LAPORAN PRAKTIKUM STRUKTUR DATA
MODUL XI

Laporan ini disusun untuk memenuhi Tugas Mata Kuliah

Praktikum struktur data

Disusun Oleh :

M HAMKA ZAINUL ARDHI

NIM: 2311103156

PROGRAM STUDI S1 SISTEM INFORMASI

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

DASAR TEORI

1. Graph

Graph adalah struktur data non-linier. Graph dapat didefinisikan sebagai kumpulan Node yang disebut juga “simpul” dan “tepi” yang menghubungkan dua simpul atau lebih graphik dalam C++ adalah struktur data non-linier yang didefinisikan sebagai kumpulan simpul dan sisi. Ada beberapa tipe graph, termasuk graph berarah dan tidak berarah, serta graph berbobot dan tidak berbobot. Representasi graph dapat dilakukan melalui matriks ketetanggan (adjacency matrix) atau daftar ketetanggan (adjacency list).

Beberapa operasi dasar pada graph meliputi penambahan dan penghapusan simpul serta sisi. Terdapat juga berbagai algoritma yang digunakan dalam pengolahan graph, seperti BFS, DFS, Dijkstra, dan Bellman-Ford. Graph memiliki banyak aplikasi dalam berbagai bidang, termasuk jejaring sosial, sistem navigasi, analisis jaringan, dan perencanaan proyek. Dengan pemahaman tentang konsep dasar ini, kita dapat mengaplikasikan graph dalam pemecahan masalah di dunia nyata menggunakan bahasa pemrograman C++. Jika di gambar kan secara matematis graph akan seperti rumus berikut.

$$G = (V,E)$$

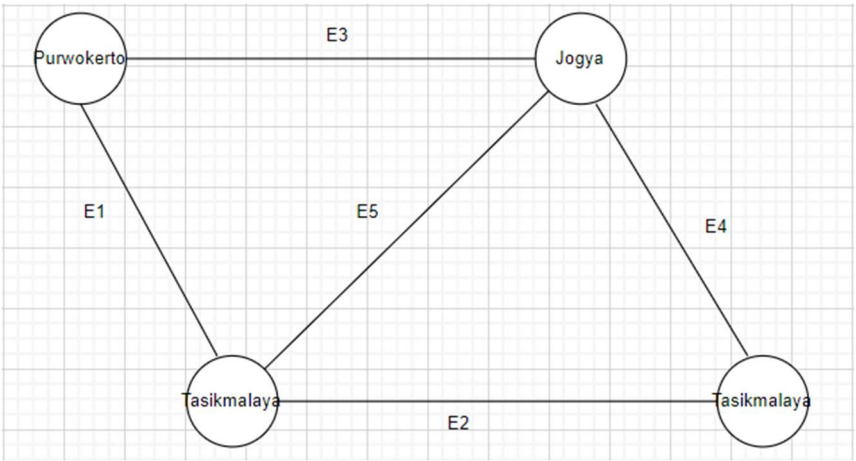
Yang artinnya bahwa

‘G’ sebagai Graph

‘V’ sebagai simpul atau busur atau node atau titik

‘E’ sebagai edge atau busur atau ruas

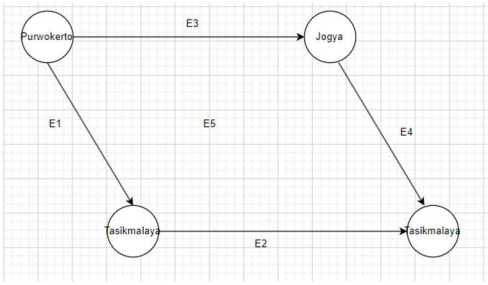
Dan jika di implmentasikan pada sebuah gambar (dibuat dengan platform drawio)



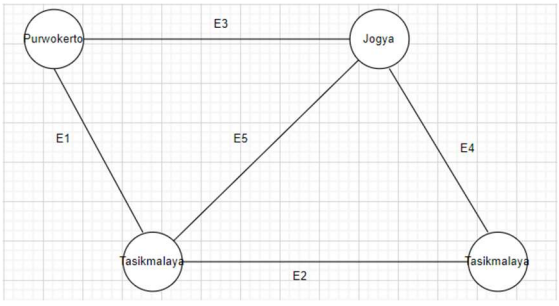
2. Jenis graph

Diatas telah di singgung bahwa garph mempunyai graph berarah dan tidak berarah, serta graph berbobot dan tidak berbobot.

a. Graph berarah adalah suatu golongan graph yang tidak mengasumsikan simetri atau timbal balik pada sisi-sisi yang terbentuk di antara simpul-simpulnya. Pada graph berarah, jika dan adalah dua simpul yang dihubungkan oleh sebuah sisi , hal ini tidak berarti bahwa terdapat juga sisi yang menghubungkan Tepi berarah biasanya direpresentasikan sebagai anak panah yang menunjuk menjauhi titik asal, atau ekor anak panah, dan menuju ke titik tujuan, atau kepala anak panah. Contoh grap berarah yang mempunya tanda panah



b. Graph tidak berarah lebih spesifik. Jika terdapat sebuah sisi (a,b)di antara dua simpul B, maka sisi tersebut (b,a)juga ada Graph tak berarah, dalam arti tertentu, lebih membatasi daripada graphik berarah, karena graph tidak mengizinkan pemodelan hubungan yang bersifat hierarkis. Namun hal ini sangat umum dalam praktiknya, dan banyak hubungan di dunia nyata paling baik dimodelkan dengan graphik tidak berarah. Hal ini biasanya terjadi jika kedua simpul dari suatu sisi dapat menjadi subjek dari hubungan tersebut seperti contoh gambar ini



