LAPORAN PRAKTIKUM STRUKTUR DATA
MODUL 12

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.

$$G = (V,E)$$

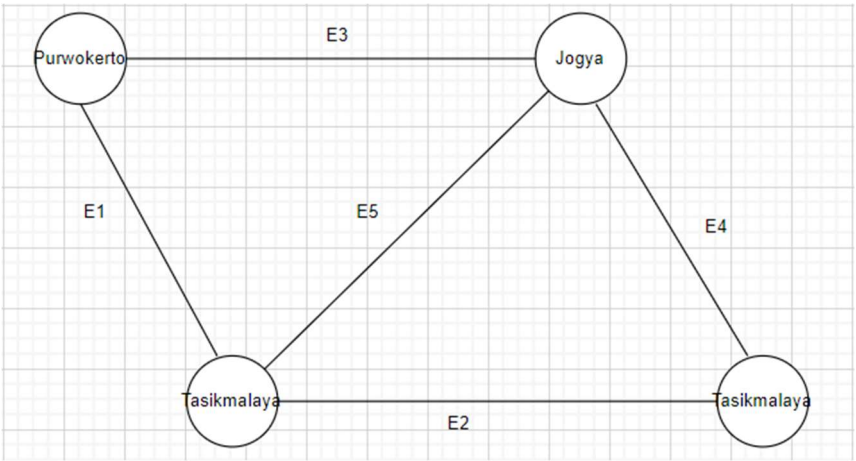
Yang artinnya bahwa

‘G’ sebagai Graph

‘V’ sebagai simpul atau busur atau node atau titik

‘E’ sebagai edge atau busur atau ruas

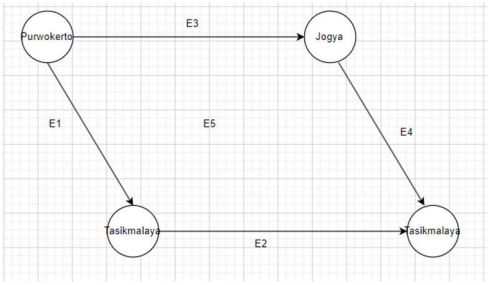
Dan jika di implmentasikan pada sebuah gambar (dibuat dengan platform drawio)



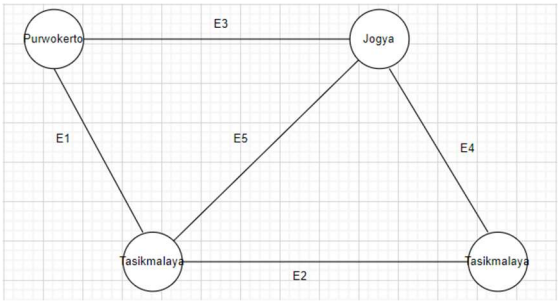
2. Jenis graph

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

II. Perbedaan antara Algoritma Warshall (Floyd-Warshall) dan Algoritma Dijkstra

Pencarian jalur terpendek pada garph adalah proses menemukan jalur dengan total bobot terkecil dari satu simpul (atau node) ke simpul lainnya dalam sebuah graf. Graf adalah struktur matematika yang terdiri dari simpul (node atau vertex) dan tepi (edge) yang menghubungkan simpul-simpul tersebut. Tepi-tepi tersebut mungkin memiliki bobot (weight).

Untuk mencari jalur terpendek nya ada algoritma yang bernama warshall dan djikstra kedua nya adalah algoritma mencari jalut terpendek pada garph kedua nya memiliki masing masing fungsi yang berbeda dan source code berbeda.

a. Algoritma Djikstra

   Algoritma Dijkstra digunakan hanya untuk menemukan jalur terpendek dari satu simpul sumber ke semua simpul lainnya yang di tentukan dalam graph berarah dengan bobot positif. Pada setiap langkah kerja nya, mengambil sisi yang berbobot minimum yang menghubungkan sebuah simpul yang telah dikunjungi dengan sebuah simpullain yang belum dikunjungi. Lintasan dari simpul asal ke simpul yang baru haruslah merupakan lintasan yang terpendek diantara semua lintasannya keimpul-simpul yang belum dikunjungi. Ini hanya mencari jalur terpendek dari node tertentu bukan dari semua node yang ada pada garph.

Contoh output yang kan di dapat kan jika mengunakan algoritma ini dalam mencari jalur terpendek garph

```
Silahkan masukkan simpul asal (0 - 5) : 0
Silahkan masukkan simpul tujuan (0 - 5) : 5
Jarak terdekat dari 0 ke 5 adalah 7
```

Hanya mencari jalur terpendek dari node yang di tentukan oleh user

b. Algoritma Floyd-Warshall

Algoritma ini bertujuan untuk mencari jalur terpendek dari semua node yang ada pada garph sehingga hasil yang menjadi output adalah susunan matriks yang isi bobot dari semua jalur antar node ini juga digunakan untuk mencari jalur terpendek antara semua pasangan simpul dalam graf berbobot. Salah satu keunggulan algoritma Floyd-Warshall adalah kesederhanaan implementasinya. Algoritma ini menggunakan tiga lapis loop bersarang, yang membuat kode lebih mudah dipahami dan diimplementasikan jika dibandingkan dengan algoritma djikstra.

