    map<string, int> dist;
    map<string, string> ruteSebelumnya;
    priority_queue<jarakPair, vector<jarakPair>, greater<jarakPair>> pq;

    for(const auto& pair : data_peta_lengkap){
        dist[pair.first] = INF;
    }

    dist[lokasiAwal] = 0;
    pq.push({0, lokasiAwal});

    while(!pq.empty()){
        int d = pq.top().first;
        string u = pq.top().second;
```

```
        pq.pop();

        if (d > dist.at(u)) continue;
        if (u == lokasiTujuan) break;

        for(const Koneksi& edge : data_peta_lengkap.at(u).daftarTetangga){
            string v = edge.namaTujuan;
            int w = edge.jarak;

            if(dist.at(u) + w < dist.at(v)){
                dist[v] = dist.at(u) + w;
                pq.push({dist.at(v), v});

                ruteSebelumnya[v] = u;
            }
        }
    }

    cout << "\n--- Hasil Rute ---" << endl;
    if(dist.at(lokasiTujuan) == INF){
        cout << "Tidak ada rute yang ditemukan dari " << lokasiAwal << " ke " << lokasiTujuan
<< endl;
    }
    else{
        cout << "Jarak terpendek : " << dist.at(lokasiTujuan) << " meter" << endl;
        cetakRute(lokasiAwal, lokasiTujuan, ruteSebelumnya);
    }
}

void DjikstraAlgorithm::cetakRute(const string& lokasiAwal, const string& lokasiTujuan,
map<string, string>& ruteSebelumnya){
    vector<string> rute;
    string saatIni = lokasiTujuan;

    while(saatIni != lokasiAwal){
        rute.push_back(saatIni);
        if(ruteSebelumnya.find(saatIni) == ruteSebelumnya.end()) break;
        saatIni = ruteSebelumnya.at(saatIni);
    }

    reverse(rute.begin(), rute.end());
    cout << "Rute           : "; cout << lokasiAwal << " -> ";
    for (size_t i = 0; i < rute.size(); ++i) {
        cout << rute[i];
        if (i < rute.size() - 1) {
            cout << " -> ";
        }
    }
    cout << endl;
}
```

3. Class MergeSort

```
#ifndef MERGESORT_HPP
#define MERGESORT_HPP

#include "dataSet.hpp"

class MergeSort{
    private:
        static bool compare(const string& a, const string& b, const string& mode);
       static void conquere(vector<string>& arr, int left, int mid, int right, const string& mode);
        static void devine(vector<string>& arr, int left, int right, const string& mode);
        static void printList(const string& judul, const vector<string>& namaDaftar);

    public:
        MergeSort();
        void jalankanSorting();
};

#endif
```

```
#include "mergeSort.hpp"

MergeSort::MergeSort(){}

void MergeSort::jalankanSorting(){
    vector<string> namaLokasi;
    for(const auto& pasangan : data_peta_lengkap){
        namaLokasi.push_back(pasangan.first);
    }

    int n = namaLokasi.size();
    int pilihan;
    string mode;

    do{
        cout << "\nIngin mengurutkan berdasarkan" << endl;
        cout << "[1]. Rating" << endl;
        cout << "[2]. Poin Tiket Termurah" << endl;

        cout << "\nPilihan : "; cin >> pilihan;
        mode = pilihan == 1 ? "rating" : "poin";
    } while (pilihan < 1 || pilihan > 2);

    cout << "Mengurutkan berdasarakan " << mode << endl;
    devine(namaLokasi, 0, n - 1, mode);
```

```
    string judul = (mode == "rating") ? "Hasil Sorting (Rating Tertinggi)" : "Hasil Sorting (Poin Termurah)";
    printList(judul, namaLokasi);
}

bool MergeSort::compare(const string& a, const string& b, const string& mode){
    if (mode == "rating") {
        float ratingA = data_peta_lengkap.at(a).rating;
        float ratingB = data_peta_lengkap.at(b).rating;
        return ratingA > ratingB;
    }
    else if (mode == "poin") {
        int poinA = data_peta_lengkap.at(a).poin;
        int poinB = data_peta_lengkap.at(b).poin;
        return poinA < poinB;
    }

    return a < b;
}

void MergeSort::conquere(vector<string>& arr, int left, int mid, int right, const string& mode){
    int n1 = mid - left + 1;
    int n2 = right - mid;

    vector<string> L(n1), R(n2);
    for (int i = 0; i < n1; i++) {
        L[i] = arr[left + i];
    }
    for (int j = 0; j < n2; j++) {
        R[j] = arr[mid + 1 + j];
    }

    int i = 0;
    int j = 0;
    int k = left;

    while (i < n1 && j < n2) {
        if (compare(L[i], R[j], mode)) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    while (i < n1) {
```

```
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

void MergeSort::devine(vector<string>& arr, int left, int right, const string& mode){
    if(left >= right) return;

    int mid = left + (right - left) / 2;
    devine(arr, left, mid, mode);
    devine(arr, mid + 1, right, mode);

    conquere(arr, left, mid, right, mode);
}

void MergeSort::printList(const string& judul, const vector<string>& namaDaftar){
    clearScreen();
    string garis(130, '-');

    gotoxy(0, 0); cout << garis;
    gotoxy(40, 1); cout << "--- " << judul << " ---";
    gotoxy(0, 2); cout << garis;

    int i = 0, j = 1;
    int col = 0;
    int startY = 4;
    const int maxHeight = 22;
    const int maxWidth = 28;

    for(const string& nama : namaDaftar){
        const Lokasi& data = data_peta_lengkap.at(nama);
        string n(to_string(abs(j)).length(), ' ');

        gotoxy(col * maxWidth, i + startY); cout << j << ". " << nama;
        gotoxy(col * maxWidth, startY + i + 1);
        cout << n << "  ( " << data.rating << " )" << " -- Poin " << data.poin;

        if (i == maxHeight){
            i = 0;
            col++;
        }
        else {
            i += 2;
```

