LAPORAN TUGAS BESAR

IMPLEMENTASI GRAPH PENGELOLAAN JARINGAN TRANSMISI LISTRIK DI DAERAH YOGYAKARTA

Disusun Oleh:
AJI PRASETYO NUGROHO - 2211104049
DHIYA ULHAQ RAMADHAN -2211104053
MUHAMMAD RALFI - 2211104054

SE-07-2

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO

2024

1. **Tujuan**
   a. Mahasiswa dapat memahami konsep Graph
   b. Mahasiswa mampu mendefinisikan tentang Graph pada pemrograman
   c. Mahasiswa dapat mengimplementasikan konsep Graph pada pemrograman

2. **Deskripsi Graph**

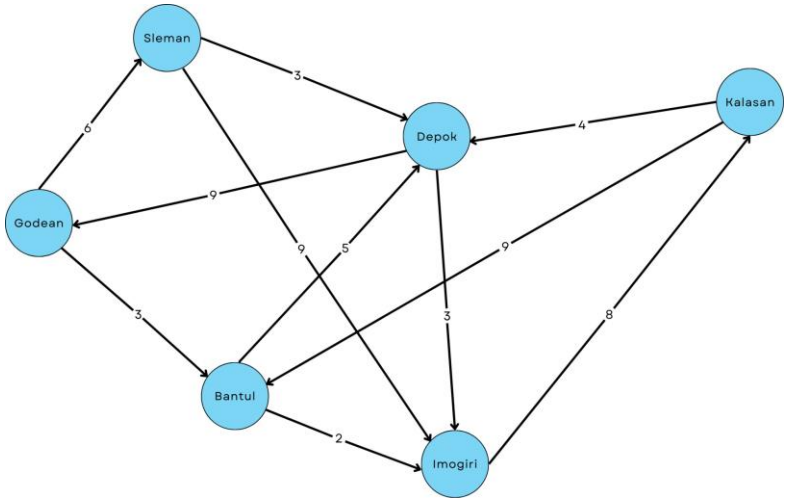
   Graph merupakan salah satu struktur data dalam bidang ilmu komputer dan matematika diskrit yang terdiri dari kumpulan simpul (node atau vertex) dan sisi (edge) yang menghubungkan simpul-simpul tersebut. Graph digunakan untuk merepresentasikan hubungan atau koneksi antar objek. Secara formal, graph dapat dinyatakan sebagai pasangan , di mana:

   1. adalah himpunan tidak kosong dari simpul.
   2. adalah himpunan dari sisi yang menghubungkan dua simpul dalam.

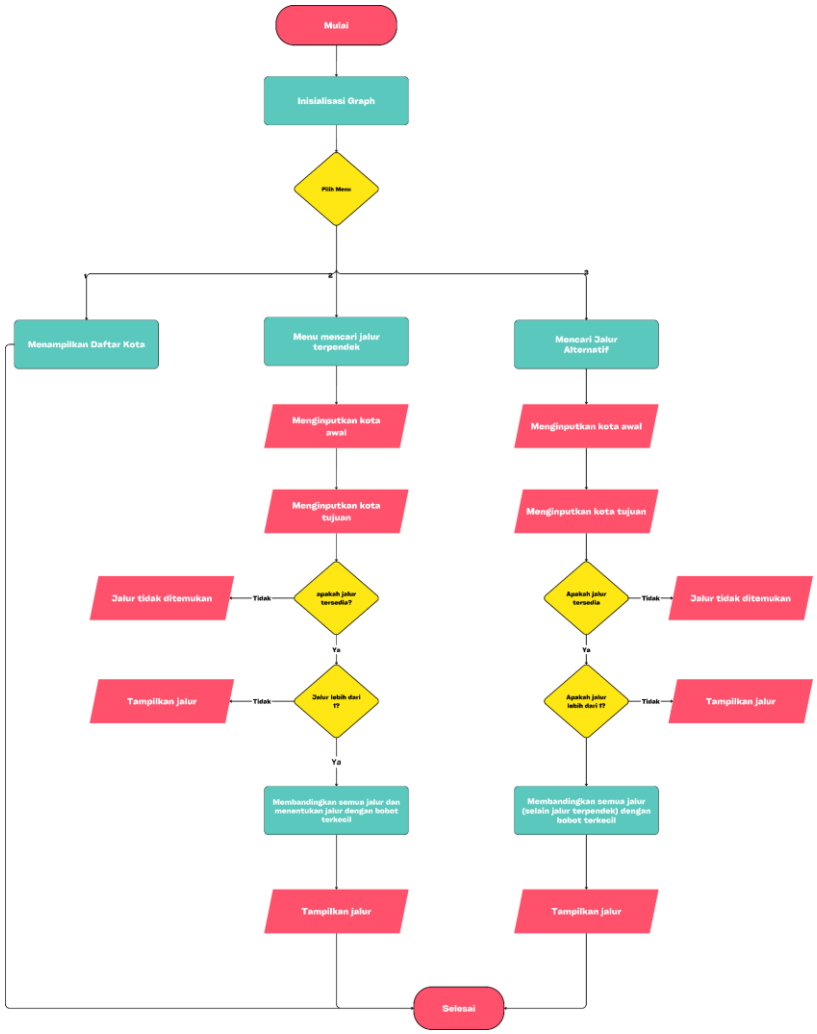
   Implementasi graph pada kasus pengelolaan jaringan transmisi listrik yang menghubungkan berbagai kota di Yogyakarta memungkinkan untuk menemukan jalur terpendek dari berbagai skenario dan mencari jalur terpendek atau tercepat antar kota. Representasi jaringan listrik dalam bentuk graph sangat efektif untuk memodelkan hubungan antar simpul (node), yang dalam konteks ini mewakili kota, dan sisi (edge) yang mewakili saluran transmisi listrik di antara mereka. Sebagai contoh, ketika terjadi gangguan di salah satu saluran transmisi, sistem dapat segera menghitung jalur alternatif terpendek untuk mengalihkan arus listrik ke tujuan yang sama tanpa mengganggu pasokan ke kota tersebut