Contoh output yang kan di dapat kan jika mengunakan algoritma ini dalam mencari jalur terpendek.

```
 --- bandung bekasi jakarta purwokerto semarang tasikmalaya jogyakarta
bandung | 0 5 33 15 10 20 18
bekasi | 6 0 28 17 5 15 13
jakarta | 7 8 0 22 13 23 21
purwokerto | 22 27 30 0 7 17 3
semarang | 15 20 23 12 0 10 8
tasikmalaya | 5 10 34 4 11 0 7
jogyakarta | 24 29 32 4 9 19 0
PS C:\Users\hamka\OneDrive\Documents\PRAKTIKUM STRUKTUR DATA\MODUL 12>
```

Output nya akan menampilkan sususan matriks yang berisi bobot dari setiap node cara membaca nya ialah dari kolom lalu menentukan akan pergi kemana sesuai kan dngan baris kota kota di atas lalu ambil bobot nilai yang berada pada sudut antara kolom dan baris itulah jarak terpendek nya, contoh saya dari bekasi dan ingin ke Jakarta maka jarak terpendek nya adalah 28, karana 28 ada pada sudut siku kedua nya

GUIDED

Membuat program pencarian lintasan terpendek dengan menerapkan yang menggunakan algoritma Djikstra.

source code

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

using namespace std;

int jumlahSimpul = 5;
string *dataSimpul;
int **dataBusur;
bool cekMatrik = false;
int indeksPosisi, simpulSaatIni, simpulAsal, simpulTujuan, jarakSaatIni,
jarakLama, jarakBaru;
int dikunjungi = 1;
int belumDikunjungi = 0;
int *jarakDiketahui;
int *kunjungan;

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 hitungJarakTerdekat(){
    if(cekMatrik){
        jarakDiketahui = new int[jumlahSimpul];
        kunjungan = new int[jumlahSimpul];
        for(int i = 0; i < jumlahSimpul; i++){
            jarakDiketahui[i] = 999; //Nilai 999 dianggap sebagai infinity
atau tak hingga
            kunjungan[i] = belumDikunjungi;
        }

        kunjungan[simpulAsal] = dikunjungi;
        jarakDiketahui[simpulAsal] = 0;
        simpulSaatIni = simpulAsal;

        while(simpulSaatIni != simpulTujuan){
            jarakLama = 999;
            jarakSaatIni = jarakDiketahui[simpulSaatIni];
            for(int i = 0; i < jumlahSimpul; i++){
                if(kunjungan[i] == belumDikunjungi){
                    jarakBaru = jarakSaatIni + dataBusur[simpulSaatIni][i];
                    if(jarakBaru < jarakDiketahui[i]){
                        jarakDiketahui[i] = jarakBaru;
                    }
                    if(jarakDiketahui[i] < jarakLama){
                        jarakLama = jarakDiketahui[i];
                        indeksPosisi = i;
                    }
                }
            }
            simpulSaatIni = indeksPosisi;
            kunjungan[simpulSaatIni] = dikunjungi;
        }
        cout<<"Jarak terdekat dari "<<simpulAsal<<" ke "<<simpulTujuan<<"
adalah "<<jarakDiketahui[simpulTujuan]<<endl;
        delete jarakDiketahui;
        delete kunjungan;
    }
}

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(){
    char keluar;
    cout<<"Silahkan masukkan jumlah kota (angka) : ";
    cin>>jumlahSimpul;
    buatMatriks();
    tampilMatriks();

    do{
        cout<<"Silahkan masukkan simpul asal (0 - "<<jumlahSimpul-1<<") : ";
        cin>>simpulAsal;
        cout<<"Silahkan masukkan simpul tujuan (0 - "<<jumlahSimpul-1<<") : ";
        cin>>simpulTujuan;
        hitungJarakTerdekat();

        cout<<endl<<endl;
        cout<<"Keluar (y/t) ? : ";
        cin>>keluar;

        if(tolower(keluar) != 'y'){
            system("cls");
        }
    }while(tolower(keluar) != 'y');

    return 0;
}
```

Analisis

Untuk mencari lintas terpendek antarkota yang harus pertama dilakukan ialah merepsentasi graphnya dengan matriks. Untuk mereprentasikan nya code di atas mengunakan fungsi ‘buatmatrik’ yang akan mengalokasikan array nama simpul dan mengalokasikan array pointer untuk baris bari dalam matriks Dibutuhkan array dua dimensi yang masing masing yang nanati akan di masukan jumlah nya user ketika indeks dari matrik sudah ada maka program akan melooping sesuai dengan indeks nya menggunakan

```
for(int i = 0; i < jumlahSimpul; i++){
    cout << "Kota " << i+1 << " : ";
    cin >> dataSimpul[i];
```

setelah terisi semua indeks nya maka user diminta untuk memasukan nilai bobot antar Simpul

section ini akan terus terlooping sampai semua tereksekusi karan terdapat dua looping bersarang

1. for(int baris = 0; baris < jumlahSimpul; baris++){ Looping luar
2. for(int kolom = 0; kolom < jumlahSimpul; kolom++) Looping dalam

setelah semua simpul tereksekusi program akan mengubah variabel ‘cekmatrki’ menjadi true menjadi indikasi untuk melanjutkan ke section berikutnya yaitu ‘hitungjarakterdekat’

mengecek apakah matriks telah di buat ‘if(cekMatrik)’ lalu mengalokasikan array untuk statatus kunjungan simpul, untuk menerapkan algoritma dijkstar guna mencari jalur terpendek ‘while(simpulSaatIni != simpulTujuan)’ untuk mengecek simpul saaat ini dan simpul yang dituju program akan mengecek emua jalur dan menetutakan jalur terpendek atau jalur dengan nilai terkecil

jarak Baru = jarakSaatIni + dataBusur[simpulSaatIni][i]; setelah jalur terpendek ditemukan fungsi ‘cekmatriks’ akan menampilkan nama simpul dan bobot antar simpul terpendek mengunakan perulangan for dan menampilkannya pada fungsi ‘hitungjarakterdekat’