3. Implemntasi Graph pada C++

```
struct Graph {
    int V; // Ini adalah inisiasi dari Jumlah simpul
    vector<vector<int>> adjList; // Daftar vertex graph

    // untuk menginisialisasi jumlah simpul
    Graph(int vertices) {
        V = vertices;
        adjList.resize(V);
    }
};

// Fungsi untuk menambahkan sisi dari u ke v
void addEdge(Graph& graph, int u, int v) {
    graph.adjList[u].push_back(v);
}
```

Menggunakan struktur (struct) Graph untuk merepresentasikan graph. Fungsi addEdge digunakan untuk menambahkan sisi dari simpul u ke simpul v membuat objek graph dengan 4 simpul dan menambahkan beberapa sisi ke dalamnya

GUIDED

1. membuat program matriks dalam C++ Berdasarkan ilustrasi graph berarah dan berbobot yang ditunjukkan pada tabel ini

Dari/Ke	Bandung	Bekasi	Jakarta	Purwokerto	Semarang	Tasikmalaya	Yogyakarta
Bandung	0	5	0	15	0	0	0
Bekasi	6	0	0	0	5	0	0
Jakarta	7	8	0	0	0	0	0
Purwokerto	0	0	0	0	7	0	3
Semarang	0	0	23	0	0	10	8
Tasikmalaya	5	0	0	4	0	0	0
Yogyakarta	0	0	0	4	9	0	0

source code

```
#include <iostream>
using namespace std;

int main() {

  int matrix[7][7] = {{99, 5, 99, 99, 99, 15, 99},
                      {6, 99, 99, 5, 99, 99, 99},
                      {7, 8, 99, 5, 99, 99, 99},
                      {99, 99, 23, 99, 99, 99, 8},
                      {5, 99, 99, 99, 99, 4, 99},
                      {99, 99, 99, 7, 99, 99, 3},
                      {99, 99, 99, 9, 99, 4, 99}};


  for (int i = 0; i < 7; i++) {
    for (int j = 0; j < 7; j++) {
      cout << matrix[i][j] << ' ';
    }
    cout << endl;
  }

  return 0;
}
```

Analisis

Untuk merepsentasi graphnya dengan matriks. Dibutuhkan array dua dimensi yang masing masing mempunyai kolom berjumlah tujuh dan baris berjumlah tujuh ini memungkin kan memasukan nilai nilai yang terdapat pada tabel secara manual, setelah nilai nilai matriks di deklrasikan terdapat perulangan for disini ada dua loop bersarang.

1. Loop luar (for (int i = 0; i < 7; i++)) mengiterasi melalui baris-baris dari matriks, sedangkan
2. loop dalam (for (int j = 0; j < 7; j++)) mengiterasi melalui kolom-kolom dari setiap baris.

Lalu elemen akan di tampilakan dari matriks dicetak diikuti oleh spasi. Dengan demikian, seluruh elemen dari matriks 7x7 dicetak dalam format yang terstruktur, di mana setiap baris matriks muncul pada baris baru di output.

Hasil Program

```
PS C:\Users\hamka\OneDrive\Documents\PRAKTIKUM STRUKTUR DATA\MODUL 11>
11\" ; if ($?) { g++ GUIDED.CPP -o GUIDED } ; if ($?) { .\GUIDED }
99 5 99 99 99 15 99
6 99 99 5 99 99 99
7 8 99 5 99 99 99
99 99 23 99 99 99 8
5 99 99 99 99 4 99
99 99 99 7 99 99 3
99 99 99 9 99 4 99
PS C:\Users\hamka\OneDrive\Documents\PRAKTIKUM STRUKTUR DATA\MODUL 11>
```

GUIDED 2

Memodifikasi guided II, buatlah program untuk menghasilkan matrik representasi dari graph pada tebel di atas dengan cara manual input nilai nya

- Manual Input

Source Code

```
#include <iostream>
#include <string>
using namespace std;

string simpul[7] = {‘Bandung’, ‘Bekasi’, ‘Jakarta’, ‘Purwokerto’, ‘Semarang’,
‘Tasikmalaya’, ‘Yogyakarta’};

int busur[7][7] = {
    {0, 5, 0, 15, 0, 0, 0},
    {6, 0, 0, 0, 5, 0, 0},
    {7, 8, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 7, 0, 3},
    {0, 0, 23, 0, 0, 10, 8},
    {5, 0, 0, 4, 0, 0, 0},
    {0, 0, 0, 4, 9, 0, 0}
};

void tampilGraph() {
    if (simpul && busur) {
        for (int baris = 0; baris < 7; baris++) {
            cout << simpul[baris] << ‘ : ‘;
            for (int kolom = 0; kolom < 7; kolom++) {
                if (busur[baris][kolom] != 0) {
                    cout << simpul[kolom] << ‘(‘ << busur[baris][kolom] << ‘)
‘;
                }
            }
            cout << endl;
        }
    }
}

int main() {
    tampilGraph();
    return 0;
}
```

Hasil Program

```
PS C:\Users\hamka\OneDrive\Documents\PRAKTIKUM STRUKTUR DATA\MODUL 11>
11\" ; if ($?) { g++ guided4.cpp -o guided4 } ; if ($?) { .\guided4 }
Bandung : Bekasi(5) Purwokerto(15)
Bekasi : Bandung(6) Semarang(5)
Jakarta : Bandung(7) Bekasi(8)
Purwokerto : Semarang(7) Yogyakarta(3)
Semarang : Jakarta(23) Tasikmalaya(10) Yogyakarta(8)
Tasikmalaya : Bandung(5) Purwokerto(4)
Yogyakarta : Purwokerto(4) Semarang(9)
PS C:\Users\hamka\OneDrive\Documents\PRAKTIKUM STRUKTUR DATA\MODUL 11>
```