```
        }

        j++;
    }
}
```

4. Class SearchingAlgorithm

```
#ifndef SEARCHING_HPP
#define SEARCHING_HPP

#include "dataSet.hpp"

class SearchingAlgorithm {
    private:
        template<typename T>
        static T bacaInputValid(const string& prompt) {
            T nilai;
            while (true) {
                cout << prompt;
                if (cin >> nilai) {
                    break;
                } else {
                    cout << "Input tidak valid. Silakan coba lagi.\n";
                    cin.clear();
                    cin.ignore(numeric_limits<streamsize>::max(), '\n');
                }
            }
            return nilai;
        }

    public:
        SearchingAlgorithm();
        static void jalankanSearching();
        static void aturOutput(string mode, vector<string>& tampilPilihan);
        static void kategori();
        static void rating();
        static void poin();
};

#endif
```

```
#include "searchingAlgorithm.hpp"

SearchingAlgorithm::SearchingAlgorithm(){}

void SearchingAlgorithm::jalankanSearching(){
    clearScreen();
```

```
    int pilih;

    vector<string> mode = {
        "kategori", "rating", "poin"
    };

    vector<function<void()>> fungsiMode = {
        []() { kategori(); },
        []() { rating(); },
        []() { poin(); }
    };

    string pilihan;
    do{

        for (size_t i = 0; i < mode.size(); i++){
            cout << "[" << i + 1 << "]. Cari Berdasarkan " << mode.at(i) << endl;
        }

        cout << "\nPilih ingin mencari apa : "; cin >> pilih;
        fungsiMode.at(pilih - 1)();

        cout << "\nIngin mencari lagi?[Y/N] "; cin >> pilihan;

        if (pilihan == "Y" || pilihan == "y") {
            clearScreen();
        }

    } while ((validasiPilihan(1, mode.size(), pilih)) && (pilihan == "Y" || pilihan == "y"));
}

void SearchingAlgorithm::kategori(){
    cout << "\n";
    vector<string> types = {
        "Fasilitas", "Dream Zone", "Adventure Zone", "Atlantis Zone"
    };

    int pilihan, i = 1;
    do{
        cout << "Anda memilih Kategori " << endl;
        for (size_t i = 0; i < types.size(); i++){
            cout << "[" << i + 1 << "]. " << types.at(i) << endl;
        }

        cout << "\nMasukkan pilihan tipe nya (1-" << types.size() << ") : "; cin >> pilihan;
        cout << "\nTempat dengan tipe " << types.at(pilihan - 1) << endl;

        for(const auto& tipes : data_peta_lengkap){
            if(tipes.second.tipe == types.at(pilihan - 1)){
```

```
            string n(to_string(abs(i)).length(), ' ');
            cout << i << ". " << tipes.first << endl;

            for(const auto& koneksi : tipes.second.daftarTetangga){
                cout << n << "  - Ke " << koneksi.namaTujuan
                    << " (" << koneksi.jarak << " meter)" << endl;
            }
            cout << endl;
            i++;
        }
    }
    break;
} while (validasiPilihan(1, types.size(), pilihan));

}

void SearchingAlgorithm::rating(){
    cout << "\n";

    vector<string> tampilPilihan = {
        "[1]. Cari dengan rating tertentu (misal: 4.5)",
        "[2]. Cari di atas rating tertentu (misal: > 4.5)",
        "[3]. Cari di bawah rating tertentu (misal: < 4.5)",
        "[4]. Cari di antara rating tertentu (misal: 4.0 - 4.5)",
    };

    aturOutput("rating", tampilPilihan);
}

void SearchingAlgorithm::poin(){
    cout << "\n";

    vector<string> tampilPilihan = {
        "[1]. Cari dengan poin tertentu (misal: 10)",
        "[2]. Cari di atas poin tertentu (misal: > 10)",
        "[3]. Cari di bawah poin tertentu (misal: < 10)",
        "[4]. Cari di antara poin tertentu (misal: 5 - 10)",
    };

    aturOutput("poin", tampilPilihan);
}

void SearchingAlgorithm::aturOutput(string mode, vector<string>& tampilPilihan){
    const float pembanding = 0.0001f;
    float rating1 = 0, rating2 = 0;
    int poin1 = 0, poin2 = 0;
    int i = 1;
    bool ditemukan = false;

    cout << "Anda memilih " << mode << endl;
```

```
    for(const auto& n : tampilPilihan){
        cout << n << endl;
    }

    int pilihan = 0;
    while(pilihan < 1 || pilihan > 4) {
        pilihan = bacaInputValid<int>("\nMasukkan pilihan pencarian (1-4) : ");
    }

    if (mode == "rating"){
        rating1 = bacaInputValid<float>("\nMasukkan rating : ");

        if(pilihan == 4) {
            rating2 = bacaInputValid<float>("Masukkan rating pembanding : ");
        }
    }
    else if (mode == "poin"){
        poin1 = bacaInputValid<int>("\nMasukkan poin : ");

        if(pilihan == 4) {
            poin2 = bacaInputValid<int>("Masukkan poin pembanding : ");
        }
    }

    cout << "\n--- Hasil Pencarian ---" << endl;
    for(const auto& tipes : data_peta_lengkap){
        if (mode == "rating") {
            if(pilihan == 1){
                if(abs(tipes.second.rating - rating1) < pembanding){
                    cout << i << ". " << tipes.first << " (Rating " << tipes.second.rating << ")" <<
endl;
                    ditemukan = true;
                    i++;
                }
            }
            else if(pilihan == 2){
                if(tipes.second.rating > rating1){
                    cout << i << ". " << tipes.first << " (Rating " << tipes.second.rating << ")" <<
endl;
                    ditemukan = true; i++;
                }
            }
            else if(pilihan == 3){
                if(tipes.second.rating < rating1){
                    cout << i << ". " << tipes.first << " (Rating " << tipes.second.rating << ")" <<
endl;
                    ditemukan = true; i++;
                }
            }
            else if(pilihan == 4){
```

