

Laporan Progress Tugas Besar Struktur Data - Ganjil 24/25



Oleh:

Ryan Gabriel Togar S. | 2311104045

Ammar Dzaki N. | 2311104071

Reza Afiansyah W. | 2311104062

Dosen :

Wahyu Andi Saputra, S.Pd., M.Eng.

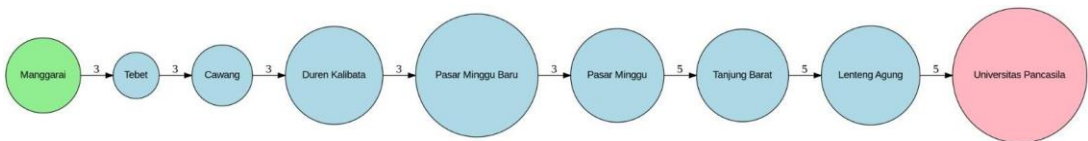
PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2023/2024

Membuat program graph untuk mencari rute tercepat atau terpendek jalur beberapa daerah stasiun di jakarta selatan dengan menggunakan algoritma djikstra

Deskripsi

Tugas besar ini bertujuan untuk mengimplementasikan struktur data dalam pengembangan program berbasis graph untuk mencari rute terpendek antara beberapa stasiun di wilayah Jakarta Selatan menggunakan algoritma Dijkstra. Program akan memanfaatkan representasi graph untuk memodelkan koneksi antar stasiun sebagai simpul (nodes) dan jarak antar stasiun sebagai bobot (weights) pada sisi (edges). Dengan algoritma Dijkstra, program dapat menentukan rute terpendek secara efisien berdasarkan input pengguna, seperti stasiun awal dan tujuan. Aplikasi ini diharapkan mampu memberikan solusi optimal dalam membantu pengguna menemukan jalur tercepat, baik untuk simulasi sistem transportasi maupun pengelolaan rute perjalanan.

Ilustrasi



Ilustrasi ini merepresentasikan graf perjalanan antar stasiun di Jakarta Selatan sebagaimana didefinisikan dalam program di atas. Setiap lingkaran menggambarkan sebuah stasiun, dan panah di antaranya menunjukkan hubungan langsung antara dua stasiun, dilengkapi dengan bobot berupa waktu perjalanan dalam menit.

- **Stasiun-Stasiun:**

- Stasiun "Manggarai" (hijau) adalah titik awal, sedangkan "Universitas Pancasila" (merah) adalah titik akhir. Stasiun lainnya, seperti "Tebet," "Cawang," "Duren Kalibata," "Pasar Minggu Baru," "Pasar Minggu," "Tanjung Barat," dan "Lenteng Agung," merupakan simpul-simpul yang menghubungkan jalur perjalanan.

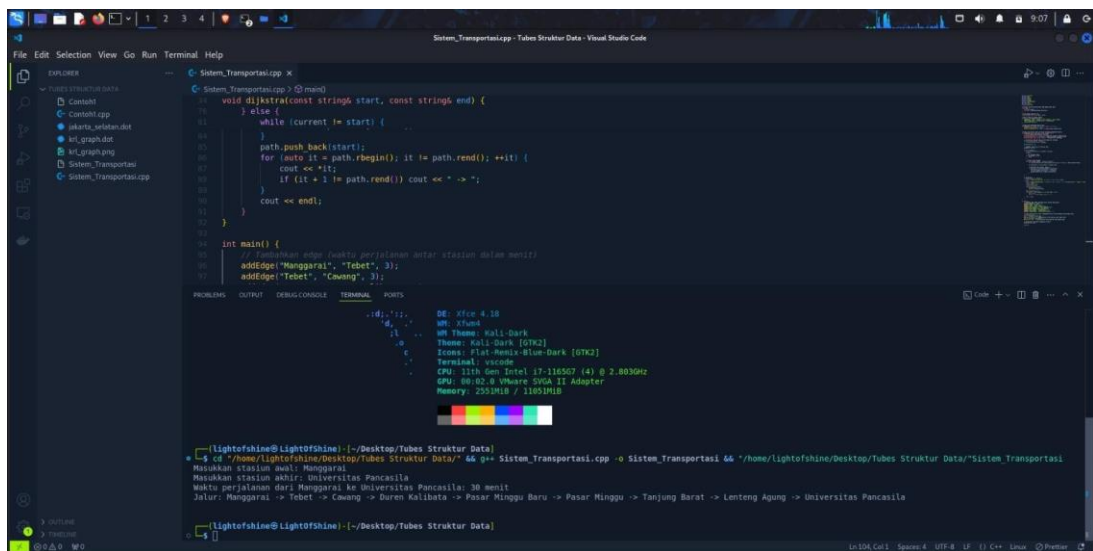
- **Bobot:**

- Angka di atas panah menunjukkan waktu perjalanan antar stasiun. Misalnya, waktu perjalanan dari "Manggarai" ke "Tebet" adalah 3 menit, dari "Pasar Minggu" ke "Tanjung Barat" adalah 5 menit, dan seterusnya.