- Input dari user

```
#include <iostream>
#include <string>

using namespace std;

int jumlahSimpul = 5;
string *dataSimpul;
int **dataBusur;
bool cekMatrik = false;

void buatMatriks(){
    dataSimpul = new string[jumlahSimpul];
    dataBusur = new int*[jumlahSimpul];
    dataBusur[0] = new int[jumlahSimpul * jumlahSimpul];
    for(int i = 1; i < jumlahSimpul; i++){
        dataBusur[i] = dataBusur[i-1] + jumlahSimpul;
    }

    cout<<'Silahkan masukkan nama simpul '<<endl;
    for(int i = 0; i < jumlahSimpul; i++){
        cout<<'Simpul '<<i+1<<' : ';
        cin>>dataSimpul[i];
    }


    cout<<'Silahkan masukkan bobot antar simpul '<<endl;
    for(int baris = 0; baris < jumlahSimpul; baris++){
        for(int kolom = 0; kolom < jumlahSimpul; kolom++){
            cout<<dataSimpul[baris]<<' --> '<<dataSimpul[kolom]<<' : ';
            cin>> dataBusur[baris][kolom];
        }
```

```
    }
    cekMatrik = true;
}

void tampilMatriks(){
    if(cekMatrik){
        for(int i = 0; i < jumlahSimpul; i++){
            cout<<dataSimpul[i]<<' ';
        }
        cout<<endl;
        for(int baris = 0; baris < jumlahSimpul; baris++){
            for(int kolom = 0; kolom < jumlahSimpul; kolom++){
                cout<<dataBusur[baris][kolom]<<' ';
            }
            cout<<endl;
        }
    }else{
        cout<<'Tidak ada matriks'<<endl;
    }
}

int main(){
    cout<<'Silahkan masukkan jumlah simpul : ';
    cin>>jumlahSimpul;
    buatMatriks();
    tampilMatriks();
    return 0;
}
```

Hasil program

```
tasikmalaya --> bandung : 5
tasikmalaya --> bekasi : 99
tasikmalaya --> jakarta : 99
tasikmalaya --> semarang : 99
tasikmalaya --> tasikmalaya : 99
tasikmalaya --> purwokerto : 4
tasikmalaya --> yogyakarta : 99
purwokerto --> bandung : 99
purwokerto --> bekasi : 99
purwokerto --> jakarta : 99
purwokerto --> semarang : 7
purwokerto --> tasikmalaya : 99
purwokerto --> purwokerto : 99
purwokerto --> yogyakarta : 3
yogyakarta --> bandung : 99
yogyakarta --> bekasi : 99
yogyakarta --> jakarta : 99
yogyakarta --> semarang : 9
yogyakarta --> tasikmalaya : 99
yogyakarta --> purwokerto : 4
yogyakarta --> yogyakarta : 99
bandung bekasi jakarta semarang tasikmalaya purwokerto yogyakarta
99 5 99 99 99 15 99
6 99 99 5 99 99 99
7 8 99 99 99 99 99
99 99 23 99 10 99 8
5 99 99 99 99 4 99
99 99 99 7 99 99 3
99 99 99 9 99 4 99
PS C:\Users\hamka\OneDrive\Documents\PRAKTIKUM STRUKTUR DATA\MODUL 11>
```

```
Silahkan masukkan jumlah simpul : 7
Silahkan masukkan nama simpul
Simpul 1 : bandung
Simpul 2 : bekasi
Simpul 3 : jakarta
Simpul 4 : semarang
Simpul 5 : tasikmalaya
Simpul 6 : purwokerto
Simpul 7 : yogyakarta
Silahkan masukkan bobot antar simpul
bandung --> bandung : 99
bandung --> bekasi : 5
bandung --> jakarta : 99
bandung --> semarang : 99
bandung --> tasikmalaya : 99
bandung --> purwokerto : 15
bandung --> yogyakarta : 99
bekasi --> bandung : 6
bekasi --> bekasi : 99
bekasi --> jakarta : 99
bekasi --> semarang : 5
bekasi --> tasikmalaya : 99
bekasi --> purwokerto : 99
bekasi --> yogyakarta : 99
jakarta --> bandung : 7
jakarta --> bekasi : 8
jakarta --> jakarta : 99
jakarta --> semarang : 99
jakarta --> tasikmalaya : 99
jakarta --> purwokerto : 99
jakarta --> yogyakarta : 99
semarang --> bandung : 99
semarang --> bekasi : 99
semarang --> jakarta : 23
semarang --> semarang : 99
semarang --> tasikmalaya : 10
semarang --> purwokerto : 99
semarang --> yogyakarta : 8
```

Analisis source code 1 dan 2

Proses dimulai dengan mengalokasikan memori untuk menyimpan nama-nama simpul dan matriks adjacency.

- ‘dataSimpul’ adalah array dinamis yang dialokasikan menggunakan ‘new string[jumlahSimpul]’, yang akan menyimpan nama-nama dari setiap simpul yang dimasukkan pengguna.
- ‘dataBusur’ adalah array pointer ke array dua dimensi yang juga dialokasikan secara dinamis. ‘dataBusur’ dialokasikan menggunakan ‘new int*[jumlahSimpul]’ untuk baris
- ‘new int[jumlahSimpul * jumlahSimpul]’ untuk seluruh elemen matriks, sehingga setiap baris menunjuk ke bagian yang sesuai dalam blok memori yang dialokasikan untuk matriks.