```
                    if((tipes.second.rating >= rating1 && tipes.second.rating <= rating2) ||
(tipes.second.rating <= rating1 && tipes.second.rating >= rating2)){
                cout << i << ". " << tipes.first << " (Rating " << tipes.second.rating << ")" <<
endl;
                ditemukan = true; i++;
            }
        }
    }
    else if (mode == "poin") {
        if(pilihan == 1){
            if(tipes.second.poin == poin1){
                cout << i << ". " << tipes.first << " (" << tipes.second.poin << " Poin)" << endl;
                ditemukan = true; i++;
            }
        }
        else if(pilihan == 2){
            if(tipes.second.poin > poin1){
                cout << i << ". " << tipes.first << " (" << tipes.second.poin << " Poin)" << endl;
                ditemukan = true; i++;
            }
        }
        else if(pilihan == 3){
            if(tipes.second.poin < poin1){
                cout << i << ". " << tipes.first << " (" << tipes.second.poin << " Poin)" << endl;
                ditemukan = true; i++;
            }
        }
        else if(pilihan == 4){
            if(tipes.second.poin >= poin1 && tipes.second.poin <= poin2){
                cout << i << ". " << tipes.first << " (" << tipes.second.poin << " Poin)" << endl;
                ditemukan = true;
                i++;
            }
        }
    }
}

    if(!ditemukan)cout << "Tidak ada lokasi yang ditemukan dengan kriteria tersebut." <<
endl;
    cout << "--------------------------\n" << endl;
}
```

5. gotoXY

```
#ifndef GOTOXY_HPP
#define GOTOXY_HPP

#include <iostream>
#include <vector>
```

```
#include <string>
#include <map>
#include <cstdlib>
#include <queue>
#include <limits>
#include <algorithm>
#include <windows.h>
#include <cmath>
#include <functional>

using namespace std;

void gotoxy(int x, int y);
bool validasiPilihan(int awal, int akhir, int pilih);
void clearScreen();

#endif
```

```
#include "gotoXY.hpp"

void gotoxy(int x, int y) {
    COORD coord;
    coord.X = x;
    coord.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}

bool validasiPilihan(int awal, int akhir, int pilih){
    if(pilih > akhir || pilih < awal){
        cout << "Masukkan sesuai ketentuan (" << awal << ", " << akhir << ")" << endl;
        return false;
    }
    return true;
}

void clearScreen() {
    system("cls");
}
```

6. Dataset

```
#ifndef DATASET_HPP
#define DATASET_HPP

#include "gotoXY.hpp"
```

```
struct Koneksi {
    string namaTujuan;
    int jarak;
};

struct Lokasi {
    string tipe;
    float rating;
    int poin;
    vector<Koneksi> daftarTetangga;
};

inline map<string, Lokasi> data_peta_lengkap = {
    // --- Pintu Masuk & Fasilitas Utama ---
    {"Gerbang Masuk", {
        "Fasilitas", 4.0, 0, {
            {"Ticketing", 10}, {"Information", 30}, {"Titik Kumpul", 20}
        }
    }},
    {"Ticketing", {
        "Fasilitas", 4.0, 0, {
            {"Gerbang Masuk", 10}
        }
    }},
    {"Information", {
        "Fasilitas", 4.2, 0, {
            {"Gerbang Masuk", 30}, {"First Aid", 10}, {"ATM", 10}, {"Toilet", 15},
{"Souvenir Shop", 20}, {"Fun Zone", 40}
        }
    }},
    {"Toilet", {
        "Fasilitas", 3.8, 0, {
            {"Information", 15}
        }
    }},
    {"First Aid", {
        "Fasilitas", 4.0, 0, {
            {"Information", 10}
        }
    }},
    {"ATM", {       "Fasilitas", 4.0, 0, {
            {"Information", 10}
        }
    }},
    {"Souvenir Shop", {
        "Fasilitas", 4.1, 0, {
```

```
            {"Information", 20}, {"Penitipan Barang", 5}
        }
    }},
    {"Penitipan Barang", {
        "Fasilitas", 4.0, 0, {
            {"Souvenir Shop", 5}
        }
    }},
    {"Titik Kumpul", {
        "Fasilitas", 4.0, 0, {
            {"Gerbang Masuk", 20}
        }
    }},

    // --- Zona Mimpi (Dream Zone) ---
    {"Fun Zone", {
        "Dream Zone", 4.0, 5, {
            {"Information", 40}, {"Video Game", 10}, {"Happy Journey", 20}
        }
    }},
    {"Video Game", {
        "Dream Zone", 4.1, 5, {
            {"Fun Zone", 10}
        }
    }},
    {"Happy Journey", {
        "Dream Zone", 4.3, 10, {
            {"Fun Zone", 20}, {"Grasshopper", 15}
        }
    }},
    {"Grasshopper", {
        "Dream Zone", 4.4, 10, {
            {"Happy Journey", 15}, {"Twister", 30}
        }
    }},
    {"Twister", {
        "Dream Zone", 4.5, 10, {
             {"Grasshopper", 30}, {"Bumper Bee", 20}, {"Fly Up", 20}, {"Rolling
Stone", 30}
        }
    }},
    {"Rolling Stone", {
        "Dream Zone", 4.6, 15, {
            {"Twister", 30}, {"Star Theater", 20}, {"Jungle Resto", 60}
        }
    }},
```