```
cout<<"Jarak terdekat dari "<<simpulAsal<<" ke "<<simpulTujuan<<" adalah "<<jarakDiketahui[simpulTujuan]<<endl;
```

didalam variebel ‘int main’ semua fungi dipanggil untuk diekseskusi dengan varibel ‘jumlahsimpul’ ini memasukan data yang di input user pada fungi fungsi di atas untuk di proses dan ditampilkan pada terminal menjadi output

Hasil Program

```
Silahkan masukkan jumlah kota (angka) : 6
Silahkan masukkan nama simpul
Kota 1 : A
Kota 2 : B
Kota 3 : C
Kota 4 : D
Kota 5 : E
Kota 6 : F
Silahkan masukkan bobot antar simpul
A --> A : 0
A --> B : 5
A --> C : 99
A --> D : 2
A --> E : 99
A --> F : 99
B --> A : 6
B --> B : 0
B --> C : 99
B --> D : 7
B --> E : 99
B --> F : 5
C --> A : 99
C --> B : 3
C --> C : 0
C --> D : 99
C --> E : 4
C --> F : 2
D --> A : 99
D --> B : 99
D --> C : 3
D --> D : 0
D --> E : 7
```

```
D --> E : 7
D --> F : 99
E --> A : 99
E --> B : 99
E --> C : 99
E --> D : 99
E --> E : 0
E --> F : 7
F --> A : 99
F --> B : 99
F --> C : 99
F --> D : 99
F --> E : 99
F --> F : 0
A B C D E F
0 5 99 2 99 99
6 0 99 7 99 5
99 3 0 99 4 2
99 99 3 0 7 99
99 99 99 99 0 7
99 99 99 99 99 0
Silahkan masukkan simpul asal (0 - 5) : 0
Silahkan masukkan simpul tujuan (0 - 5) : 5
Jarak terdekat dari 0 ke 5 adalah 7


Keluar (y/t) ? : Y
```

GUIDED 2

Jika pada guided 1 mencari lintaan terpendek mengunakan algoritma djiktra sekrang kan membuat program pencarian lintasan terpendek dengan menerapkan algoritma Flyod Warshall.Manual input loohh

Source Code

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

using namespace std;

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

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 hitungJarakTerdekat(){
    if(cekMatrik){
        //Membuat matrik yang sama dengan matrik dataBusur
        jalurTerdekat = new int*[jumlahSimpul];
        jalurTerdekat[0] = new int[jumlahSimpul * jumlahSimpul];
```

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

        //Duplikasi isi matrik dataBusur kedalam matrik jalurTerdekat
        for(int baris = 0; baris < jumlahSimpul; baris++){
            for(int kolom = 0; kolom < jumlahSimpul; kolom++){
                jalurTerdekat[baris][kolom] = dataBusur[baris][kolom];
            }
        }

        //Pencarian jalur terdekat dengan algoritma Flyod Warshall
        for(int k = 0; k < jumlahSimpul; k++){
            for(int baris = 0; baris < jumlahSimpul; baris++){
                for(int kolom = 0; kolom < jumlahSimpul; kolom++){
                    if(jalurTerdekat[baris][k] + jalurTerdekat[k][kolom] <
jalurTerdekat[baris][kolom]){
                        jalurTerdekat[baris][kolom] = jalurTerdekat[baris][k]
+ jalurTerdekat[k][kolom];
                    }
                }
            }
        }

        //Tampilkan hasil
        cout<<" --- ";
        for(int kolom = 0; kolom < jumlahSimpul; kolom++){
            cout<<dataSimpul[kolom]<<" ";
        }
        cout<<endl;

        for(int baris = 0; baris < jumlahSimpul; baris++){
            cout<<dataSimpul[baris]<<" | ";
            for(int kolom = 0; kolom < jumlahSimpul; kolom++){
                cout<<jalurTerdekat[baris][kolom]<<" ";
            }
            cout<<endl;
        }
    }
}