Fungsi ‘tampilMatriks’ bertanggung jawab untuk menampilkan matriks yang dihasilkan. Fungsi ini pertama-tama memeriksa apakah matriks telah dibuat dengan memeriksa nilai ‘cekMatrik’. Jika matriks telah dibuat, fungsi ini mencetak nama-nama simpul terlebih dahulu sebagai header untuk kolom. Kemudian, untuk setiap baris dalam matriks, fungsi mencetak bobot busur untuk setiap kolom, sehingga menampilkan seluruh matriks dengan cara yang terstrukturjika matriks belum dibuat, fungsi akan menampilkan pesan ‘Tidak ada matriks’ untuk memberi tahu pengguna bahwa matriks belum tersedia. Pada program program di atas terdapat dua jenis kondisi input objek vertex yang berbeda yaitu

1. Input manual

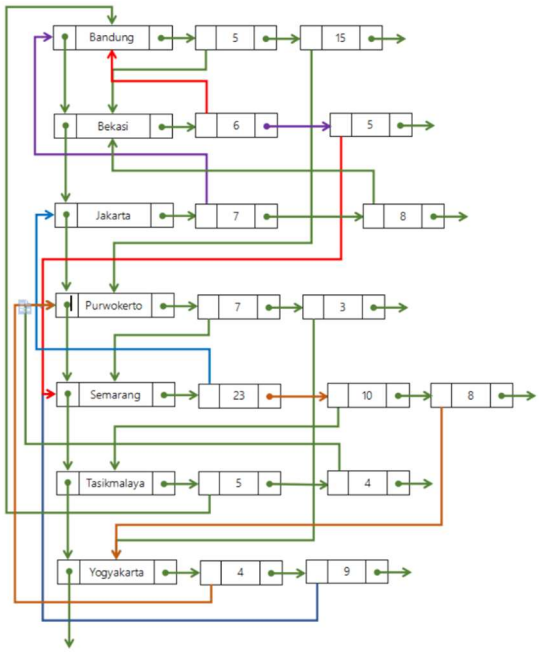
   Disini objek vertex garph langsung dimasukan kedalam kode tanpa ada tambahan di output ketika program di jalanan kan Artinya, nilai-nilai simpul (vertex) dan busur (edge) sudah ditentukan dan tidak dapat diubah oleh pengguna saat program dijalankan
2. Input oleh user

   Disini program dimodifikasi untuk memungkikan user memasukan data simpul dan busur ketika program di run ini bisa meningkatkan fleksibilitas dan tentu nya program lebih interaktif kepada user di banding dengan program input manual

GUIDED 3

membuat repsentasi graph menggunkan list Berdasarkan ilustrasi graph berarah dan berbobot yang ditunjukkan di bawah ini

graph dengan daftar (list) adalah salah satu cara untuk menyimpan graph dalam memori komputer representasi graph disni saya akan menggunakan daftar ialah List Adjacency (Daftar Ketetanggaan) berikut adlah reprensentasi garph di atas menggunkan list



Untuk merepresentasikan graph berarah dan berbobot dari gambar t dengan list, disisni bisa menggunakan adjacency list. Dalam adjacency list, setiap node menyimpan daftar tetangganya bersama dengan bobot tepinya

1. Simpul ‘Bandung’ terhubung ke simpul :
   - Bekasi [5]
   - Semarang [15]
2. Simpul Bekasi terhubung ke simpul :
   - Bandung [6]
   - Semarang [5]
3. Simpul ‘Jakarta’ terhubung ke simpul
   - bandung [7]
   - bekasi [8]
   -

4. Simpul ‘Purwokerto, terhubung ke simpul :
    - Semarang [7]
    - Yogyakarta [3]
5. Simpul ‘Semarang’ terhubung ke simpul :
    - Jakarta [23]
    - Tasikmalaya [10]
    - Yogyakarta [8]
6. Simpul ‘Tasikmalaya’ terhubung ke simpul
    - bandung [5]
    - purwokerto [4]
7. simpul ‘Yogyakarta’ terhubung ke simpul :
    - Purwokerto [4]
    - Semarang [9]

Disini setiap baris di atas menunjukkan node/kota diikuti oleh daftar tetangganya bersama dengan bobot dari setiap objekedge yang menghubungkan node tersebut dengan tetangganya. Misalnya, ‘Bandung: - Bekasi, 5 - Semarang, 15’ berarti ada objekedge dari Bandung ke Bekasi dengan bobot 5 dan edge dari Bandung ke Semarang dengan bobot 15.

GUIDED 4

Implementasikan lah guided 3 dalam program, buatlah program dari untuk menghasilkan list representasi dari graph yang telah di buat di list tersebut**.**

