LAPORAN PROGRES TUGAS BESAR

STRUKTUR DATA

Anggota Kelompok :

Tiurma Grace Angelina (2311104042)

Rengganis Tantri Pramudita (2311104065)

Dosen :

Wahyu Andy Saputra

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

1. **JUDUL**

   Analisis Graf untuk Jaringan Transportasi Penerbangan di Indonesia: Pesawat Lion Air rute pekanbaru - Semarang

2. **DESKRIPSI PROYEK**

   Proyek ini bertujuan untuk menganalisis jaringan transportasi penerbangan di Indonesia khususnya Pesawat Lion Air rute Pekanbaru - Semarang menggunakan struktur data graf. Proyek ini menggunakan *Adjacency List* untuk merepresentasikan hubungan langsung antar bandara dan *Adjacency Matri*x untuk memberikan gambaran matriks hubungan antar bandara. Bandara diwakili sebagai simpul (node), dan rute penerbangan sebagai sisi (edge) dengan bobot berupa waktu tempuh antar bandara. Implementasi tiap topik pada project

   a. Struktur Data:

   - Menggunakan struktur *map<string, vector<pair<string, int>>* untuk menyimpan daftar adjacency. Struktur ini efisien untuk menyimpan data rute berarah berbobot.
   - Vektor *vector<string>* digunakan untuk mencatat nama simpul (bandara).

   b. Fungsi Utama:

   - *cetakDaftarAdjacency*: Fungsi ini mencetak setiap simpul (bandara) beserta semua tetangganya (bandara yang langsung terhubung) dan waktu tempuh.
   - *cetakMatriksAdjacency:* Fungsi ini mencetak matriks adjacency. Matriks menunjukkan waktu tempuh antara bandara; jika tidak ada rute, maka nilai 0 diberikan.

   c. Input Data: Data input adalah daftar bandara utama dan rute penerbangan berikut waktu tempuhnya:

   - Bandara: Pekanbaru (PKU), Medan (KNO), Palembang (PLM), Jakarta (CGK), Semarang (SRG), Surabaya (SUB), Bali (DPS), Makassar (UPG).
   - Rute: Total 15 rute dengan waktu tempuh seperti yang diinputkan dalam program.

   d. Proses Output:

   - Output pertama adalah daftar adjacency, menunjukkan rute langsung dan waktu tempuhnya.
   - Output kedua adalah matriks adjacency dalam format tabel.

   e. Algoritma Cetak Matriks:- Untuk setiap simpul asal, periksa semua simpul tujuan.

- Jika ada rute langsung, masukkan bobot ke matriks.
- Jika tidak ada rute, masukkan nilai 0.

Proyek ini juga mencakup implementasi visualisasi jaringan transportasi menggunakan SDL2 (Simple DirectMedia Layer), sebuah pustaka multimedia lintas platform yang digunakan untuk menggambar grafis 2D secara interaktif. Visualisasi ini memberikan gambaran intuitif dari jaringan transportasi yang direpresentasikan dalam struktur data graf.

a. Representasi Graf di Visualisasi

- **Node (Simpul)**: Setiap bandara direpresentasikan sebagai lingkaran pada layar. Nama bandara ditampilkan di tengah lingkaran.
- **Edge (Sisi)**: Rute penerbangan antara dua bandara digambarkan sebagai garis penghubung antara dua lingkaran, dengan bobot (waktu tempuh) ditampilkan di tengah garis. Jika rute memiliki arah, sebuah panah kecil akan ditambahkan pada ujung garis.

b. Struktur Visualisasi

- **SDL2**: Digunakan untuk menggambar lingkaran (node), garis (edge), teks, dan panah arah.
- **SDL_ttf**: Dipakai untuk menampilkan teks seperti nama bandara dan waktu tempuh antar bandara.
- **Pengaturan Tampilan**:
    - Layar memiliki ukuran tertentu (misalnya 800x600 piksel) dengan latar belakang putih.
    - Posisi node diatur secara manual untuk memastikan graf terlihat rapi dan tidak saling tumpang tindih.

c. Fungsi Utama Visualisasi

- **AdjustLineToCircle**: Menyesuaikan garis agar tidak menabrak lingkaran, dengan memotong garis sejauh radius lingkaran.
- **DrawNode**: Menggambar lingkaran dan label untuk setiap bandara.