```
    {"Star Theater", {
       "Dream Zone", 4.2, 5, {
          {"Rolling Stone", 20}
       }
    }},
    {"Butterfly", {
       "Dream Zone", 4.3, 10, {
          {"Bumper Bee", 25}
       }
    }},
    {"Fly Up", {
       "Dream Zone", 4.5, 10, {
          {"Twister", 20}
       }
    }},
    {"Bumper Bee", {
       "Dream Zone", 4.2, 5, {
          {"Twister", 20}, {"Butterfly", 25}
       }
    }},

    // --- Zona Petualangan (Adventure Zone) ---
    {"Jungle Resto", {
       "Fasilitas", 4.3, 0, {
            {"Rolling Stone", 60}, {"Koomba Dance", 20}, {"Dino Tracker", 25},
{"Charging Station", 5}, {"Simpang Atlantis Path", 80} // Path ke Atlantis
       }
    }},
    {"Koomba Dance", {
       "Adventure Zone", 4.4, 10, {
          {"Jungle Resto", 20}, {"Wild Ride", 15}
       }
    }},
    {"Wild Ride", {
       "Adventure Zone", 4.6, 15, {
          {"Koomba Dance", 15}
       }
    }},
    {"Carnival Games", {
       "Fasilitas", 3.9, 0, {
          {"Dino Tracker", 20}
       }
    }},
    {"Dino Tracker", {
       "Adventure Zone", 4.2, 5, {
```

```
            {"Jungle Resto", 25}, {"Dino Egg", 10}, {"Carnival Games", 20},
{"Volcano", 40}
        }
    }},
    {"Dino Egg", {
        "Adventure Zone", 4.0, 5, {
            {"Dino Tracker", 10}
        }
    }},
    {"T-Rex", {
        "Adventure Zone", 4.5, 10, {
            {"Buzz Coaster", 25}, {"Dino vs Dino", 15}, {"Raptor Show", 20}
        }
    }},
    {"Volcano", {
        "Adventure Zone", 4.7, 15, {
            {"Dino Tracker", 40}, {"Buzz Coaster", 30}, {"Jurassic Tree", 30}
        }
    }},
    {"Buzz Coaster", {
        "Adventure Zone", 4.4, 10, {
            {"Volcano", 30}, {"T-Rex", 25}
        }
    }},
    {"Dino vs Dino", {
        "Adventure Zone", 4.3, 10, {
            {"T-Rex", 15}
        }
    }},
    {"Flying Traveller", {
        "Adventure Zone", 4.4, 10, {
            {"Jurassic Tree", 25}
        }
    }},
    {"The Peak", {
        "Adventure Zone", 4.4, 10, {
            {"Jurassic Tree", 40}, {"Pterosaurs", 20}
        }
    }},
    {"Jurassic Tree", {
        "Adventure Zone", 4.6, 15, {
            {"Volcano", 30}, {"Flying Traveller", 25}, {"The Peak", 40}
        }
    }},
    {"Pterosaurs", {
        "Adventure Zone", 4.5, 10, {
```

```
        {"The Peak", 20}
    }
}},
{"Raptor Show", {
    "Adventure Zone", 4.2, 5, {
        {"T-Rex", 20}
    }
}},
{"Simpang Atlantis Path", {
    "Fasilitas", 0.0, 0, {
        {"Jungle Resto", 80}, {"Sea Monster", 30}, {"Musholla", 10}
    }
}},

// --- Zona Atlantis (Atlantis Zone) ---
{"Sea Monster", {
    "Atlantis Zone", 4.5, 10, {
        {"Simpang Atlantis Path", 30}, {"Sea Carousel", 15}, {"Flying Fish", 20}
    }
}},
{"Sea Carousel", {
    "Atlantis Zone", 4.1, 5, {
        {"Sea Monster", 15}, {"Bon Voyage", 10}
    }
}},
{"Flying Fish", {
    "Atlantis Zone", 4.4, 10, {
        {"Sea Monster", 20}
    }
}},
{"Happy Ocean", {
    "Atlantis Zone", 4.0, 5, {
        {"Dolphin Bay", 25}
    }
}},
{"Shark Attack", {
    "Atlantis Zone", 4.3, 10, {
        {"The Wave", 20}
    }
}},
{"Sea Horse Racer", {
    "Atlantis Zone", 4.2, 5, {
        {"Splash", 15}
    }
}},
{"Splash", {
```

```
      "Atlantis Zone", 4.3, 10, {
        {"Dolphin Bay", 30}, {"Sea Horse Racer", 15}
      }
   }},
   {"Jelly Swing", {
      "Atlantis Zone", 4.4, 10, {
        {"Dolphin Bay", 40}
      }
   }},
   {"Dolphin Bay", {
      "Atlantis Zone", 4.5, 10, {
         {"Simpang Atlantis Path", 60}, {"Splash", 30}, {"Happy Ocean", 25},
{"Jelly Swing", 40}
      }
   }},
   {"The Wave", {
      "Atlantis Zone", 4.8, 15, {
        {"Shark Attack", 20}, {"Tsunami", 30}
      }
   }},
   {"Ahooy!", {
      "Atlantis Zone", 4.0, 5, {
        {"Pirates Cafe", 30}
      }
   }},
   {"Pirates Cafe", {
      "Fasilitas", 4.1, 0, {
        {"The Falls", 20}, {"Ahooy!", 30}
      }
   }},
   {"The Falls", {
      "Atlantis Zone", 4.6, 10, {
        {"Simpang Atlantis Path", 40}, {"Pirates Cafe", 20}
      }
   }},
   {"Ocean Chair", {
      "Atlantis Zone", 4.2, 5, {
        {"Tsunami", 20}
      }
   }},
   {"Tsunami", {
      "Atlantis Zone", 4.9, 15, {
        {"The Wave", 30}, {"Ocean Chair", 20}
      }
   }},
   {"Bon Voyage", {
```