- Arah:
 - Meskipun program menggunakan graf tak berarah (dua arah), ilustrasi ini menampilkan arah perjalanan yang dapat digunakan dalam simulasi.

Ilustrasi ini membantu memahami struktur graf dalam program serta memvisualisasikan rute terpendek yang dapat dihitung dengan algoritma Dijkstra berdasarkan input stasiun awal dan tujuan. Program akan menggunakan data seperti pada graf ini untuk menghitung waktu dan jalur perjalanan terpendek.

Hasil Running Script



```
void dijkstra(const strings start, const strings end) {
    while (current != start) {
        path.push_back(start);
        for (auto it = path.rbegin(); it != path.rend(); ++it) {
            cout << *it;
            if (it + 1 != path.rend()) cout << " -> ";
        }
        cout << endl;
    }
}

int main() {
    // Menentukan data untuk perjalanan antar stasiun dalam menit
    addEdge("Manggarai", "Tebet", 3);
    addEdge("Tebet", "Cawang", 3);
    // ... (other edges) ...

    dijkstra("Manggarai", "Universitas Pancasila");
}
```

DE: Xfce 4.10
WM: Xfwm
QT: Qt 5.12.0
Theme: Kali-Desktop [GTK2]
Icons: Flat-Remix-Blue-Dark [GTK2]
Terminal: xterm
CPU: 11th Gen Intel i7-110507 (4) @ 2.803GHz
GPU: 00:02.0 VMware SVGA II Adapter
Memory: 2509MiB / 3105MiB

lightofahne@lightofahne: ~/Desktop/Tubes Struktur Data
cd ~/home/lightofahne/Desktop/Tubes Struktur Data && g++ Sistem_Transportasi.cpp -o Sistem_Transportasi && ./Sistem_Transportasi
Masukkan stasiun awal: Manggarai
Masukkan stasiun akhir: Universitas Pancasila
Waktu perjalanan dari Manggarai ke Universitas Pancasila: 30 menit
Jalur: Manggarai -> Tebet -> Cawang -> Duren Kalibata -> Pasar Minggu Baru -> Pasar Minggu -> Tanjung Barat -> Lenteng Agung -> Universitas Pancasila

Program ini dirancang untuk menghitung waktu perjalanan terpendek antar stasiun di Jakarta Selatan menggunakan algoritma Dijkstra. Output program ini terdiri dari dua bagian utama:

Waktu Perjalanan: Program akan mencetak waktu perjalanan terpendek dari stasiun awal ke stasiun tujuan dalam hitungan menit. Jika tidak ada jalur yang menghubungkan kedua stasiun tersebut, program akan mencetak pesan bahwa tidak ada jalur yang tersedia.

Rute Perjalanan: Program juga akan mencetak rute perjalanan terpendek dalam bentuk daftar nama stasiun yang harus dilewati, dimulai dari stasiun awal hingga stasiun tujuan. Rute ini akan membantu pengguna memahami jalur yang harus diambil untuk mencapai tujuan dengan waktu paling efisien.

Sebagai contoh, jika pengguna memasukkan stasiun awal "Manggarai" dan stasiun tujuan "Universitas Pancasila", program akan menampilkan waktu perjalanan total dalam menit, diikuti oleh jalur lengkap yang mencakup setiap stasiun di sepanjang rute. Jika input stasiun berada di luar daftar stasiun Jakarta Selatan atau tidak ada jalur yang menghubungkan kedua stasiun, program akan memberikan notifikasi yang sesuai.