   Selain itu, implementasi graph dalam pengelolaan jaringan transmisi listrik juga memungkinkan analisis optimasi distribusi daya untuk mengurangi kerugian energi selama transmisi. Dengan algoritma seperti Dijkstra atau Floyd-Warshall, sistem dapat menghitung jalur yang tidak hanya terpendek secara jarak, tetapi juga paling efisien berdasarkan kapasitas saluran dan beban jaringan. Analisis semacam ini sangat penting dalam perencanaan dan pengelolaan jaringan untuk memastikan bahwa pasokan listrik tetap stabil dan andal, terutama di wilayah dengan kebutuhan listrik yang terus meningkat seperti Yogyakarta. Model ini juga dapat digunakan untuk simulasi dan mitigasi risiko jika terjadi pemadaman mendadak, sehingga mendukung respons cepat dalam kondisi darurat.

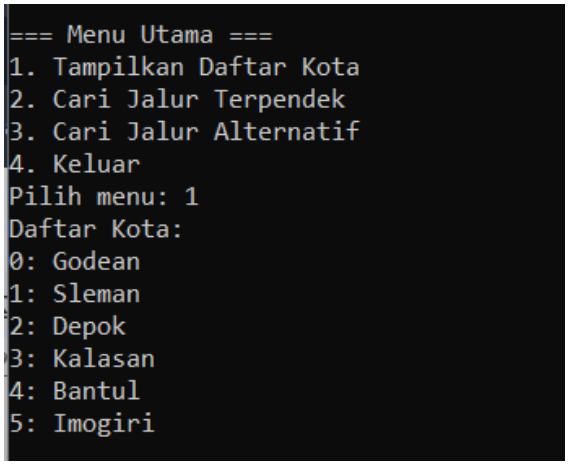
3. Ilustrasi Graph

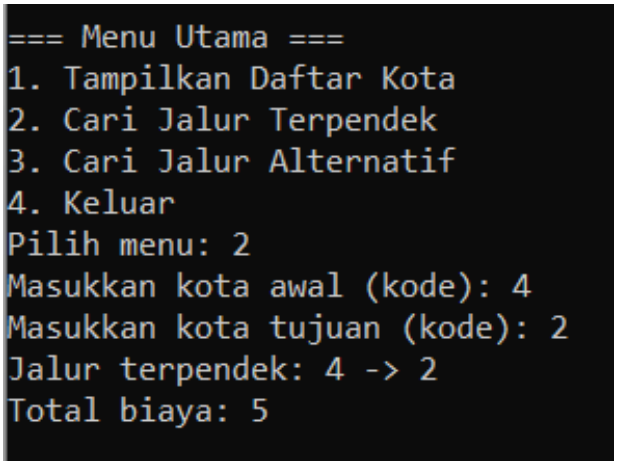


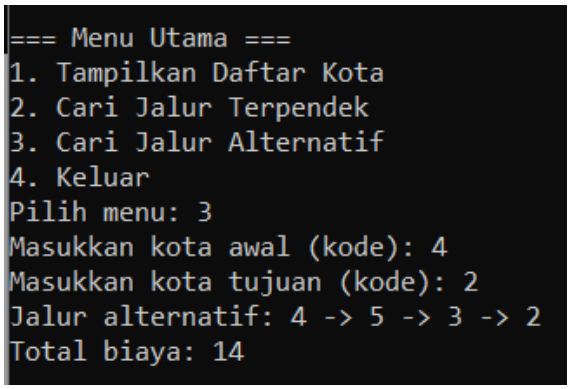
4. Ilustrasi Desain Algoritma



5. **Hasil Running Code**







Penjelasan Output

Output dari implementasi kode graph menampilkan beberapa menu, diantaranya: menampilkan daftar kota (node), mencari jalur terpendek antar kota (node), dan mencari jalur alternatif jika sebuah node memiliki lebih dari satu jalur. Apabila sebuah node memiliki lebih dari dua jalur, jalur alternatif yang akan ditampilkan atau dipilih adalah jalur yang bukan merupakan jalur terpendek tetapi memiliki bobot lebih kecil dibandingkan jalur lainnya.

Disclaimer

Gambar output hanya menampilkan sebagian output saja, tetapi kalau misal mau mencari dari kota manapun untuk menu 1 dan 2 sudah dapat mencari jalur terpendek maupun alternatif semuanya.

6. File Code

```
#include <iostream>
#include <vector>
#include <queue>
#include <climits>
#include <algorithm>
using namespace std;

struct Edge {
    int to, weight;
};

// Fungsi untuk menemukan jalur terpendek menggunakan Dijkstra
void findShortestPath(vector<vector<Edge>> &graph, int start, int
end, vector<int> &shortestPath, int &totalCost) {
    int n = graph.size();
    vector<int> dist(n, INT_MAX);
    vector<int> prev(n, -1);
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<>>
pq;

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

    while (!pq.empty()) {
        int cost = pq.top().first;
        int current = pq.top().second;
        pq.pop();

        if (cost > dist[current]) continue;

        for (Edge &e : graph[current]) {
            int next = e.to;
            int weight = e.weight;
            if (dist[current] + weight < dist[next]) {
                dist[next] = dist[current] + weight;
                prev[next] = current;
                pq.push({dist[next], next});
            }
        }
    }

    // Menyimpan jalur terpendek
    shortestPath.clear();
    totalCost = dist[end];
    if (dist[end] == INT_MAX) {
        cout << "Tidak ada jalur dari " << start << " ke " << end <<
endl;
        return;
    }
    for (int at = end; at != -1; at = prev[at]) {
        shortestPath.push_back(at);
    }
    reverse(shortestPath.begin(), shortestPath.end());
}

// Fungsi untuk menemukan jalur alternatif (jalur kedua terbaik)
void findAlternativePath(vector<vector<Edge>> &graph, int start, int
```