```
    "Atlantis Zone", 4.1, 5, {
        {"Sea Carousel", 10}
    }
}},
{"Sea Battle", {
    "Atlantis Zone", 4.4, 10, {
        {"The Breeze", 15}, {"Bubble Cab", 10}
    }
}},
{"The Breeze", {
    "Atlantis Zone", 4.3, 10, {
        {"Sea Battle", 15}
    }
}},
{"Bubble Cab", {
    "Atlantis Zone", 4.0, 5, {
        {"Sea Battle", 10}
    }
}},

// --- Fasilitas & Permainan Tambahan ---
{"Musholla", {
    "Fasilitas", 4.3, 0, {
        {"Simpang Atlantis Path", 10}
    }
}},
{"Beverage & Snack", {
    "Fasilitas", 3.9, 0, {
        {"Fun Zone", 30}
    }
}},
{"Gazebo", {
    "Fasilitas", 4.0, 0, {
        {"Jungle Resto", 30}
    }
}},
{"Charging Station", {
    "Fasilitas", 4.0, 0, {
        {"Jungle Resto", 5}
    }
}},
{"Archery", {
    "Fasilitas", 4.2, 0, {
        {"Koomba Dance", 40}
    }
}},
```

```
    {"Paint Ball", {
      "Fasilitas", 4.3, 0, {
        {"Koomba Dance", 45}
      }
    }}
};

#endif
```

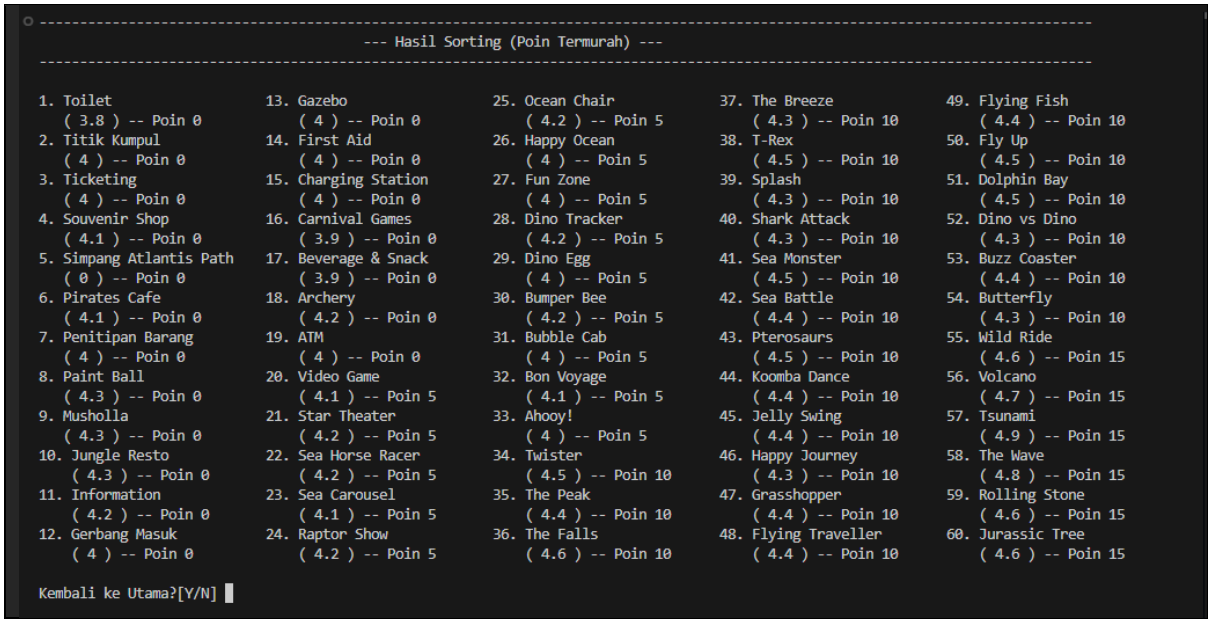
3.1.2. HASIL UJI COBA

1. Main()

```
--- Daftar Lokasi di Mikie Funland ---

1. ATM                      16. First Aid               31. Musholla                46. Splash
2. Ahooy!                   17. Fly Up                  32. Ocean Chair             47. Star Theater
3. Archery                  18. Flying Fish             33. Paint Ball              48. T-Rex
4. Beverage & Snack         19. Flying Traveller        34. Penitipan Barang        49. The Breeze
5. Bon Voyage               20. Fun Zone                35. Pirates Cafe            50. The Falls
6. Bubble Cab               21. Gazebo                  36. Pterosaurs              51. The Peak
7. Bumper Bee               22. Gerbang Masuk           37. Raptor Show             52. The Wave
8. Butterfly                23. Grasshopper             38. Rolling Stone           53. Ticketing
9. Buzz Coaster             24. Happy Journey           39. Sea Battle              54. Titik Kumpul
10. Carnival Games          25. Happy Ocean             40. Sea Carousel            55. Toilet
11. Charging Station        26. Information             41. Sea Horse Racer         56. Tsunami
12. Dino Egg                27. Jelly Swing             42. Sea Monster             57. Twister
13. Dino Tracker            28. Jungle Resto            43. Shark Attack            58. Video Game
14. Dino vs Dino            29. Jurassic Tree           44. Simpang Atlantis Path   59. Volcano
15. Dolphin Bay             30. Koomba Dance            45. Souvenir Shop           60. Wild Ride

