MODUL 10

GRAPH II

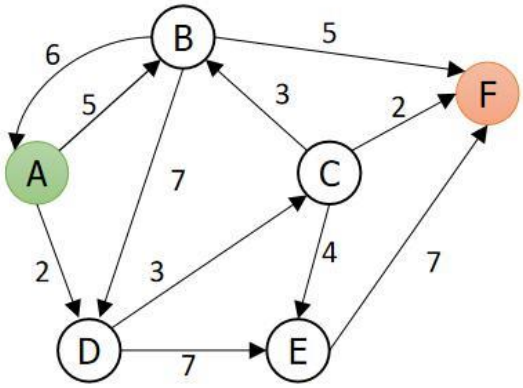
DASAR TEORI

A. Algoritma Dijkstra

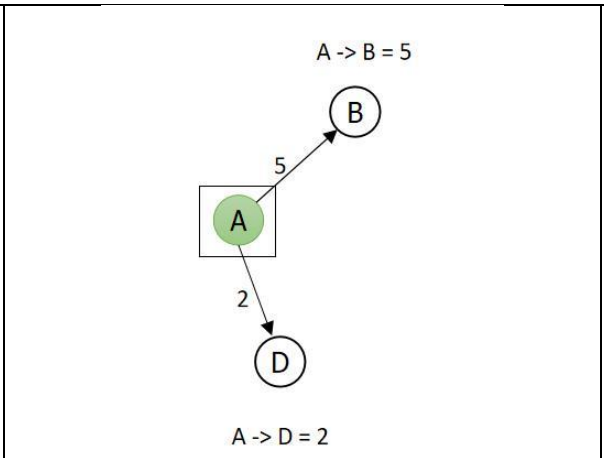
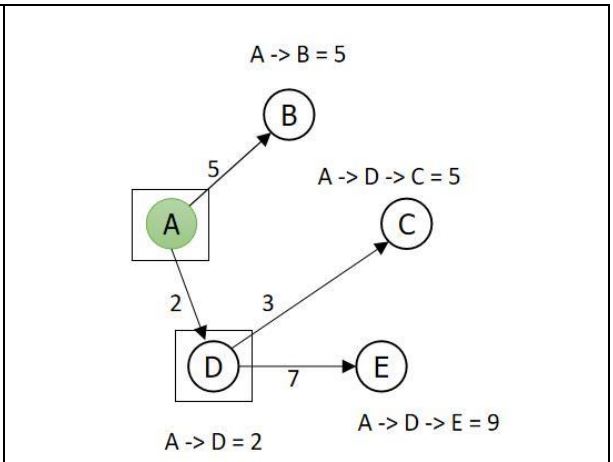
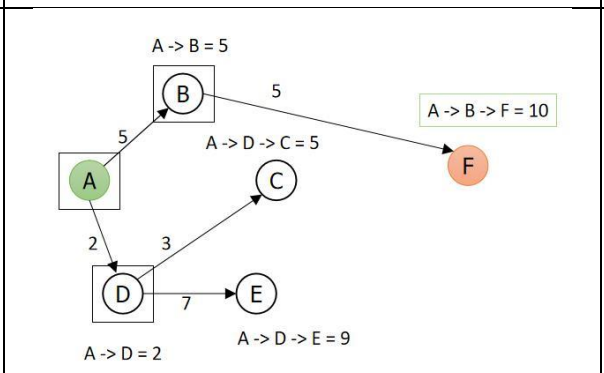
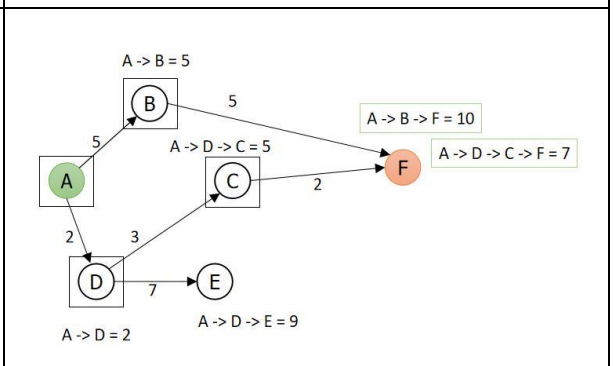
Algoritma Dijkstra merupakan algoritma dalam pencarian lintasan/rute terpendek. Algoritma ini ditemukan oleh **Edger W. Dijkstra** (1930 – 2002), merupakan tokoh yang paling berpengaruh dari generasi penemu ilmu komputer. Pada setiap langkah dalam algoritma Dijkstra, ambil sisi yang berbobot minimum yang menghubungkan sebuah simpul yang telah dikunjungi dengan sebuah simpul lain yang belum dikunjungi. Lintasan dari simpul asal ke simpul yang baru haruslah merupakan lintasan yang terpendek diantara semua lintasannya ke simpul-simpul yang belum dikunjungi.