- **DrawEdge**: Menggambar garis penghubung antara dua node, menampilkan bobot, dan menambahkan panah jika rute berarah.
- **DrawArrow:** Membuat panah kecil di ujung garis sebagai penunjuk arah.
- **Loop Render**: Memastikan bahwa semua elemen digambar ulang setiap frame dan memproses event seperti input dari pengguna atau penutupan aplikasi.

d. Manfaat Visualisasi

- Membantu pengguna memahami hubungan antar bandara secara lebih intuitif.
- Memberikan cara untuk memvalidasi struktur data graf dengan membandingkan adjacency list/matrix dengan visualisasi yang dihasilkan.
- Menyediakan alat interaktif untuk memperlihatkan jaringan transportasi dengan elemen estetika yang sederhana.

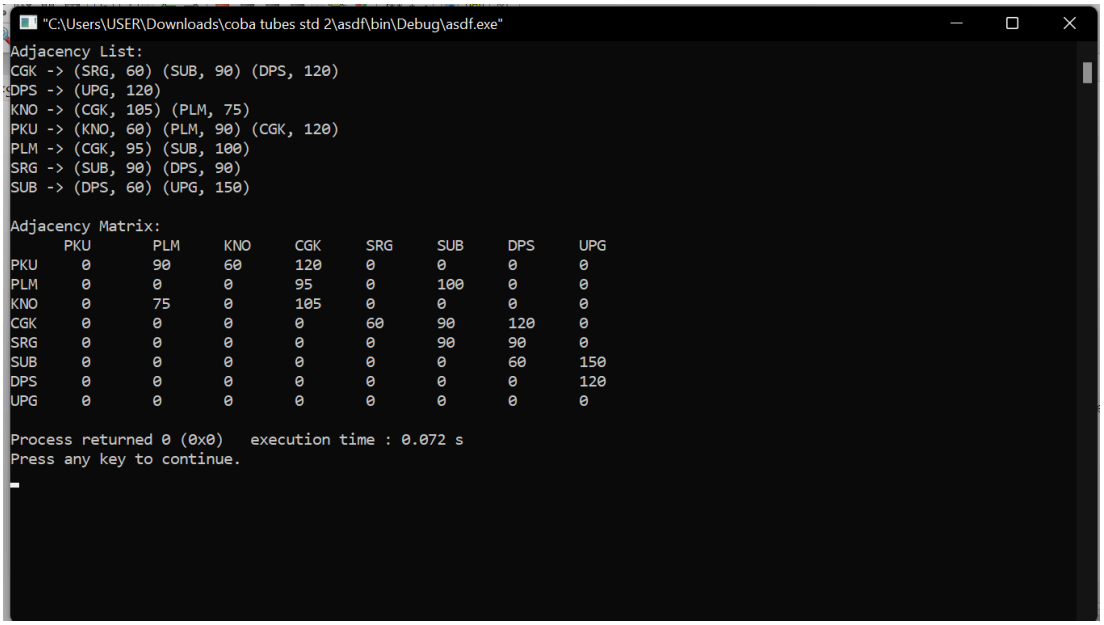
e. Contoh Hasil Visualisasi

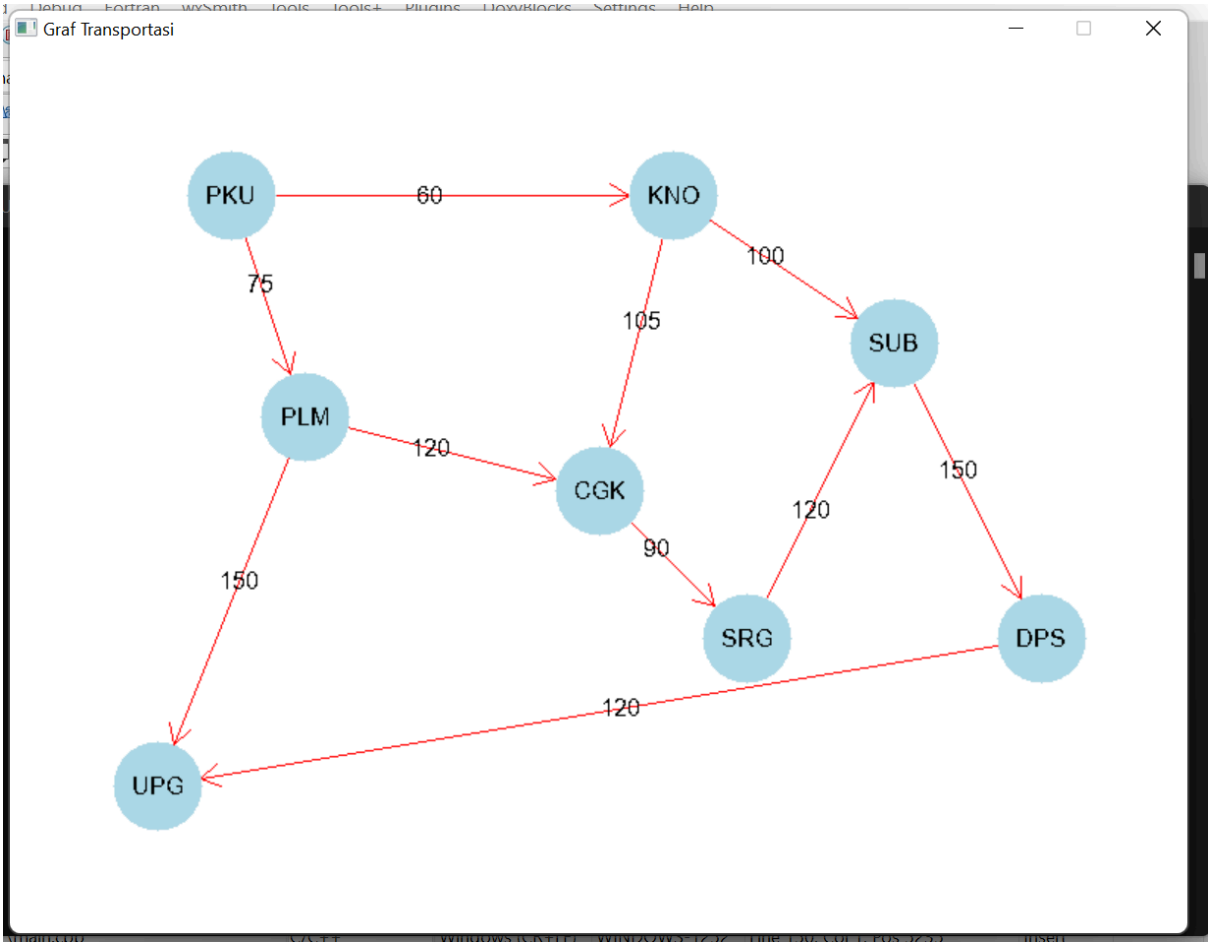
- Bandara PKU digambarkan sebagai lingkaran dengan label "PKU" di tengahnya.
- Rute dari PKU ke KNO digambarkan sebagai garis merah dengan label waktu tempuh "60" di tengah garis dan panah menuju KNO.
- Rute dua arah, seperti antara CGK dan SRG, memiliki dua panah pada garis yang sama untuk menunjukkan arah bolak-balik.

f. Kombinasi Data dan Visualisasi

- Data dari adjacency list digunakan untuk menggambar node dan edge, memastikan bahwa semua informasi jaringan tercermin dalam visualisasi.
- Visualisasi juga dapat menjadi dasar untuk interaktivitas, seperti memperbesar/memperkecil tampilan graf atau menambahkan rute baru secara langsung.

3. HASIL RUNNING





4. KODE PROGRAM

1. KODE PROGRAM Representasi Logis Graf Transportasi

```
#include <iostream>
```

```
#include <vector>
#include <map>
#include <string>
using namespace std;

void printAdjacencyList(map<string, vector<pair<string, int>>>& graph) {
    cout << "Adjacency List:" << endl;
    for (auto& node : graph) {
        cout << node.first << " -> ";