Source Code

```
#include <iostream>
#include <string>

using namespace std;

int jumlahSimpul = 5;
string *dataSimpul;
int **dataBusur;
bool cekMatrik = false;

struct graph{
    graph *kanan;
    string data;
    graph *kiri;
};
```

```
graph *simpul;
graph *busur;
graph *awal;
graph *akhir;
graph **alamat;
graph *helperA;
graph *helperB;

void inisiasi(){
    awal = NULL;
    akhir = NULL;
}

bool graphKosong(){
    if(awal == NULL && akhir == NULL){
        return true;
    }else{
        return false;
    }
}

void buatMatriks(){
    dataSimpul = new string[jumlahSimpul];
    dataBusur = new int*[jumlahSimpul];
    dataBusur[0] = new int[jumlahSimpul * jumlahSimpul];

    for(int i = 1; i < jumlahSimpul; i++){
        dataBusur[i] = dataBusur[i-1] + jumlahSimpul;
    }

    cout<<'Silahkan masukkan nama simpul '<<endl;
    for(int i = 0; i < jumlahSimpul; i++){
        cout<<'Kota '<<i+1<<' : ';
        cin>>dataSimpul[i];
    }

    cout<<'Silahkan masukkan bobot antar simpul '<<endl;
    for(int baris = 0; baris < jumlahSimpul; baris++){
        for(int kolom = 0; kolom < jumlahSimpul; kolom++){
            cout<<dataSimpul[baris]<<' --> '<<dataSimpul[kolom]<<' : ';
            cin>> dataBusur[baris][kolom];
        }
    }
    cekMatrik = true;

}

void buatSimpulGraph(){
```

```
alamat = new graph*[jumlahSimpul];
buatMatriks();
    for(int i = 0; i < jumlahSimpul; i++){
        if(graphKosong()){
            simpul = new graph;
            simpul->data = dataSimpul[i];
            simpul->kanan = NULL;
            simpul->kiri = NULL;
            awal = simpul;
            akhir = simpul;
            alamat[i] = awal;
        }else{
            simpul = new graph;
            simpul->data = dataSimpul[i];
            akhir->kiri = simpul;
            akhir = simpul;
            simpul->kiri = NULL;
            simpul->kanan = NULL;
            alamat[i] = akhir;
        }
    }

    helperA = awal;
    for(int baris = 0; baris < jumlahSimpul; baris++){
        helperB = helperA;
        for(int kolom = 0; kolom < jumlahSimpul; kolom++){
            if(dataBusur[baris][kolom] != 0){
                simpul = new graph;
                simpul->data = to_string(dataBusur[baris][kolom]);
                helperB->kanan = simpul;
                simpul->kiri = alamat[kolom];
                simpul->kanan = NULL;
                helperB = simpul;
            }
        }
        helperA = helperA->kiri;
    }
}

void tampilGraph(){
    if(!graphKosong()){
        helperA = awal;
        while(helperA != NULL){
            cout<<helperA->data<<' : ';
            helperB = helperA->kanan;
            while(helperB != NULL){
                cout<<helperB->kiri->data<<' : '<<helperB->data<<' ';
                helperB = helperB->kanan;
```

```
            }
            cout<<endl;
            helperA = helperA->kiri;
        }
    }else{
        cout<<'Graph kosong...!!!'<<endl;
    }
}

int main(){
    inisiasi();
    cout<<'Silahkan masukkan jumlah kota : ';
    cin>>jumlahSimpul;
    buatSimpulGraph();
    tampilGraph();
    return 0;
}
```

Penerapan guided 3 dalam program garph

membuat struktur data graph yang direpresentasikan dengan daftar terdekat (adjacency lists), di mana setiap simpul dalam graph berisi string yang mewakili nama sebuah kota yang ada di guided 3. Lalu Pengguna diminta untuk memasukkan jumlah kota, nama-nama kota, dan bobot dari tepian (edge) antara kota-kota tersebut. Program kemudian membuat graph berdasarkan input ini dan menampilkannya disini terdiri dari sejumlah fungsi yang berperan penting dalam pembuatan dan penampilan graph.

- fungsi 'inisiasi()' digunakan untuk mengatur kondisi awal graph dgn menginisialisasi pointer 'awal' dan 'akhir' menjadi 'NULL', menandakan bahwa graph awalnya kosong.
- fungsi 'graphKosong()' memeriksa apakh graph kosong dengan memeriksa apakah kedua pointer 'awal' dan 'akhir' berada pda 'NULL'.
- Fungsi 'buatMatriks()' bertugas untuk membuat matriks kejadian terdekat berdasarkan input pengguna, yg kemudian digunakan untuk membuat graph. Di dalamnya, program meminta pengguna untuk memasukkan nama kota dan bobot tepian antar kota, lalu menyimpannya dalm struktur data yang sesuai.
- fungsi 'buatSimpulGraph()' membuat graph berdasarkan matriks kejadian terdekat yg telah dibuat sebelumnya. Proses ini melibatkan iterasi melalui matriks untuk membuat simpul-simpul dan tepian antara simpul-simpul tsb.
- fungsi 'tampilGraph()' digunakan untuk menampilkan graph yag telah dibuat kepada pengguna. disini program mengiterasi melalui setiap simpul dan tepian

dalam graph lalu menampilkan nya secara terstruktur sesuai format yang telah ditentukan code.