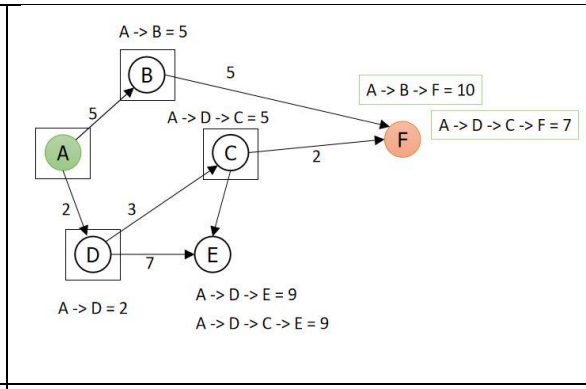
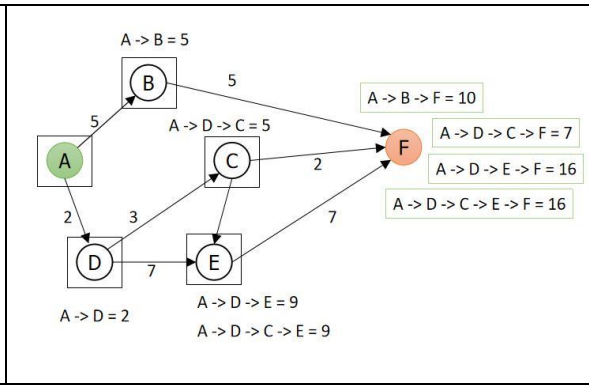
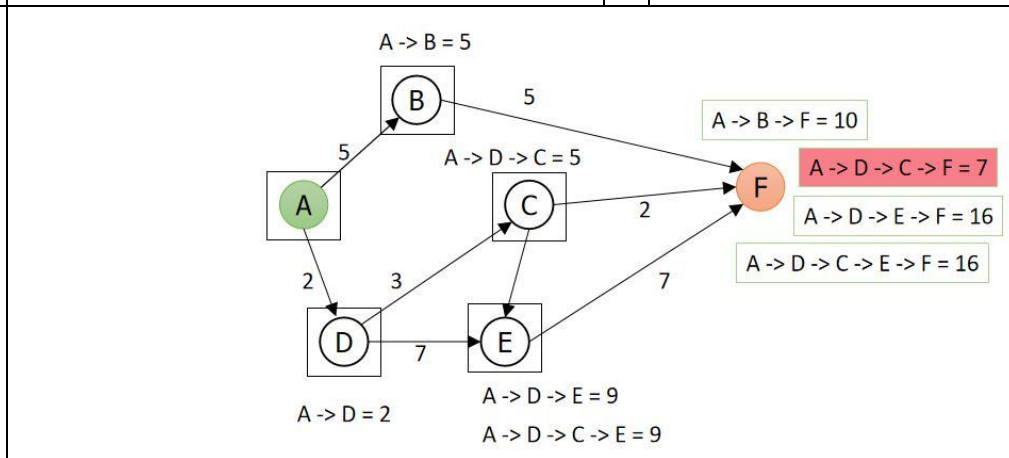
Berikut langkah-langkah algoritma Dijkstra:

1. Berikan nilai bobot untuk setiap titik ke titik lainnya, berikan nilai 0 pada node awal dan nilai tak hingga terhadap node lain (yang belum terisi).
2. Atur semua node dengan label “Belum dikunjungi” dan atur node awal sebagai “Node keberangkatan”.
3. Dari node keberangkatan, pertimbangkan node tetangga yang belum terjamah dan hitung jaraknya dari titik keberangkatan. Sebagai contoh, jika titik keberangkatan X ke Y memiliki bobot jarak 7 dan dari Y ke node Z berjarak 13, maka jarak ke Z melewati Y menjadi 7+13=20. Jika jarak ini lebih kecil dari jarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru.
4. Saat kita selesai mempertimbangkan setiap jarak terhadap node tetangga, tandai node yang telah terjamah sebagai “Node dikunjungi”. Node yang telah dikunjungi tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.
5. Set “node belum dikunjungi” dengan jarak terkecil (dari node asal) sebagai “node asal” selanjutnya, kemudian lanjutkan dengan mengulang langkah 3.



Gambar 1 Contoh Pencarian Lintasan Terpendek dari A ke F

1		2	
3		4	

5	
6	
7	

Tabel 1 Langkah-langkah algoritma Dijkstra (A ke F)

B. Algoritma Floyd – Warshall

Algoritma pencarian rute terpendek Flyod – Warshall melakukan perbandingan jarak/bobot semua simpul sehingga hasil dari perhitungan algoritma ini adalah lintasan terpendek untuk semua pasangan simpul yang terdapat pada graph. Kelebihan dari algoritma ini yakni dapat diterapkan meskipun nilai jarak/bobot suatu busur/edge negatif berbeda dengan algoritma Dijkstra yang hanya dapat digunakan pada graph yang bernilai positif untuk semua jarak/bobot busurnya.

Langkah awal dalam menerapkan algoritma ini untuk menemukan lintasan terpendek dari sebuah graph adalah dengan membuat representasi graph dalam bentuk matrik adjacency. Graph dalam bentuk matrik tersebut yang kemudian diproses. Algoritma Flyod – Warshall merupakan algoritma yang bersifat iteratif, jumlah proses

iterasi yang mungkin terjadi sebanyak jumlah simpul pangkat 3 (n^3), sehingga apabila terdapat 7 simpul maka akan terjadi perulangan sebanyak 343.

```
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];
            }
        }
    }
}
```

GUIDED

Program Pencarian Lintasan Terpendek Algoritma Dijkstra

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

using namespace std;

///PROGRAM PENCARIAN LINTASAN TERPENDEK (ALGORITMA DIJKSTRA)

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

//Buat Graph
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 << endl;

    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];
        }
    }
    cout << endl;

    cekMatriks = true;
}

//Hitung Jarak Terpendek
void jarakTerdekat()
{
    if (cekMatriks){
        jarakDiketahui = new int[jumlahSimpul];
        kunjungan = new int[jumlahSimpul];

        for (int i=0; i<jumlahSimpul; i++){
            jarakDiketahui[i] = 999; //Nilai 999 dianggap sebagai nilai
infinity (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 << " Lintasan terpendek dari simpul indeks ke-" << simpulAsal
<<  "  ke  simpul  indeks  ke-"  <<  simpulTujuan  <<  "  adalah  "  <<
jarakDiketahui[simpulTujuan] << endl;
        delete jarakDiketahui;
        delete kunjungan;
    }
}