        for (auto& neighbor : node.second) {
            cout << "(" << neighbor.first << ", " << neighbor.second << ") ";
        }
        cout << endl;
    }
}

void printAdjacencyMatrix(map<string, vector<pair<string, int>>>& graph,
vector<string>& nodes) {
    cout << "\nAdjacency Matrix:" << endl;

    cout << "      ";
    for (const auto& node : nodes) {
        cout << node << "\t";
    }
    cout << endl;

    for (const auto& node : nodes) {
        cout << node << "\t";
        for (const auto& target : nodes) {
            bool found = false;
            for (const auto& neighbor : graph[node]) {
                if (neighbor.first == target) {
                    cout << neighbor.second << "\t";
                    found = true;
                    break;
                }
            }
            if (!found) {
                cout << "0\t";
            }
        }
        cout << endl;
    }
}

int main() {
    vector<string> nodes = {"PKU", "PLM", "KNO", "CGK", "SRG", "SUB",
"DPS", "UPG"};
```

```
    map<string, vector<pair<string, int>>> graph;

    graph["PKU"].push_back({"KNO", 60});
    graph["PKU"].push_back({"PLM", 90});
    graph["PKU"].push_back({"CGK", 120});
    graph["KNO"].push_back({"CGK", 105});
    graph["KNO"].push_back({"PLM", 75});
    graph["PLM"].push_back({"CGK", 95});
    graph["PLM"].push_back({"SUB", 100});
    graph["CGK"].push_back({"SRG", 60});
    graph["CGK"].push_back({"SUB", 90});
    graph["CGK"].push_back({"DPS", 120});
    graph["SRG"].push_back({"SUB", 90});
    graph["SRG"].push_back({"DPS", 90});
    graph["SUB"].push_back({"DPS", 60});
    graph["SUB"].push_back({"UPG", 150});
    graph["DPS"].push_back({"UPG", 120});


    printAdjacencyList(graph);

    printAdjacencyMatrix(graph, nodes);

    return 0;
}
```