Pilihan fitur
[1]. Urutkan lokasi
[2]. Mulai pencarian
[3]. Cari rute terpendek

Pilih fitur (1-3) :
```

2. MergeSort()

```
--- Hasil Sorting (Rating Tertinggi) ---

1. Tsunami
   ( 4.9 ) -- Poin 15
2. The Wave
   ( 4.8 ) -- Poin 15
3. Volcano
   ( 4.7 ) -- Poin 15
4. Wild Ride
   ( 4.6 ) -- Poin 15
5. The Falls
   ( 4.6 ) -- Poin 10
6. Rolling Stone
   ( 4.6 ) -- Poin 15
7. Jurassic Tree
   ( 4.6 ) -- Poin 15
8. Twister
   ( 4.5 ) -- Poin 10
9. T-Rex
   ( 4.5 ) -- Poin 10
10. Sea Monster
    ( 4.5 ) -- Poin 10
11. Pterosaurs
    ( 4.5 ) -- Poin 10
12. Fly Up
    ( 4.5 ) -- Poin 10
13. Dolphin Bay
    ( 4.5 ) -- Poin 10
14. The Peak
    ( 4.4 ) -- Poin 10
15. Sea Battle
    ( 4.4 ) -- Poin 10
16. Koomba Dance
    ( 4.4 ) -- Poin 10
17. Jelly Swing
    ( 4.4 ) -- Poin 10
18. Grasshopper
    ( 4.4 ) -- Poin 10
19. Flying Traveller
    ( 4.4 ) -- Poin 10
20. Flying Fish
    ( 4.4 ) -- Poin 10
21. Buzz Coaster
    ( 4.4 ) -- Poin 10
22. The Breeze
    ( 4.3 ) -- Poin 10
23. Splash
    ( 4.3 ) -- Poin 10
24. Shark Attack
    ( 4.3 ) -- Poin 10
25. Paint Ball
    ( 4.3 ) -- Poin 0
26. Musholla
    ( 4.3 ) -- Poin 0
27. Jungle Resto
    ( 4.3 ) -- Poin 0
28. Happy Journey
    ( 4.3 ) -- Poin 10
29. Dino vs Dino
    ( 4.3 ) -- Poin 10
30. Butterfly
    ( 4.3 ) -- Poin 10
31. Star Theater
    ( 4.2 ) -- Poin 5
32. Sea Horse Racer
    ( 4.2 ) -- Poin 5
33. Raptor Show
    ( 4.2 ) -- Poin 5
34. Ocean Chair
    ( 4.2 ) -- Poin 5
35. Information
    ( 4.2 ) -- Poin 0
36. Dino Tracker
    ( 4.2 ) -- Poin 5
37. Bumper Bee
    ( 4.2 ) -- Poin 5
38. Archery
    ( 4.2 ) -- Poin 0
39. Video Game
    ( 4.1 ) -- Poin 5
40. Souvenir Shop
    ( 4.1 ) -- Poin 0
41. Sea Carousel
    ( 4.1 ) -- Poin 5
42. Pirates Cafe
    ( 4.1 ) -- Poin 0
43. Bon Voyage
    ( 4.1 ) -- Poin 5
44. Titik Kumpul
    ( 4 ) -- Poin 0
45. Ticketing
    ( 4 ) -- Poin 0
46. Penitipan Barang
    ( 4 ) -- Poin 0
47. Happy Ocean
    ( 4 ) -- Poin 5
48. Gerbang Masuk
    ( 4 ) -- Poin 0
49. Gazebo
    ( 4 ) -- Poin 0
50. Fun Zone
    ( 4 ) -- Poin 5
51. First Aid
    ( 4 ) -- Poin 0
52. Dino Egg
    ( 4 ) -- Poin 5
53. Charging Station
    ( 4 ) -- Poin 0
54. Bubble Cab
    ( 4 ) -- Poin 5
55. Ahooy!
    ( 4 ) -- Poin 5
56. ATM
    ( 4 ) -- Poin 0
57. Carnival Games
    ( 3.9 ) -- Poin 0
58. Beverage & Snack
    ( 3.9 ) -- Poin 0
59. Toilet
    ( 3.8 ) -- Poin 0
60. Simpang Atlantis Path
    ( 0 ) -- Poin 0