Telkom University Purwokerto

```
end, vector<int> &altPath, int &altCost, vector<int> &shortestPath,
int shortestCost) {
    int n = graph.size();
    vector<int> dist(n, INT_MAX);
    vector<int> prev(n, -1);
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<>>
pq;

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

    while (!pq.empty()) {
        int cost = pq.top().first;
        int current = pq.top().second;
        pq.pop();

        if (cost > dist[current]) continue;

        for (Edge &e : graph[current]) {
            int next = e.to;
            int weight = e.weight;

            // Abaikan jalur terpendek
            if (find(shortestPath.begin(), shortestPath.end(),
current) != shortestPath.end() &&
                find(shortestPath.begin(), shortestPath.end(), next)
!= shortestPath.end())
                continue;

            if (dist[current] + weight < dist[next]) {
                dist[next] = dist[current] + weight;
                prev[next] = current;
                pq.push({dist[next], next});
            }
        }
    }

    // Menyimpan jalur alternatif
    altPath.clear();
    altCost = dist[end];
    if (dist[end] == INT_MAX || altCost <= shortestCost) {
        altCost = INT_MAX;
        return;
    }
    for (int at = end; at != -1; at = prev[at]) {
        altPath.push_back(at);
    }
    reverse(altPath.begin(), altPath.end());
}

// Fungsi untuk menampilkan daftar kota
void displayCities() {
    cout << "Daftar Kota:\n";
    cout << "0: Godean\n";
    cout << "1: Sleman\n";
    cout << "2: Depok\n";
    cout << "3: Kalasan\n";
    cout << "4: Bantul\n";
```

```
    cout << "5: Imogiri\n";
}

int main() {
   // Graph berdasarkan gambar
    vector<vector<Edge>> graph(6);
    graph[0].push_back({1, 6});  // Godean -> Sleman
    graph[0].push_back({4, 3});  // Godean -> Bantul
    graph[1].push_back({2, 3});  // Sleman -> Depok
    graph[1].push_back({5, 9});  // Sleman -> Imogiri
    graph[2].push_back({0, 9});  // Depok -> Godean
    graph[2].push_back({5, 3});  // Depok -> Imogiri
    graph[3].push_back({2, 4});  // Kalasan -> Depok
    graph[3].push_back({4, 9});  // Kalasan -> Bantul
    graph[4].push_back({2, 5});  // Bantul -> Depok
    graph[4].push_back({5, 2});  // Bantul -> Imogiri
    graph[5].push_back({3, 8});  // Imogiri -> Kalasan

    vector<int> shortestPath, altPath;
    int shortestCost, altCost;
    int choice;

    do {
        cout << "\n=== Menu Utama ===\n";

        cout << "1. Tampilkan Daftar Kota\n";
        cout << "2. Cari Jalur Terpendek\n";
        cout << "3. Cari Jalur Alternatif\n";
        cout << "4. Keluar\n";
        cout << "Pilih menu: ";
        cin >> choice;

        switch (choice) {
            case 1:
                displayCities();
                break;
            case 2: {
                int start, end;
                cout << "Masukkan kota awal (kode): ";
                cin >> start;
                cout << "Masukkan kota tujuan (kode): ";
                cin >> end;
                findShortestPath(graph, start, end, shortestPath,
shortestCost);
                if (!shortestPath.empty()) {
                    cout << "Jalur terpendek: ";
                    for (int i = 0; i < shortestPath.size(); ++i) {
                        if (i > 0) cout << " -> ";
                        cout << shortestPath[i];
                    }
                    cout << "\nTotal biaya: " << shortestCost <<
endl;
                }
                break;
            }
            case 3: {
                int start, end;
                cout << "Masukkan kota awal (kode): ";
```

```
                cin >> start;
                cout << "Masukkan kota tujuan (kode): ";
                cin >> end;
                findShortestPath(graph, start, end, shortestPath,
shortestCost);
                findAlternativePath(graph, start, end, altPath,
altCost, shortestPath, shortestCost);
                if (!altPath.empty() && altCost != INT_MAX) {
                    cout << "Jalur alternatif: ";
                    for (int i = 0; i < altPath.size(); ++i) {
                        if (i > 0) cout << " -> ";
                        cout << altPath[i];
                    }
                    cout << "\nTotal biaya: " << altCost << endl;
            } else {
                    cout << "Tidak ada jalur alternatif yang
tersedia.\n";
                }
                break;
            }
            case 4:
                cout << "Keluar dari aplikasi. Terima kasih!\n";
                break;
            default:
                cout << "Pilihan tidak valid. Coba lagi.\n";
                break;
        }
    } while (choice != 4);

    return 0;
}
```

7. Prosentase Pembagian Tugas dan Link Video Presentasi

No	Nama	NIM	Jobdesk
1	Aji Prasetyo Nugroho	2211104049	• Membuat visualisasi graph • Membuat code program code • Membuat algoritma pemrograman • Membuat laporan
2	Dhiya Ulhaq Ramadhan	2211104053	• Membuat visualisasi graph • Membuat code program code • Membuat algoritma pemrograman • Membuat laporan
3	Muhammad Ralfi	2211104054	• Membuat visualisasi

			graph ● Membuat code program code ● Membuat algoritma pemrograman ● Membuat laporan

https://drive.google.com/file/d/1yWjeZdCiRO4eRvs39_4zF6ECGOlm7fMW/view?usp=sharing