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 kota : ";
    cin>>jumlahSimpul;
    buatMatriks();
    tampilMatriks();
    cout<<endl<<endl;
    hitungJarakTerdekat();
    return 0;
}
```

Analisis source code

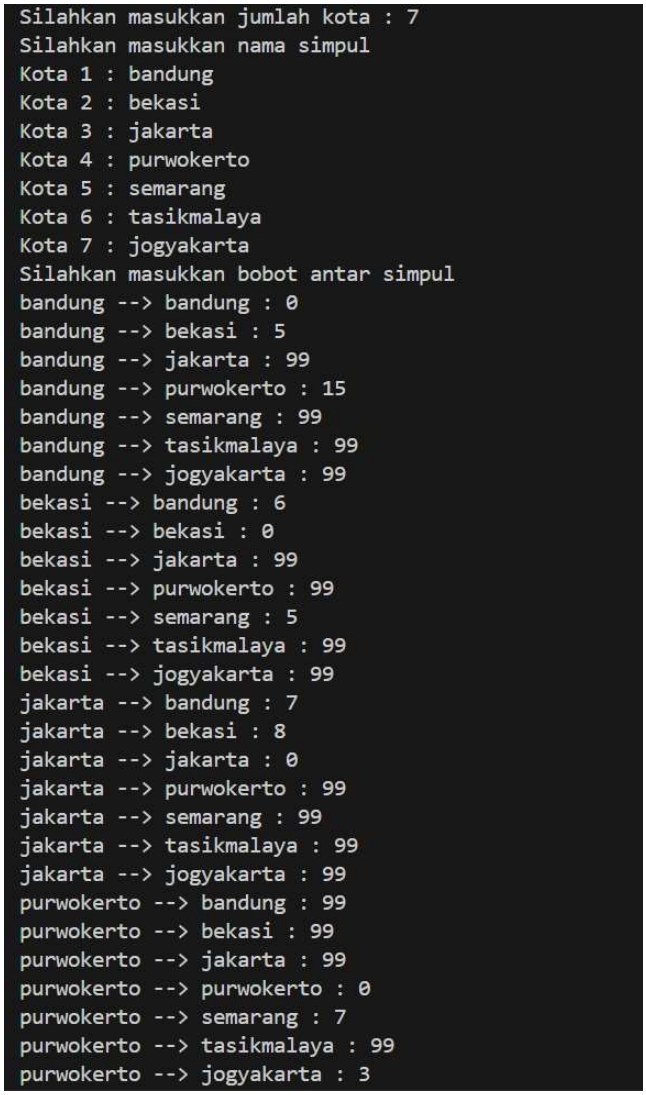
Disini pertama tama dan paling utama mari lah kita menginisiasi varibel nya pada code di atas terdapat variabel untuk menunjang sistem program ada ‘jumlahSimpul’ Guana menyimpan jumlah simpul dalam graph ‘dataSimpul’ guna Menyimpan nama-nama simpul. ‘dataBusur’ m guna menyimpan bobot antara simpul-simpul. ‘cekMatrik’ mengecek apakah matriks telah diisi dan diberi nilai false ‘jalurTerdekat’ Menyimpan hasil perhitungan jarak terdekat antara setiap pasangan simpul.

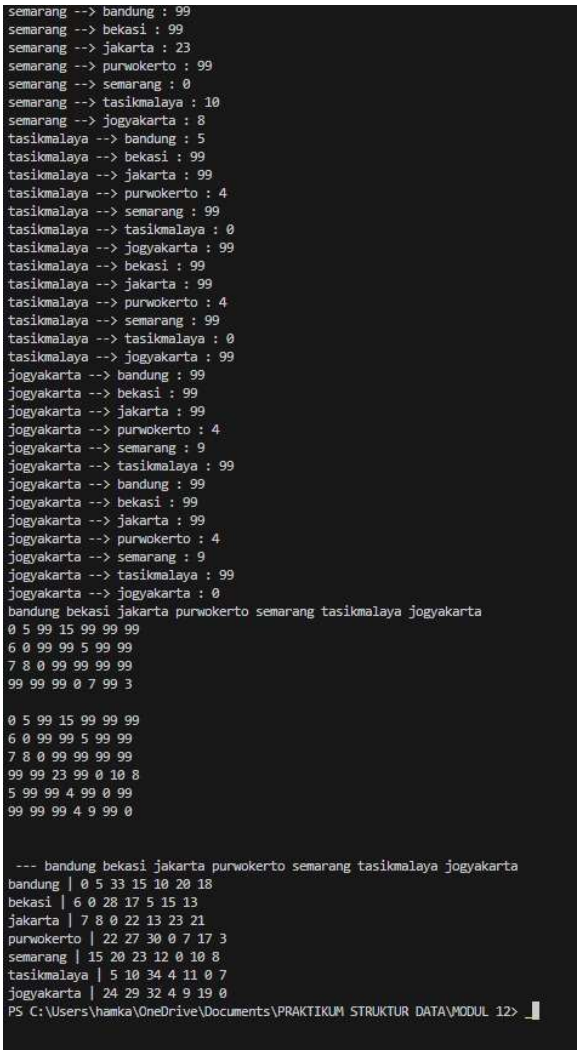
code di atas mengunakan fungsi ‘buatmatrik’ yang akan mengalokasikan array nama simpul dan mengalokasikan array pointer untuk baris bari dalam matriks Dibutuhkan array dua dimensi yang masing masing yang nanati akan di masukan jumlah nya user ketika indeks dari matrik sudah ada maka program akan melooping sesuai dengan indeks nya yang di inputkan pengguna

pada ‘funsi hitung jarak terpendek’ ini lah ynag membedakan algioritma wrshall ada algoritma ini umum nya menggunakan matriks adjacency (vector<vector<int>>) untuk menyimpan bobot antara simpul-simpul yang berarti Algoritma Warshall memanfaatkan matriks adjacency untuk merepresentasikan graf dan mencari jalur terdekat. yang mana disni mengalokasikan memori untuk matriks ‘jalurTerdekat’, yang merupakan salinan dari matriks ‘dataBusur’ lalu menyalin nilai-nilai dari ‘dataBusur’ ke ‘jalurTerdekat’.

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 terstruktur jika matriks belum dibuat output dari ini adalah susunan matriks kolom 7 baris 7 sesuai dengan data ng di inputkan pengguna, fungsi akan menampilkan pesan ‘Tidak ada matriks’ untuk memberi tahu pengguna bahwa matriks belum tersedia.





UNGUIDED

1. Modifikasi program pada tugas guided I, agar menampilkan lintasan terpendek dari simpul awal ke simpul tujuan (tidak hanya jaraknya saja), untuk graph yang digunakan silahkan tentukan sendiri, digambar, dan disertakan dalam laporan.

Source code

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

using namespace std;

int jumlahSimpul = 5;
string *dataSimpul;
int **dataBusur;
bool cekMatrik = false;
int indeksPosisi, simpulSaatIni, simpulAsal, simpulTujuan, jarakSaatIni,
jarakLama, jarakBaru;
int dikunjungi = 1;
int belumDikunjungi = 0;
int *jarakDiketahui;
int *kunjungan;
int *sebelumnya; // array to store the previous node

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 hitungJarakTerdekat(){
    if(cekMatrik){
        jarakDiketahui = new int[jumlahSimpul];
        kunjungan = new int[jumlahSimpul];
        sebelumnya = new int[jumlahSimpul];
        for(int i = 0; i < jumlahSimpul; i++){
            jarakDiketahui[i] = 999;
            kunjungan[i] = belumDikunjungi;
            sebelumnya[i] = -1;
        }

        kunjungan[simpulAsal] = dikunjungi;
        jarakDiketahui[simpulAsal] = 0;
        simpulSaatIni = simpulAsal;

        while(simpulSaatIni != simpulTujuan){
            jarakLama = 999;
            jarakSaatIni = jarakDiketahui[simpulSaatIni];
            for(int i = 0; i < jumlahSimpul; i++){
                if(kunjungan[i] == belumDikunjungi){
                    jarakBaru = jarakSaatIni + dataBusur[simpulSaatIni][i];
                    if(jarakBaru < jarakDiketahui[i]){
                        jarakDiketahui[i] = jarakBaru;
                        sebelumnya[i] = simpulSaatIni;
                    }
                    if(jarakDiketahui[i] < jarakLama){
                        jarakLama = jarakDiketahui[i];
                        indeksPosisi = i;
                    }
                }
            }
            simpulSaatIni = indeksPosisi;
            kunjungan[simpulSaatIni] = dikunjungi;
        }

        cout<<"Jarak terdekat dari "<<dataSimpul[simpulAsal]<<" ke
"<<dataSimpul[simpulTujuan]<<" adalah "<<jarakDiketahui[simpulTujuan]<<endl;


        vector<int> path;
        for (int at = simpulTujuan; at != -1; at = sebelumnya[at]) {
            path.push_back(at);
        }
        cout << "dari simpul ";
```