Kembali ke Utama?[Y/N]
```


--- Hasil Sorting (Poin Termurah) ---
1. Toilet (3.8) -- Poin 0
2. Titik Kumpul (4) -- Poin 0
3. Ticketing (4) -- Poin 0
4. Souvenir Shop (4.1) -- Poin 0
5. Simpang Atlantis Path (0) -- Poin 0
6. Pirates Cafe (4.1) -- Poin 0
7. Penitipan Barang (4) -- Poin 0
8. Paint Ball (4.3) -- Poin 0
9. Musholla (4.3) -- Poin 0
10. Jungle Resto (4.3) -- Poin 0
11. Information (4.2) -- Poin 0
12. Gerbang Masuk (4) -- Poin 0
13. Gazebo (4) -- Poin 0
14. First Aid (4) -- Poin 0
15. Charging Station (4) -- Poin 0
16. Carnival Games (3.9) -- Poin 0
17. Beverage & Snack (3.9) -- Poin 0
18. Archery (4.2) -- Poin 0
19. ATM (4) -- Poin 0
20. Video Game (4.1) -- Poin 5
21. Star Theater (4.2) -- Poin 5
22. Sea Horse Racer (4.2) -- Poin 5
23. Sea Carousel (4.1) -- Poin 5
24. Raptor Show (4.2) -- Poin 5
25. Ocean Chair (4.2) -- Poin 5
26. Happy Ocean (4) -- Poin 5
27. Fun Zone (4) -- Poin 5
28. Dino Tracker (4.2) -- Poin 5
29. Dino Egg (4) -- Poin 5
30. Bumper Bee (4.2) -- Poin 5
31. Bubble Cab (4) -- Poin 5
32. Bon Voyage (4.1) -- Poin 5
33. Ahooy! (4) -- Poin 5
34. Twister (4.5) -- Poin 10
35. The Peak (4.4) -- Poin 10
36. The Falls (4.6) -- Poin 10
37. The Breeze (4.3) -- Poin 10
38. T-Rex (4.5) -- Poin 10
39. Splash (4.3) -- Poin 10
40. Shark Attack (4.3) -- Poin 10
41. Sea Monster (4.5) -- Poin 10
42. Sea Battle (4.4) -- Poin 10
43. Pterosaurs (4.5) -- Poin 10
44. Koomba Dance (4.4) -- Poin 10
45. Jelly Swing (4.4) -- Poin 10
46. Happy Journey (4.3) -- Poin 10
47. Grasshopper (4.4) -- Poin 10
48. Flying Traveller (4.4) -- Poin 10
49. Flying Fish (4.4) -- Poin 10
50. Fly Up (4.5) -- Poin 10
51. Dolphin Bay (4.5) -- Poin 10
52. Dino vs Dino (4.3) -- Poin 10
53. Buzz Coaster (4.4) -- Poin 10
54. Butterfly (4.3) -- Poin 10
55. Wild Ride (4.6) -- Poin 15
56. Volcano (4.7) -- Poin 15
57. Tsunami (4.9) -- Poin 15
58. The Wave (4.8) -- Poin 15
59. Rolling Stone (4.6) -- Poin 15
60. Jurassic Tree (4.6) -- Poin 15
Kembali ke Utama?[Y/N]

3. Searching()

```
[1]. Cari Berdasarkan kategori
[2]. Cari Berdasarkan rating
[3]. Cari Berdasarkan poin

Pilih ingin mencari apa : 2

Anda memilih rating
[1]. Cari dengan rating tertentu (misal: 4.5)
[2]. Cari di atas rating tertentu (misal: > 4.5)
[3]. Cari di bawah rating tertentu (misal: < 4.5)
[4]. Cari di antara rating tertentu (misal: 4.0 - 4.5)

Masukkan pilihan pencarian : 4

Masukkan rating : 4
Masukkan rating pembanding : 2

--- Hasil Pencarian ---
1. ATM (Rating 4)
2. Ahooy! (Rating 4)
3. Beverage & Snack (Rating 3.9)
4. Bubble Cab (Rating 4)
5. Carnival Games (Rating 3.9)
6. Charging Station (Rating 4)
7. Dino Egg (Rating 4)
8. First Aid (Rating 4)
9. Fun Zone (Rating 4)
10. Gazebo (Rating 4)
11. Gerbang Masuk (Rating 4)
12. Happy Ocean (Rating 4)
13. Penitipan Barang (Rating 4)
14. Ticketing (Rating 4)
15. Titik Kumpul (Rating 4)
16. Toilet (Rating 3.8)
---------------------------
```

4. DjikstraAlgorithm()

```
--- Daftar Lokasi di Mikie Funland ---

1. ATM                          16. First Aid                   31. Musholla                    46. Splash
2. Ahooy!                       17. Fly Up                      32. Ocean Chair                 47. Star Theater
3. Archery                      18. Flying Fish                 33. Paint Ball                  48. T-Rex
4. Beverage & Snack             19. Flying Traveller            34. Penitipan Barang            49. The Breeze
5. Bon Voyage                   20. Fun Zone                    35. Pirates Cafe                50. The Falls
6. Bubble Cab                   21. Gazebo                      36. Pterosaurs                  51. The Peak
7. Bumper Bee                   22. Gerbang Masuk               37. Raptor Show                 52. The Wave
8. Butterfly                    23. Grasshopper                 38. Rolling Stone               53. Ticketing
9. Buzz Coaster                 24. Happy Journey               39. Sea Battle                  54. Titik Kumpul
10. Carnival Games              25. Happy Ocean                 40. Sea Carousel                55. Toilet
11. Charging Station            26. Information                 41. Sea Horse Racer             56. Tsunami
12. Dino Egg                    27. Jelly Swing                 42. Sea Monster                 57. Twister
13. Dino Tracker                28. Jungle Resto                43. Shark Attack                58. Video Game
14. Dino vs Dino                29. Jurassic Tree               44. Simpang Atlantis Path       59. Volcano
15. Dolphin Bay                 30. Koomba Dance                45. Souvenir Shop               60. Wild Ride

Pilihan fitur
[1]. Urutkan lokasi
[2]. Mulai pencarian
[3]. Cari rute terpendek

Pilih fitur (1-3) : 3

Masukkan angka sesuai dengan nomor kota di atas
Masukkan posisi anda sekarang : 22
Masukkan lokasi tujuan anda   : 60