Hasil Program

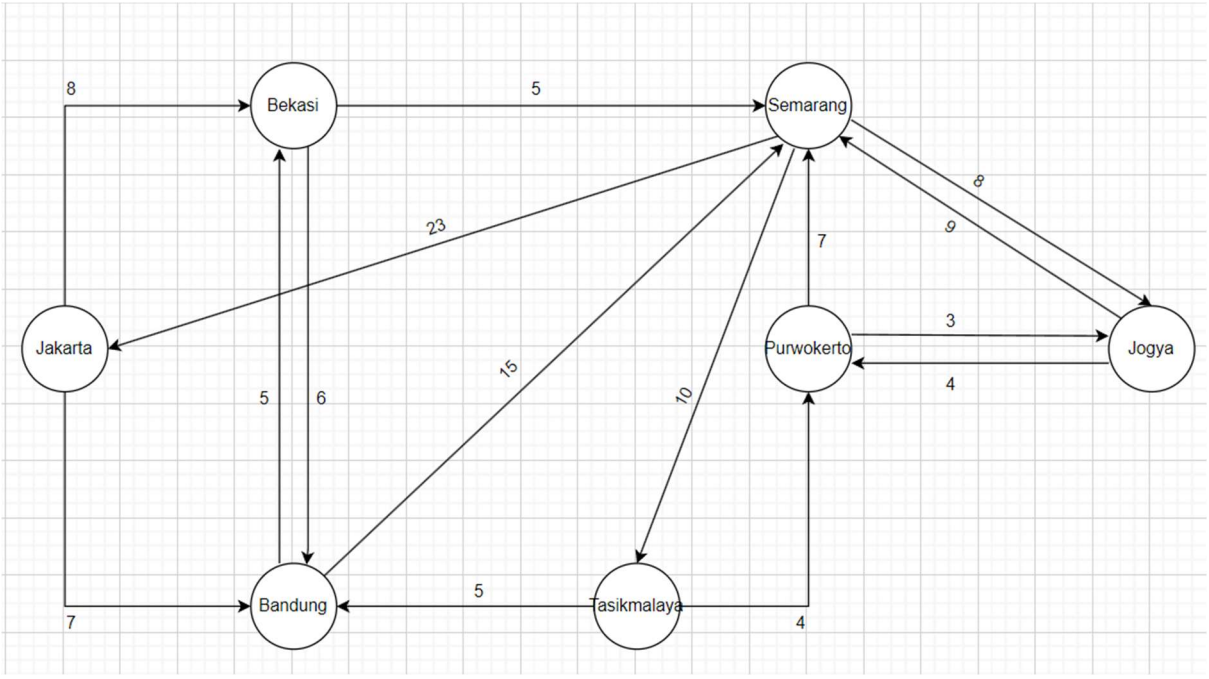
```
Silahkan masukkan jumlah kota : 7
Silahkan masukkan nama simpul
Kota 1 : bandung
Kota 2 : bekasi
Kota 3 : jakarta
Kota 4 : purwokerto
Kota 5 : semarang
Kota 6 : tasikmalaya
Kota 7 : yogyakarta
Silahkan masukkan bobot antar simpul
bandung --> bandung : 99
bandung --> bekasi : 5
bandung --> jakarta : 99
bandung --> purwokerto : 99
bandung --> semarang : 15
bandung --> tasikmalaya : 99
bandung --> yogyakarta : 99
bekasi --> bandung : 6
bekasi --> bekasi : 99
bekasi --> jakarta : 99
bekasi --> purwokerto : 99
bekasi --> semarang : 5
bekasi --> tasikmalaya : 99
bekasi --> yogyakarta : 99
jakarta --> bandung : 7
jakarta --> bekasi : 8
jakarta --> jakarta : 99
jakarta --> purwokerto : 99
jakarta --> semarang : 99
jakarta --> tasikmalaya : 99
jakarta --> yogyakarta : 99
```

```
tasikmalaya --> purwokerto : 4
tasikmalaya --> semarang : 99
tasikmalaya --> tasikmalaya : 99
tasikmalaya --> yogyakarta : 99
yogyakarta --> bandung : 99
yogyakarta --> bekasi : 99
yogyakarta --> jakarta : 99
yogyakarta --> purwokerto : 4
yogyakarta --> semarang : 9
yogyakarta --> tasikmalaya : 99
yogyakarta --> yogyakarta : 99
bandung : bandung : 99 bekasi : 5 jakarta : 99 purwokerto : 99 semarang : 15 tasikmalaya : 99 yogyakarta : 99
bekasi : bandung : 6 bekasi : 99 jakarta : 99 purwokerto : 99 semarang : 5 tasikmalaya : 99 yogyakarta : 99
jakarta : bandung : 7 bekasi : 8 jakarta : 99 purwokerto : 99 semarang : 99 tasikmalaya : 99 yogyakarta : 99
purwokerto : bandung : 99 bekasi : 99 jakarta : 99 purwokerto : 99 semarang : 7 tasikmalaya : 99 yogyakarta : 3
semarang : bandung : 99 bekasi : 99 jakarta : 23 purwokerto : 99 semarang : 99 tasikmalaya : 10 yogyakarta : 8
tasikmalaya : bandung : 5 bekasi : 99 jakarta : 99 purwokerto : 4 semarang : 99 tasikmalaya : 99 yogyakarta : 99
yogyakarta : bandung : 99 bekasi : 99 jakarta : 99 purwokerto : 4 semarang : 9 tasikmalaya : 99 yogyakarta : 99
PS C:\Users\hamka\O9SDSSe\Documents\PRAKTIKUM STRUKTUR DATA\MODUL 11> 9
```

UNGUIDED

1. menjelaskan dengan gambar urutan setiap proses yang terjadi ketika fungsi buatSimpulGraph() pada tugas guided IV dijalankan. (disini mengunakan graph pada Guided 1)

Untuk membuat gambaran nya disini simpul-simpul untuk setiap kota dibuat satu per satu dan dihubungkan dengan kanan dan kiri. data Simpul berisi nama-nama kota:Jakarta, Bekasi, Bandung, Tasikmalaya, Purwokerto, Semarang, Yogyakarta. Graph ini menunjukkan simpul-simpul dengan busur yang terhubung sesuai dengan bobot/nilai yang ada pada guided IV. Setiap simpul merepresentasikan kota, dan busur menunjukkan jalur antar kota dengan bobot/nilai yang sama dengan di guided IV. Langkah langkah dalam penginisiaan dan initrasian setiap simpul nya