2. KODE PROGRAM Visualisasi Graf Transportasi Menggunakan SDL2

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <iostream>
#include <vector>
#include <cmath>

const int SCREEN_WIDTH = 800;
const int SCREEN_HEIGHT = 600;
const int NODE_RADIUS = 30;
const SDL_Color NODE_COLOR = {173, 216, 230, 255}; // Light Blue
const SDL_Color EDGE_COLOR = {255, 0, 0, 255}; // Red
const SDL_Color TEXT_COLOR = {0, 0, 0, 255}; // Black

struct Node {
    int x, y;
    std::string label;
};

struct Edge {
    int from, to;
    int weight;
```

```
};

void AdjustLineToCircle(int &x1, int &y1, int &x2, int &y2, int radius) {
   double angle = atan2(y2 - y1, x2 - x1);
   x2 -= radius * cos(angle);
   y2 -= radius * sin(angle);
}

void DrawArrow(SDL_Renderer *renderer, int x1, int y1, int x2, int y2) {
   const double ARROW_ANGLE = M_PI / 6;
   const double ARROW_LENGTH = 15.0;

   double angle = atan2(y2 - y1, x2 - x1);

   int arrowX1 = x2 - ARROW_LENGTH * cos(angle - ARROW_ANGLE);
   int arrowY1 = y2 - ARROW_LENGTH * sin(angle - ARROW_ANGLE);
   int arrowX2 = x2 - ARROW_LENGTH * cos(angle + ARROW_ANGLE);
   int arrowY2 = y2 - ARROW_LENGTH * sin(angle + ARROW_ANGLE);

   SDL_RenderDrawLine(renderer, x2, y2, arrowX1, arrowY1);
   SDL_RenderDrawLine(renderer, x2, y2, arrowX2, arrowY2);
}

void DrawNode(SDL_Renderer *renderer, TTF_Font *font, Node node) {
   SDL_SetRenderDrawColor(renderer, NODE_COLOR.r, NODE_COLOR.g,
NODE_COLOR.b, NODE_COLOR.a);
   for (int w = -NODE_RADIUS; w <= NODE_RADIUS; w++) {
      for (int h = -NODE_RADIUS; h <= NODE_RADIUS; h++) {
         if (w * w + h * h <= NODE_RADIUS * NODE_RADIUS) {
            SDL_RenderDrawPoint(renderer, node.x + w, node.y + h);
         }
      }
   }

   SDL_Surface *textSurface = TTF_RenderText_Solid(font, node.label.c_str(),
TEXT_COLOR);
   SDL_Texture *textTexture = SDL_CreateTextureFromSurface(renderer,
textSurface);
   SDL_Rect textRect = {node.x - textSurface->w / 2, node.y - textSurface->h / 2,
textSurface->w, textSurface->h};
   SDL_RenderCopy(renderer, textTexture, nullptr, &textRect);
   SDL_FreeSurface(textSurface);
   SDL_DestroyTexture(textTexture);
}

void DrawEdge(SDL_Renderer *renderer, TTF_Font *font, Node from, Node to,
int weight) {
   SDL_SetRenderDrawColor(renderer, EDGE_COLOR.r, EDGE_COLOR.g,
EDGE_COLOR.b, EDGE_COLOR.a);
```

```
    int x1 = from.x, y1 = from.y, x2 = to.x, y2 = to.y;
    AdjustLineToCircle(x1, y1, x2, y2, NODE_RADIUS);

    SDL_RenderDrawLine(renderer, x1, y1, x2, y2);