--- Hasil Rute ---
Jarak terpendek : 260 meter
Rute            : Gerbang Masuk -> Information -> Fun Zone -> Happy Journey -> Grasshopper -> Twister -> Rolling Stone -> Jungle Resto -> Koomba Dance
-> Wild Ride
```

BAB IV
ANALISIS HASIL

4.1. DESKRIPSI UMUM PROGRAM

Program ini adalah sebuah aplikasi pemandu lokasi Mikie Funland berbasis *Command Line Interface* (CLI) yang dibangun menggunakan C++. Program ini berfungsi sebagai sistem informasi interaktif yang mengimplementasikan tiga konsep utama Struktur Data: Graph Berbobot, Algoritma Sorting, dan Algoritma Searching.

Data utama program tersimpan dalam dataset.hpp, yang merepresentasikan 60 lokasi di Mikie Funland sebagai *graph* (peta). Program ini menyediakan tiga fungsionalitas utama kepada pengguna:

1. Mengurutkan Lokasi: Menggunakan Merge Sort untuk menampilkan daftar wahana berdasarkan Rating tertinggi atau Poin tiket termurah.
2. Mencari Lokasi: Menggunakan Linear Filtering (Sequential Search) untuk menyaring dan menampilkan wahana berdasarkan Kategori, Rating, atau Poin.
3. Mencari Rute Terpendek: Menggunakan Algoritma Dijkstra untuk menghitung dan menampilkan jarak terpendek (dalam meter) serta rute navigasi antara dua lokasi yang dipilih.

4.2. ANALISIS FITUR DAN ALGORITMA

Fitur-fitur program secara langsung mengimplementasikan materi Struktur Data:

1. Fitur 1: Urutkan Lokasi (Sorting)
   - Algoritma: Merge Sort (O(n log n)).
   - Fungsi: Mengurutkan 57 lokasi berdasarkan Rating Tertinggi (descending) atau Poin Termurah (ascending). Ini membantu pengunjung menemukan wahana terbaik atau termurah.

2. Fitur 2: Cari Lokasi (Searching/Filtering)
   - Algoritma: Pencarian Linear / Linear Filtering (O(n)).
   - Fungsi: Memfilter lokasi berdasarkan kriteria spesifik: Kategori (misal: "Atlantis Zone"), Rating (misal: > 4.5), atau Poin (misal: < 10). Ini efisien karena jumlah data (n=57) kecil.

3. Fitur 3: Cari Rute Terpendek (Graph)
   - Algoritma: Dijkstra (O(E log V)).
   - Fungsi: Menghitung jarak terpendek (dalam meter) dan rute navigasi antara dua lokasi. Program merepresentasikan peta sebagai weighted graph.

4.3. SIMULASI HASIL EKSEKUSI

Berikut adalah contoh output nyata berdasarkan kodingan dan dataset.hpp:

- Hasil Sorting (Rating): Menampilkan 1. Tsunami (4.9), 2. The Wave (4.8), 3. Volcano (4.7), dst.
- Hasil Searching (Kategori "Atlantis Zone"): Menampilkan Sea Monster, Sea Carousel, Flying Fish, dan 16 wahana Atlantis lainnya.
- Hasil Rute (Dijkstra) dari "Gerbang Masuk" ke "Bumper Bee":

```
--- Hasil Rute ---
Jarak terpendek : 155 meter
Rute            : Gerbang Masuk -> Information -> Fun Zone -> Happy Journey -> Grasshopper -> Twister -> Bumper Bee
```

4.4. ANALISIS ALGORITMA YANG DIGUNAKAN

Selama analisis, ditemukan bahwa beberapa lokasi (seperti Tsunami dan The Wave) terisolasi di dalam graph dan tidak terhubung ke jalur utama (Simpang Atlantis Path). Akibatnya, Algoritma Dijkstra akan melaporkan "Tidak ada rute yang ditemukan" saat mencoba mencapai lokasi-lokasi tersebut dari Gerbang Masuk, yang merupakan temuan penting tentang integritas data peta.

BAB V PENUTUP

5.1. KESIMPULAN

Permasalahan navigasi pengunjung di Mikie Funland dapat diselesaikan secara komputasional dengan mengimplementasikan struktur data Graf Berbobot (Weighted Graph). Dalam program ini, 57 lokasi direpresentasikan sebagai vertex (simpul), dan jalur penghubung antar lokasi direpresentasikan sebagai edge (sisi) yang memiliki bobot berupa jarak (meter).

Algoritma Dijkstra berhasil diterapkan untuk menemukan rute terpendek (shortest path) antara dua lokasi. Program mampu menghitung total jarak minimum dan menampilkan urutan vertex (rute) yang harus dilalui, serta mengidentifikasi jika tidak ada rute yang ditemukan.

Algoritma Merge Sort berhasil digunakan untuk menyediakan fungsionalitas tambahan "Urutkan Lokasi". Algoritma ini (O(n log n)) secara efisien mengurutkan daftar wahana berdasarkan kriteria dinamis, yaitu Rating Tertinggi (descending) atau Poin Termurah (ascending).

Algoritma Pencarian Linear (Filtering) berhasil diimplementasikan untuk fitur "Cari Lokasi". Program ini dapat menyaring (O(n)) dan menampilkan lokasi berdasarkan Kategori (tipe), Rating (rentang nilai), dan Poin (rentang nilai) yang spesifik.

Secara keseluruhan, program yang dibangun telah berhasil mengintegrasikan ketiga konsep (Sorting, Searching, dan Graph) dan memenuhi tujuan penulisan, yaitu menciptakan sebuah sistem informasi fungsional yang dapat membantu pengunjung Mikie Funland dalam hal navigasi dan pemilihan wahana.

5.2. SARAN

Untuk pengembangan program ini di masa depan, terdapat beberapa hal yang perlu diperhatikan. Validasi dan Perbaikan Dataset: Melakukan validasi ulang data dataset.hpp. Berdasarkan analisis, ditemukan beberapa lokasi (contoh: 'Tsunami' dan 'The Wave') yang terisolasi dari graph utama, sehingga rute tidak dapat ditemukan. Koneksi edge untuk lokasi-lokasi ini perlu ditambahkan agar peta menjadi utuh.

5.3. PEMBAGIAN TUGAS

1. Source code dan bab 4 : Doni Rivaldo Simamora
2. Bab 1 : Nayla Syifa Tanjung
3. Landasan teori : Risky Jonalson Silaen
4. Flowchart : Muhammad Diaz William Bevan
5. Buat laporan dan bab 5: Farid Sani Ahdaputra
6. Syahid Lukman (Tidak Memiliki tugas)