2. menambahkan fungsi untuk mencari busur terpendek pada program pada tugas guided IV. (disni menggunakan graph pada Guided 1)

source Cod

```
#include <iostream>
#include <string>

using namespace std;

const int MAX_SIMPUL = 10;
const int INF = 999999;

int jumlahSimpul = 5;
string *dataSimpul;
int **dataBusur;
bool cekMatrik = false;

struct graph {
    graph *kanan;
    string data;
    graph *kiri;
};

graph *simpul;
graph *awal;
graph *akhir;
graph **alamat;
graph *helperA;
graph *helperB;

void inisiasi() {
    awal = NULL;
    akhir = NULL;
}

bool graphKosong() {
    if (awal == NULL && akhir == NULL) {
        return true;
    } else {
        return false;
    }
}

void buatMatriks() {
    dataSimpul = new string[MAX_SIMPUL];
    dataBusur = new int*[MAX_SIMPUL];
    dataBusur[0] = new int[MAX_SIMPUL * MAX_SIMPUL];

    for (int i = 1; i < MAX_SIMPUL; i++) {
        dataBusur[i] = dataBusur[i-1] + MAX_SIMPUL;
```

```
    }

    cout << "Silahkan masukkan nama simpul " << endl;
    for (int i = 0; i < jumlahSimpul; i++) {
        cout << "Kota " << i + 1 << " : ";
        cin >> dataSimpul[i];
    }

    cout << "Silahkan masukkan bobot antar simpul " << endl;
    for (int baris = 0; baris < jumlahSimpul; baris++) {
        for (int kolom = 0; kolom < jumlahSimpul; kolom++) {
            cout << dataSimpul[baris] << " --> " << dataSimpul[kolom] << " :
";
            cin >> dataBusur[baris][kolom];
        }
    }
    cekMatrik = true;
}

void buatSimpulGraph() {
    alamat = new graph*[MAX_SIMPUL];
    buatMatriks();
    for (int i = 0; i < jumlahSimpul; i++) {
        if (graphKosong()) {
            simpul = new graph;
            simpul->data = dataSimpul[i];
            simpul->kanan = NULL;
            simpul->kiri = NULL;
            awal = simpul;
            akhir = simpul;
            alamat[i] = awal;
        } else {
            simpul = new graph;
            simpul->data = dataSimpul[i];
            akhir->kiri = simpul;
            akhir = simpul;
            simpul->kiri = NULL;
            simpul->kanan = NULL;
            alamat[i] = akhir;
        }
    }

    helperA = awal;
    for (int baris = 0; baris < jumlahSimpul; baris++) {
        helperB = helperA;
        for (int kolom = 0; kolom < jumlahSimpul; kolom++) {
            if (dataBusur[baris][kolom] != 0) {
                simpul = new graph;
```

```
                simpul->data = to_string(dataBusur[baris][kolom]);
                helperB->kanan = simpul;
                simpul->kiri = alamat[kolom];
                simpul->kanan = NULL;
                helperB = simpul;
            }
        }
        helperA = helperA->kiri;
    }
}

void tampilGraph() {
    if (!graphKosong()) {
        helperA = awal;
        while (helperA != NULL) {
            cout << helperA->data << " : ";
            helperB = helperA->kanan;
            while (helperB != NULL) {
                cout << helperB->kiri->data << " : " << helperB->data << " ";
                helperB = helperB->kanan;
            }
            cout << endl;
            helperA = helperA->kiri;
        }
    } else {
        cout << "Graph kosong...!!!" << endl;
    }
}

void dijkstra(int src, int dest) {
    int dist[MAX_SIMPUL];
    bool sptSet[MAX_SIMPUL];
    int parent[MAX_SIMPUL];

    for (int i = 0; i < jumlahSimpul; i++) {
        dist[i] = INF;
        sptSet[i] = false;
        parent[i] = -1;
    }

    dist[src] = 0;

    for (int count = 0; count < jumlahSimpul - 1; count++) {
        int u = -1;

        for (int i = 0; i < jumlahSimpul; i++) {
            if (!sptSet[i] && (u == -1 || dist[i] < dist[u])) {
                u = i;
```

```
            }
        }

        sptSet[u] = true;

        for (int v = 0; v < jumlahSimpul; v++) {
            if (!sptSet[v] && dataBusur[u][v] && dist[u] != INF && dist[u] +
dataBusur[u][v] < dist[v]) {
                dist[v] = dist[u] + dataBusur[u][v];
                parent[v] = u;
            }
        }
    }

    cout << "Jarak terpendek dari " << dataSimpul[src] << " ke " <<
dataSimpul[dest] << " adalah: " << dist[dest] << endl;
    cout << "Jalur: ";
    int crawl = dest;
    while (parent[crawl] != -1) {
        cout << dataSimpul[crawl];
        crawl = parent[crawl];
        if (crawl != -1) {
            cout << " <- ";
        }
    }
    cout << endl;
}

int main() {
    inisiasi();
    cout << "Silahkan masukkan jumlah kota: ";
    cin >> jumlahSimpul;
    buatSimpulGraph();
    tampilGraph();

    int src, dest;
    cout << "Masukkan nomor kota awal (0 - " << jumlahSimpul - 1 << "): ";
    cin >> src;
    cout << "Masukkan nomor kota tujuan (0 - " << jumlahSimpul - 1 << "): ";
    cin >> dest;

    dijkstra(src, dest);

    return 0;
}
```

Untuk mencari jalur tercepat disini mengunakan algoritma Dijkstra, Algoritma Dijkstra adalah sebuah algoritma yang digunakan untuk menentukan jalur terpendek dalam graph berarah yang memiliki bobot non-negatif. Cara kerja algoritma ini adalah dengan memilih titik asal dan menghitung jarak terpendek ke setiap titik lain dalam graph. Proses ini diulangi hingga semua titik dalam graph telah dikunjungi, dan hasilnya adalah jarak terpendek antara titik asal dan setiap titik lain dalam graph.