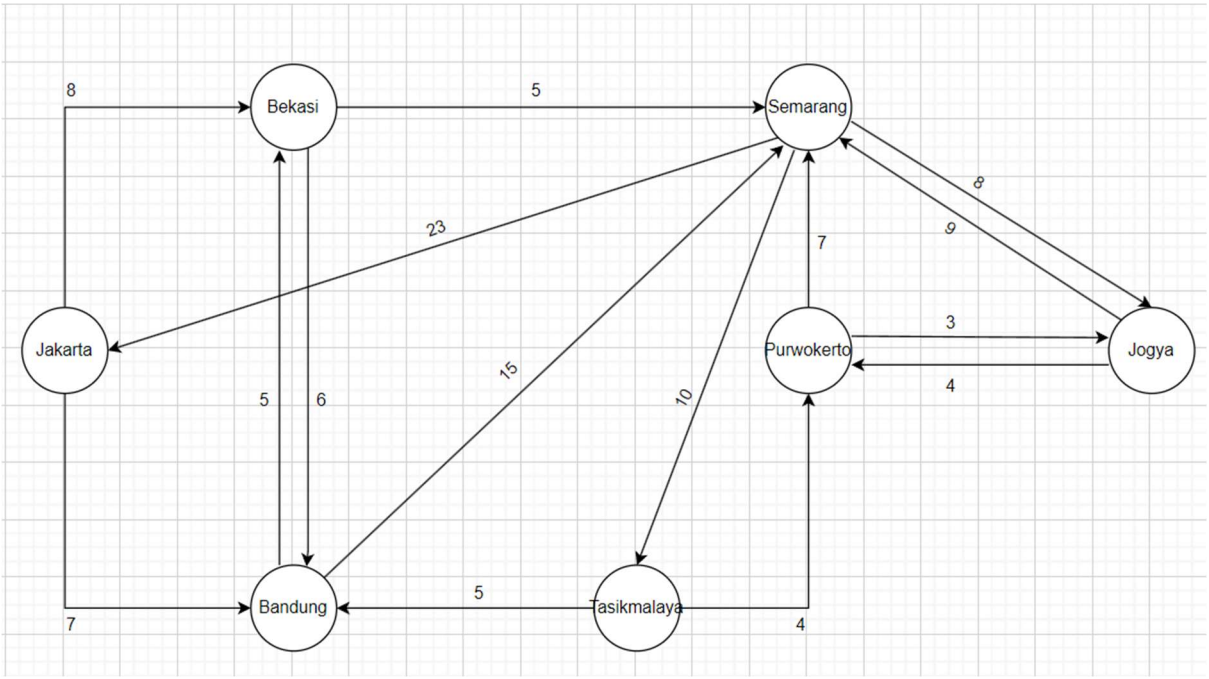
```
        for (int i = path.size() - 1; i >= 0; i--) {
            cout << dataSimpul[path[i]];
            if (i > 0) {
                cout << " melalui ";
            }
        }
        cout << endl;

        delete[] jarakDiketahui;
        delete[] kunjungan;
        delete[] sebelumnya;
    }
}

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(){
    char keluar;
    cout<<"Silahkan masukkan jumlah kota (angka) : ";
    cin>>jumlahSimpul;
    buatMatriks();
    tampilMatriks();

    do{
        cout<<"Silahkan masukkan simpul asal (0 - "<<jumlahSimpul-1<<") : ";
        cin>>simpulAsal;
        cout<<"Silahkan masukkan simpul tujuan (0 - "<<jumlahSimpul-1<<") : ";
        cin>>simpulTujuan;
        hitungJarakTerdekat();


        cout<<endl<<endl;
        cout<<"Keluar (y/t) ? : ";
```

```
        cin>>keluar;

        if(tolower(keluar) != 'y'){
            system("cls");
        }
    }while(tolower(keluar) != 'y');

    return 0;
}
```