    DrawArrow(renderer, x1, y1, x2, y2);

    int midX = (x1 + x2) / 2;
    int midY = (y1 + y2) / 2;

    std::string weightStr = std::to_string(weight);
    SDL_Surface *textSurface = TTF_RenderText_Solid(font, weightStr.c_str(),
TEXT_COLOR);
    SDL_Texture *textTexture = SDL_CreateTextureFromSurface(renderer,
textSurface);
    SDL_Rect textRect = {midX - textSurface->w / 2, midY - textSurface->h / 2,
textSurface->w, textSurface->h};
    SDL_RenderCopy(renderer, textTexture, nullptr, &textRect);
    SDL_FreeSurface(textSurface);
    SDL_DestroyTexture(textTexture);
}

int main(int argc, char *argv[]) {
    if (SDL_Init(SDL_INIT_VIDEO) < 0) {
        std::cerr << "Failed to initialize SDL: " << SDL_GetError() << std::endl;
        return -1;
    }

    if (TTF_Init() < 0) {
        std::cerr << "Failed to initialize SDL_ttf: " << TTF_GetError() << std::endl;
        SDL_Quit();
        return -1;
    }

    TTF_Font *font = TTF_OpenFont("arial.ttf", 16);
    if (!font) {
        std::cerr << "Failed to load font: " << TTF_GetError() << std::endl;
        TTF_Quit();
        SDL_Quit();
        return -1;
    }

    SDL_Window *window = SDL_CreateWindow("Graf Transportasi",
SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED,
SCREEN_WIDTH, SCREEN_HEIGHT, SDL_WINDOW_SHOWN);
    if (!window) {
        std::cerr << "Failed to create window: " << SDL_GetError() << std::endl;
        TTF_CloseFont(font);
```

```
        TTF_Quit();
        SDL_Quit();
        return -1;
    }

    SDL_Renderer *renderer = SDL_CreateRenderer(window, -1,
SDL_RENDERER_ACCELERATED);
    if (!renderer) {
        std::cerr << "Failed to create renderer: " << SDL_GetError() << std::endl;
        SDL_DestroyWindow(window);
        TTF_CloseFont(font);
        TTF_Quit();
        SDL_Quit();
        return -1;
    }

    std::vector<Node> nodes = {
        {150, 100, "PKU"}, {200, 250, "PLM"}, {450, 100, "KNO"},
        {400, 300, "CGK"}, {500, 400, "SRG"}, {600, 200, "SUB"}, {700, 400,
"DPS"}, {100, 500, "UPG"}
    };

    std::vector<Edge> edges = {
        {0, 1, 75}, {0, 2, 60}, {1, 3, 120}, {2, 3, 105}, {3, 4, 90},
        {4, 5, 120}, {5, 6, 150}, {6, 7, 120}, {1, 7, 150}, {2, 5, 100}
    };

    bool isRunning = true;
    SDL_Event event;

    while (isRunning) {
        while (SDL_PollEvent(&event)) {
            if (event.type == SDL_QUIT) {
                isRunning = false;
            }
        }

        SDL_SetRenderDrawColor(renderer, 255, 255, 255, 255);
        SDL_RenderClear(renderer);

        for (auto &edge : edges) {
            DrawEdge(renderer, font, nodes[edge.from], nodes[edge.to], edge.weight);
        }

        for (auto &node : nodes) {
            DrawNode(renderer, font, node);
        }

        SDL_RenderPresent(renderer);
```

```
    }

    SDL_DestroyRenderer(renderer);
    SDL_DestroyWindow(window);
    TTF_CloseFont(font);
    TTF_Quit();
    SDL_Quit();

    return 0;
}
```

5. **Link Video**

   https://drive.google.com/drive/folders/1UUT9XbVODRGzjaOp8Km51MN46u_AloIK

6. **Pembagian Tugas dalam Kelompok**

Nama	**Tugas**
Tiurma Grace Angelina (2311104042)	**a. Program** - Membuat struktur data graf menggunakan map<string, vector<pair<string, int>>>. - Mencetak daftar adjacency ke layar, termasuk iterasi pada simpul dan tetangga. - Menyusun logika matriks adjacency (Adjacency Matrix) **b. Laporan** - Mengerjakan Implementasi Proyek - Menambahkan penjelasan terkait pengujian keseluruhan program.
Rengganis Tantri Pramudita (2311104065)	**a. Program** - Menambahkan rute penerbangan (edges) beserta bobot (waktu tempuh) ke dalam graf. - Memastikan semua rute dan waktu tempuh tercetak dengan benar. - Mengintegrasikan kedua fungsi (cetak daftar adjacency dan cetak matriks adjacency). **b. Laporan** - Mengerjakan Implementasi Proyek dan rincian pembagian tugas

	- Menyusun hasil berupa screenshot program.