Dalam implematasi algoritma ini pada program unguided di atas dilakukan dalam fungsi ‘dijkstra(int src, int dest)’ inisialisasi Variabel Pada awal fungsi, ‘variabel dist, sptSet’, dan ‘parent’ diinisialisasi. Array ‘dist’ digunakan untuk menyimpan jarak terpendek dari titik awal (src) ke setiap titik lain dalam graph. Array ‘sptSet’ digunakan untuk menandai apakah suatu titik sudah termasuk dalam set jalur terpendek dengan mengunnakan :

1. Proses iterasi dilakukan sebanyak ‘jumlahSimpul’ - 1 kali, karena setiap iterasi akan menambah satu titik ke dalam ‘SPT.
2. Pemilihan TitikDi setiap iterasi, titik yang belum termasuk dalam ‘SPT dan memiliki jarak terpendek dari titik awal dipilih. Titik tersebut ditandai sebagai bagian dari SPT.
3. Cetak Jalur Terpendek Setelah proses iterasi selesai, memiliki informasi tentang jarak terpendek dari titik awal ke semua titik lain dalam graph dan jalur terpendek yang ditemukan. Lalu mencetak jarak terpendek antara titik awal dan tujuan (dest), serta jalur yang dilalui untuk mencapainya dengan pernyataan ‘cout’.

Hasil Program

```
Silahkan masukkan jumlah kota: 7
Silahkan masukkan nama simpul
Kota 1 : bandung
Kota 2 : bekasi
Kota 3 : jakata
Kota 4 : purwokerto
Kota 5 : semarang
Kota 6 : tasikmalaya
Kota 7 : yogyakarta
Silahkan masukkan bobot antar simpul
bandung --> bandung : 99
bandung --> bekasi : 5
bandung --> jakata : 99
bandung --> purwokerto : 99
bandung --> semarang : 15
bandung --> tasikmalaya : 99
bandung --> yogyakarta : 99
bekasi --> bandung : 6
bekasi --> bekasi : 99
bekasi --> jakata : 99
bekasi --> purwokerto : 99
bekasi --> semarang : 5
bekasi --> tasikmalaya : 99
bekasi --> yogyakarta : 99
jakata --> bandung : 7
jakata --> bekasi : 8
jakata --> jakata : 99
jakata --> purwokerto : 99
jakata --> semarang : 99
jakata --> tasikmalaya : 99
jakata --> yogyakarta : 99
purwokerto --> bandung : 99
```

```
purwokerto --> bekasi : 99
tasikmalaya --> jakata : 99
tasikmalaya --> purwokerto : 4
tasikmalaya --> semarang : 99
tasikmalaya --> tasikmalaya : 99
tasikmalaya --> yogyakarta : 99
yogyakarta --> bandung : 99
yogyakarta --> bekasi : 99
yogyakarta --> jakata : 99
yogyakarta --> purwokerto : 4
yogyakarta --> semarang : 9
yogyakarta --> tasikmalaya : 99
yogyakarta --> yogyakarta : 99
bandung : bandung : 99 bekasi : 5 jakata : 99 purwokerto : 99 semarang : 15 tasikmalaya : 99 yogyakarta : 99
bekasi : bandung : 6 bekasi : 99 jakata : 99 purwokerto : 99 semarang : 5 tasikmalaya : 99 yogyakarta : 99
jakata : bandung : 7 bekasi : 8 jakata : 99 purwokerto : 99 semarang : 99 tasikmalaya : 99 yogyakarta : 99
purwokerto : bandung : 99 bekasi : 99 jakata : 99 purwokerto : 99 semarang : 7 tasikmalaya : 99 yogyakarta : 3
semarang : bandung : 99 bekasi : 99 jakata : 23 purwokerto : 99 semarang : 99 tasikmalaya : 10 yogyakarta : 8
tasikmalaya : bandung : 5 bekasi : 99 jakata : 99 purwokerto : 4 semarang : 99 tasikmalaya : 99 yogyakarta : 99
yogyakarta : bandung : 99 bekasi : 99 jakata : 99 purwokerto : 4 semarang : 9 tasikmalaya : 99 yogyakarta : 99
Masukkan nomor kota awal (0 - 6): 1
Masukkan nomor kota tujuan (0 - 6): 4
Jarak terpendek dari bekasi ke semarang adalah: 5
Jalur: semarang <-
```

KESIMPULAN

Graph dalam pemrograman adalah cara untuk menghubungkan data seperti titik-titik (disebut simpul atau nodes) dengan garis-garis (disebut tepi atau edges). Bayangkan graph seperti peta jalan, di mana kota-kota adalah simpul dan jalan-jalan yang menghubungkan kota-kota tersebut adalah tepi. Ada dua jenis graph: graph berarah, di mana garis memiliki arah seperti jalan satu arah, dan graph tak berarah, di mana garis tidak memiliki arah seperti jalan dua arah. Graph berguna untuk memecahkan berbagai masalah seperti menemukan rute terpendek, penjadwalan tugas, atau menghubungkan jaringan komputer.

REFRENSI

1. MODUL 11 STRUKTUR DATA
2. https://www.softwaretestinghelp.com/graph-implementation-cpp/
3. https://www.baeldung.com/cs/graphs-directed-vs-undirected-graph
4. https://mti.binus.ac.id/2017/11/28/algoritma-dijkstra/