Disini untuk menampilakan nama dari node yang di inputkan menggunakan lebrary <vector> karna ukuran vector dapat berubah-ubah sesuai kebutuhan saat runtime. Sehingga sangat mudah digunakan karena menyediakan berbagai metode yang mempermudah manipulasi data seperti menampilkan nama simpul dan jalur jalur yang di lalui untuk mencari jalan tercepat dalam garph.

di program di atas cara bagaiamana program dapat menampilkan nam jalur yang di lewati dengan fungsi dari library <vector> ialah mendeklarasikan vector dengan cara vector<int> path; untuk menyimpan jalur (path) dari simpul tujuan ke simpul asal dan untuk membaca elemen dari vector menggunakan indeks lalu menampilkan ‘dari simpul,, mirip seperti array. Misalnya, path[i] untuk membaca elemen di posisi ‘i’ dan jika ‘i’ kurang dari 0 maka program akan mempilkan output “melalui, yang mana akan terletak di antara nama nama node yang menjadi lintsan untuk mencari jalur terpendek ini lah yang bisa menapilkan nam node yang di lalui ketika mencari jalur tependek.

Lalu pada fungsi int main () fungsi dari ‘hitungjarakterdekat’ akan di panggil untuk mengeksekusi data yang di masukan pengguna seperti jumlah dari node, nama node dan bobot jalur antar simpul, setelah semua nya di deklarasikan pengguna akan di mita untuk menentukan jalur terpendek dari simpul tertentu lalu program akan mengeksekusi nya lalu menampilkan hasil dari pencarian jalur tependek. Disini saya akan memasukan 6 indeks yang nama simpul nya adalah nama nama kota dan bobot jalur nya bisa di ibaratkan adlh jarak antar kota tersebut



HASIL PROGRAM

```
Silahkan masukkan jumlah kota (angka) : 7
Silahkan masukkan nama simpul
Kota 1 : jakarta
Kota 2 : bekasi
Kota 3 : semarang
Kota 4 : jogyakarta
Kota 5 : purwokerto
Kota 6 : tasikmalaya
Kota 7 : bandung
Silahkan masukkan bobot antar simpul
jakarta --> jakarta : 0
jakarta --> bekasi : 8
jakarta --> semarang : 99
jakarta --> jogyakarta : 99
jakarta --> purwokerto : 99
jakarta --> tasikmalaya : 99
jakarta --> bandung : 7
bekasi --> jakarta : 99
bekasi --> bekasi : 0
bekasi --> semarang : 5
bekasi --> jogyakarta : 99
bekasi --> purwokerto : 99
bekasi --> tasikmalaya : 99
bekasi --> bandung : 6
semarang --> jakarta : 23
semarang --> bekasi : 99
semarang --> semarang : 0
semarang --> jogyakarta : 8
semarang --> purwokerto : 99
semarang --> tasikmalaya : 10
semarang --> bandung : 99
jogyakarta --> jakarta : 99
jogyakarta --> bekasi : 99
jogyakarta --> semarang : 9
jogyakarta --> jogyakarta : 0
jogyakarta --> purwokerto : 4
jogyakarta --> tasikmalaya : 99
jogyakarta --> bandung : 99
```

```
purwokerto --> jakarta : 99
purwokerto --> bekasi : 99
purwokerto --> semarang : 7
purwokerto --> jogyakarta : 3
purwokerto --> purwokerto : 0
purwokerto --> tasikmalaya : 99
purwokerto --> bandung : 99
tasikmalaya --> jakarta : 99
tasikmalaya --> bekasi : 99
tasikmalaya --> semarang : 99
tasikmalaya --> jogyakarta : 99
tasikmalaya --> purwokerto : 4
tasikmalaya --> tasikmalaya : 0
tasikmalaya --> bandung : 5
bandung --> jakarta : 99
bandung --> bekasi : 5
bandung --> semarang : 15
bandung --> jogyakarta : 99
bandung --> purwokerto : 99
bandung --> tasikmalaya : 99
bandung --> bandung : 0
jakarta bekasi semarang jogyakarta purwokerto tasikmalaya bandung
0 8 99 99 99 99 7
99 0 5 99 99 99 6
23 99 0 8 99 10 99
99 99 9 0 4 99 99
99 99 7 3 0 99 99
99 99 99 99 4 0 5
99 5 15 99 99 99 0
Silahkan masukkan simpul asal (0 - 6) : 0
Silahkan masukkan simpul tujuan (0 - 6) : 3
Jarak terdekat dari jakarta ke jogyakarta adalah 21
dari simpul jakarta melalui bekasi melalui semarang melalui jogyakarta
```

1. Gabungkan dan perbaiki program pada guided I dan II sehingga menjadi sebuah program yang memiliki menu utama pencarian jalur terpendek dengan Djikstra dan Flyod Warshall.