//Tampil Data Graph
void tampilMatriks()
{
    if (cekMatriks){
        cout << left << setw(4) << " ";
        for (int i=0; i<jumlahSimpul; i++){
            cout  << left << setw(5) << dataSimpul[i] << " ";
        }
        cout << endl;

        for (int baris = 0; baris < jumlahSimpul; baris++){
            cout << " " << setw(5) << dataSimpul[baris] << left << setw(2)
<< " ";
            for (int kolom = 0; kolom < jumlahSimpul; kolom++){
                cout << left << setw(5) << dataBusur[baris][kolom] << "
";
            }
            cout << endl;
        }
        cout << endl;
    }
    else{
        cout << " Graph masih kosong!" << endl;
        cout << endl;
    }
}

int main()
{
    char keluar;

    cout << " Silahkan masukkan jumlah simpul: ";
    cin >> jumlahSimpul;

    buatMatriks();
    tampilMatriks();

    do{
```

```
        cout << " Silahkan masukkan indeks simpul asal [0 - " <<
jumlahSimpul-1 << "]: ";
        cin >> simpulAsal;
        cout << " Silahkan masukkan indeks simpul tujuan [0 - " <<
jumlahSimpul-1 << "]: ";
        cin >> simpulTujuan;

        jarakTerdekat();

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

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

    }while (tolower(keluar) != 'y');

    return 0;
}
```

Program Pencarian Lintasan Terpendek Algoritma Floyd – Warshall

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

using namespace std;

///PROGRAM PENCARIAN LINTASAN TERPENDEK (ALGORITMA FLOYD-WARSHALL)

//Deklarasi Global
int jumlahSimpul;
string *dataSimpul;
int **dataBusur;
int **jalurTerdekat;
bool cekMatriks = false;

//Buat Matriks Graph
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 memasukkan nama simpul" << endl;
    for (int i=0; i<jumlahSimpul; i++){
        cout << " Simpul " << i+1 << ": ";
        cin >> dataSimpul[i];
    }
    cout << endl;

    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];
        }
    }
    cout << endl;

    cekMatriks = true;
}

//Hitung Lintasan Terpendek
void jarakTerdekat()
{
    if (cekMatriks){
        //Membuat matriks yang sama dengan matriks 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 matriks dataBusur ke dalam matriks jalurTerdekat
        for (int baris = 0; baris < jumlahSimpul; baris++){
            for (int kolom = 0; kolom < jumlahSimpul; kolom++){
                jalurTerdekat[baris][kolom] = dataBusur[baris][kolom];
            }
        }
```

```
        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 << left << setw(4) << " ";
        for (int i = 0; i < jumlahSimpul; i++){
            cout  << left << setw(5) << dataSimpul[i] << " ";
        }
        cout << endl;

        for (int baris = 0; baris < jumlahSimpul; baris++){
            cout << " " << dataSimpul[baris] << left << setw(2) << " ";
            for (int kolom = 0; kolom < jumlahSimpul; kolom++){
                cout << left << setw(5) << jalurTerdekat[baris][kolom] <<
" ";
            }
            cout << endl;
        }
        cout << endl;
    }
}

//Tampil
void tampilMatriks()
{
    if (cekMatriks){
        cout << left << setw(4) << " ";
        for (int i=0; i<jumlahSimpul; i++){
            cout  << left << setw(5) << dataSimpul[i] << " ";
        }
        cout << endl;

        for (int baris = 0; baris < jumlahSimpul; baris++){
            cout << " " << dataSimpul[baris] << left << setw(2) << " ";
            for (int kolom = 0; kolom < jumlahSimpul; kolom++){
                cout << left << setw(5) << dataBusur[baris][kolom] << "
";
            }
```

```
            cout << endl;
        }
        cout << endl;
    }
    else{
        cout << " Matriks masih kosong!" << endl;
        cout << endl;
    }
}

int main()
{
    cout << " Silahkan masukkan jumlah simpul: ";
    cin >> jumlahSimpul;

    buatMatriks();
    tampilMatriks();
    jarakTerdekat();

    return 0;
}
```

TUGAS

Modifikasi program Guided I (Algoritma Dijkstra) dalam pencarian jalur terpendek agar menampilkan data nama simpulnya **[Bobot 60]**, dan buatlah menjadi program menu dengan menggabungkan algoritma Dijkstra dan Floyd – Warshall **[Bobot 40]**!

~ SELAMAT MENGERJAKAN 😊 ~