Isi Seluruh Script

```
#include <iostream>

#include <map>

#include <vector>

#include <queue>

#include <string>

#include <unordered_map>

#include <set>

#include <climits>

using namespace std;

// Struktur untuk merepresentasikan edge dengan bobot waktu

struct Edge {

    string destination;

    int time; // Waktu perjalanan dalam menit

};

// Graf sebagai adjacency list
```

```
unordered_map<string, vector<Edge>> graph;

// Fungsi untuk menambahkan edge ke graf

void addEdge(const string& from, const string& to, int time) {

    graph[from].push_back({to, time});

    graph[to].push_back({from, time}); // Karena grafnya tidak berarah
}

// Fungsi untuk mencari waktu perjalanan terpendek menggunakan Dijkstra

void dijkstra(const string& start, const string& end) {

    // Priority queue untuk Dijkstra (min-heap)

    priority_queue<pair<int, string>, vector<pair<int, string>>, greater<>> pq;

    unordered_map<string, int> distances; // Waktu terpendek ke setiap stasiun

    unordered_map<string, string> previous; // Melacak jalur sebelumnya

    for (const auto& station : graph) {

        distances[station.first] = INT_MAX;

    }

    distances[start] = 0;

    pq.push({0, start});
```

```
while (!pq.empty()) {  
    auto [currentDistance, currentNode] = pq.top();  
    pq.pop();  
  
    // Jika mencapai tujuan  
    if (currentNode == end) {  
        break;  
    }  
  
    // Periksa semua tetangga  
    for (const Edge& neighbor : graph[currentNode]) {  
        int newDistance = currentDistance + neighbor.time;  
  
        // Jika waktu lebih singkat, perbarui  
        if (newDistance < distances[neighbor.destination]) {  
            distances[neighbor.destination] = newDistance;  
            previous[neighbor.destination] = currentNode;  
            pq.push({newDistance, neighbor.destination});  
        }  
    }  
}
```

```
// Cetak hasil

if (distances[end] == INT_MAX) {

    cout << "Tidak ada jalur dari " << start << " ke " << end << endl;

} else {

    cout << "Waktu perjalanan dari " << start << " ke " << end << ": " <<
distances[end] << " menit" << endl;

    cout << "Jalur: ";

    string current = end;

    vector<string> path;

    while (current != start) {

        path.push_back(current);

        current = previous[current];

    }

    path.push_back(start);

    for (auto it = path.rbegin(); it != path.rend(); ++it) {

        cout << *it;

        if (it + 1 != path.rend()) cout << " -> ";

    }

    cout << endl;

}

}

int main() {

    // Tambahkan edge (waktu perjalanan antar stasiun dalam menit)
```

```
addEdge("Manggarai", "Tebet", 3);

addEdge("Tebet", "Cawang", 3);

addEdge("Cawang", "Duren Kalibata", 3);

addEdge("Duren Kalibata", "Pasar Minggu Baru", 3);

addEdge("Pasar Minggu Baru", "Pasar Minggu", 3);

addEdge("Pasar Minggu", "Tanjung Barat", 5);

addEdge("Tanjung Barat", "Lenteng Agung", 5);

addEdge("Lenteng Agung", "Universitas Pancasila", 5);


// Input stasiun awal dan akhir (menggunakan getline untuk mendukung input
dengan spasi)

string start, end;

cout << "Masukkan stasiun awal: ";

getline(cin, start);

cout << "Masukkan stasiun akhir: ";

getline(cin, end);


// Hitung waktu perjalanan menggunakan Dijkstra

dijkstra(start, end);

return 0;

}
```


Link Video :

<https://drive.google.com/drive/folders/1ugTjEPN02OdAPYGfgW52HeKkBhbhVOgW?usp=sharing>

Pembagian Tugas Anggota Kelompok

Gabriel:

- Bertanggung jawab dalam perancangan struktur graph.
- Mengimplementasikan bagian awal program, termasuk pembuatan simpul (node) dan jalur (edge) pada graph.
- Membantu Reza dalam memastikan algoritma Dijkstra berjalan dengan baik.

Reza:

- Bertanggung jawab untuk mengimplementasikan algoritma Dijkstra dalam program.
- Mengintegrasikan fungsi pencarian rute terpendek ke dalam sistem.
- Melakukan pengujian program untuk memastikan hasil pencarian rute sudah sesuai.

Ammar:

- Bertugas membuat laporan proyek secara menyeluruh, termasuk pendahuluan, landasan teori, penjelasan implementasi, hasil pengujian, dan kesimpulan.
- Mendokumentasikan diagram graph, penjelasan algoritma, serta hasil pengujian program.
- Mengompilasi semua tugas anggota ke dalam laporan akhir.