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

using namespace std;

int jumlahSimpul = 5;
string *dataSimpul;
int **dataBusur;
bool cekMatrik = false;
int indeksPosisi, simpulSaatIni, simpulAsal, simpulTujuan, jarakSaatIni,
jarakLama, jarakBaru;
int dikunjungi = 1;
int belumDikunjungi = 0;
int *jarakDiketahui;
int *kunjungan;
int **jalurTerdekat;

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 hitungJarakTerdekatdjistra(){
    if(cekMatrik){
        jarakDiketahui = new int[jumlahSimpul];
        kunjungan = new int[jumlahSimpul];
        for(int i = 0; i < jumlahSimpul; i++){
            jarakDiketahui[i] = 999; //Nilai 999 dianggap sebagai infinity
atau tak hingga
            kunjungan[i] = belumDikunjungi;
        }

        kunjungan[simpulAsal] = dikunjungi;
        jarakDiketahui[simpulAsal] = 0;
        simpulSaatIni = simpulAsal;

        while(simpulSaatIni != simpulTujuan){
            jarakLama = 999;
            jarakSaatIni = jarakDiketahui[simpulSaatIni];
            for(int i = 0; i < jumlahSimpul; i++){
                if(kunjungan[i] == belumDikunjungi){
                    jarakBaru = jarakSaatIni + dataBusur[simpulSaatIni][i];
                    if(jarakBaru < jarakDiketahui[i]){
                        jarakDiketahui[i] = jarakBaru;
                    }
                    if(jarakDiketahui[i] < jarakLama){
                        jarakLama = jarakDiketahui[i];
                        indeksPosisi = i;
                    }
                }
            }
            simpulSaatIni = indeksPosisi;
            kunjungan[simpulSaatIni] = dikunjungi;
        }
        cout<<"Jarak terdekat dari "<<simpulAsal<<" ke "<<simpulTujuan<<"
adalah "<<jarakDiketahui[simpulTujuan]<<endl;
        delete jarakDiketahui;
        delete kunjungan;
    }
}

void hitungJarakTerdekatwarshall(){
    if(cekMatrik){
        //Membuat matrik yang sama dengan matrik dataBusur
        jalurTerdekat = new int*[jumlahSimpul];
        jalurTerdekat[0] = new int[jumlahSimpul * jumlahSimpul];
        for(int i = 1; i < jumlahSimpul; i++){
            jalurTerdekat[i] = jalurTerdekat[i-1] + jumlahSimpul;
        }
```

```
        //Duplikasi isi matrik dataBusur kedalam matrik jalurTerdekat
        for(int baris = 0; baris < jumlahSimpul; baris++){
            for(int kolom = 0; kolom < jumlahSimpul; kolom++){
                jalurTerdekat[baris][kolom] = dataBusur[baris][kolom];
            }
        }

        //Pencarian jalur terdekat dengan algoritma Flyod Warshall
        for(int k = 0; k < jumlahSimpul; k++){
            for(int baris = 0; baris < jumlahSimpul; baris++){
                for(int kolom = 0; kolom < jumlahSimpul; kolom++){
                    if(jalurTerdekat[baris][k] + jalurTerdekat[k][kolom] <
jalurTerdekat[baris][kolom]){
                        jalurTerdekat[baris][kolom] = jalurTerdekat[baris][k]
+ jalurTerdekat[k][kolom];
                    }
                }
            }
        }

        //Tampilkan hasil
        cout<<" --- ";
        for(int kolom = 0; kolom < jumlahSimpul; kolom++){
            cout<<dataSimpul[kolom]<<" ";
        }
        cout<<endl;

        for(int baris = 0; baris < jumlahSimpul; baris++){
            cout<<dataSimpul[baris]<<" | ";
            for(int kolom = 0; kolom < jumlahSimpul; kolom++){
                cout<<jalurTerdekat[baris][kolom]<<" ";
            }
            cout<<endl;
        }
    }
}


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(){
    char keluar;
    cout<<"Silahkan masukkan jumlah kota (angka) : ";
    cin>>jumlahSimpul;
    buatMatriks();
    tampilMatriks();
    cout<<endl<<endl;
    cout<<"ini adalah algoritma flody-warshall" << endl;
    hitungJarakTerdekatwarshall();
    cout<<endl;

    do{
        cout<<"ini adalah algoritma djistra" << endl;
        cout<<"Silahkan masukkan simpul asal (0 - "<<jumlahSimpul-1<<") : ";
        cin>>simpulAsal;
        cout<<"Silahkan masukkan simpul tujuan (0 - "<<jumlahSimpul-1<<") : ";
        cin>>simpulTujuan;
        hitungJarakTerdekatdjistra();


        cout<<endl<<endl;
        cout<<"Keluar (y/t) ? : ";
        cin>>keluar;

        if(tolower(keluar) != 'y'){
            system("cls");
        }
    }while(tolower(keluar) != 'y');

    return 0;
}
}
```

Analisis program

Program di atas karana pengabung antara algoritma wasrhall dan algoritma djistra jadi ada dua fungsi di sini yaitu ‘hitungjarakterdekatwarshall” dan fungsi ‘hitungjarakterpendekdjistra’

Pada fungsi ‘hitungjarakterpdendekdjistra’ pertama ialah mengecek apakah matriks telah di buat ‘if(cekMatrik)’ lalu mengalokasikan array untuk statatus kunjungan simpul, untuk menerapkan algoritma dijkstar guna mencari jalur terpendek ‘while(simpulSaatIni !=

simpulTujuan)’ untuk mengecek simpul saaat ini dan simpul yang dituju program akan mengecek emua jalur dan menetutakan jalur terpendek atau jalur dengan nilai terkecil

jarak Baru = jarakSaatIni + dataBusur[simpulSaatIni][i]; setelah jalur terpendek ditemukan fungsi ‘cekmatriks’ akan menampilkan nama simpul dan bobot antar simpul terpendek mengunakan perulangan for dan menampilkannya pada fungsi ‘hitungjarakterdekat’

cout<<"Jarak terdekat dari "<<simpulAsal<<" ke "<<simpulTujuan<<" adalah "<<jarakDiketahui[simpulTujuan]<<endl;

didalam variebel ‘int main’ semua fungi dipanggil untuk diekseskusi dengan varibel ‘jumlahsimpul’ ini memasukan data yang di input user pada fungi fungsi di atas untuk di proses dan ditampilkan pada terminal menjadi output

- Algoritma flody-washall

sedangkan pada fpada ‘funsi hitung jarak terpendek warshall’ ini lah ynag membedakan algioritma wrshall ada algoritma ini umum nya menggunakan matriks adjacency (vector<vector<int>>) untuk menyimpan bobot antara simpul-simpul yang berarti Algoritma Warshall memanfaatkan matriks adjacency untuk merepresentasikan graf dan mencari jalur terdekat. yang mana disni mengalokasikan memori untuk matriks ‘jalurTerdekat’, yang merupakan salinan dari matriks ‘dataBusur’ lalu menyalin nilai-nilai dari ‘dataBusur’ ke ‘jalurTerdekat’.

- Int Main

Lalu pada fungsi int main () fungsi dari ‘hitungjarakterdekat’ akan di panggil untuk mengeksekusi data yang di masukan pengguna seperti jumlah dari node, nama node dan bobot jalur antar simpul, setelah semua nya di deklarasikan pengguna akan di mita untuk menentukan jalur terpendek dari simpul tertentu lalu program akan mengeksekusi nya lalu menampilkan hasil dari pencarian jalur tependek. Disini saya akan memasukan 6 indeks yang nama simpul nya adalah nama nama kota dan bobot jalur nya bisa di ibaratkan adlh jarak antar kota tersebut

```
Silahkan masukkan jumlah kota (angka) : 4
Silahkan masukkan nama simpul
Kota 1 : jakarta
Kota 2 : bogor
Kota 3 : depok
Kota 4 : bekasi
Silahkan masukkan bobot antar simpul
jakarta --> jakarta : 0
jakarta --> bogor : 5
jakarta --> depok : 99
jakarta --> bekasi : 99
bogor --> jakarta : 99
bogor --> bogor : 0
bogor --> depok : 5
bogor --> bekasi : 99
depok --> jakarta : 99
depok --> bogor : 99
depok --> depok : 0
depok --> bekasi : 5
bekasi --> jakarta : 5
bekasi --> bogor : 99
bekasi --> depok : 99
bekasi --> bekasi : 0
jakarta bogor depok bekasi
0 5 99 99
99 0 5 99
99 99 0 5
5 99 99 0


ini adalah algoritma flody-warshall
 --- jakarta bogor depok bekasi
jakarta | 0 5 10 15
bogor | 15 0 5 10
depok | 10 15 0 5
bekasi | 5 10 15 0

ini adalah algoritma djistra
Silahkan masukkan simpul asal (0 - 3) : 0
Silahkan masukkan simpul tujuan (0 - 3) : 3
Jarak terdekat dari 0 ke 3 adalah 15
```

KESIMPULAN

algoritma Dijkstra dan Warshall adalah dua pendekatan berbeda untuk menyelesaikan masalah jalur terpendek dalam graf, namun mereka memiliki perbedaan signifikan dalam penggunaan dan efisiensi. Algoritma Dijkstra digunakan untuk menemukan jalur terpendek dari satu simpul sumber ke simpul tujuan lainnya dalam graf dengan bobot positif, berfokus pada satu titik asal dan memperluas cakupan hingga mencapai tujuan. Algoritma ini efisien untuk graf dengan banyak simpul karena berfokus pada satu jalur pada satu waktu. Di sisi lain, algoritma Warshall (atau lebih tepatnya, algoritma Floyd-Warshall) digunakan untuk menemukan jalur terpendek antara semua pasangan simpul dalam graf, baik dengan bobot positif maupun negatif, dengan menciptakan matriks jarak yang diupdate secara iteratif

REFRENSI

1. MODUL 11 STRUKTUR DATA
2. MODUL 12 STRUKTURDATA
3. https://e-journal.stmik-tegal.ac.id/index.php/batirsi/article/view/42#:~:text=Algoritma%20Dijkstra%20merupakan%20salah%20satu,graf%20berbobot%20positif%20atau%20negatif.
4. https://jtiik.ub.ac.id/index.php/jtiik/article/view/2866
5. https://mti.binus.ac.id/2017/11/28/algoritma-dijkstra/
6. https://repository.unikom.ac.id/4819/