MODUL XII

GRAPH II

I. Tujuan

1. Mahasiswa mampu menjelaskan konsep dari Algoritma Djikstra
2. Mahasiswa mampu menjelaskan konsep dari Algoritma Flyod Warshall
3. Mahasiswa mampu membuat program dengan menerapkan algoritma Djikstra
4. Mahasiswa mampu membuat program dengan menerapkan algoritma Flyod Warshall

II. Dasar Teori

2.1 Algoritma Djikstra

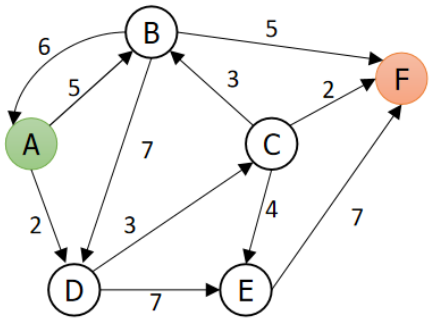
Pada setiap langkah, ambil sisi yang berbobot minimum yang menghubungkan sebuah simpul yang telah dikunjungi dengan sebuah simpul lain yang belum dikunjungi. Lintasan dari simpul asal ke simpul yang baru haruslah merupakan lintasan yang terpendek diantara semua lintasannya ke simpul-simpul yang belum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

Langkah-langkah algoritma dijkstra adalah sebagai berikut:

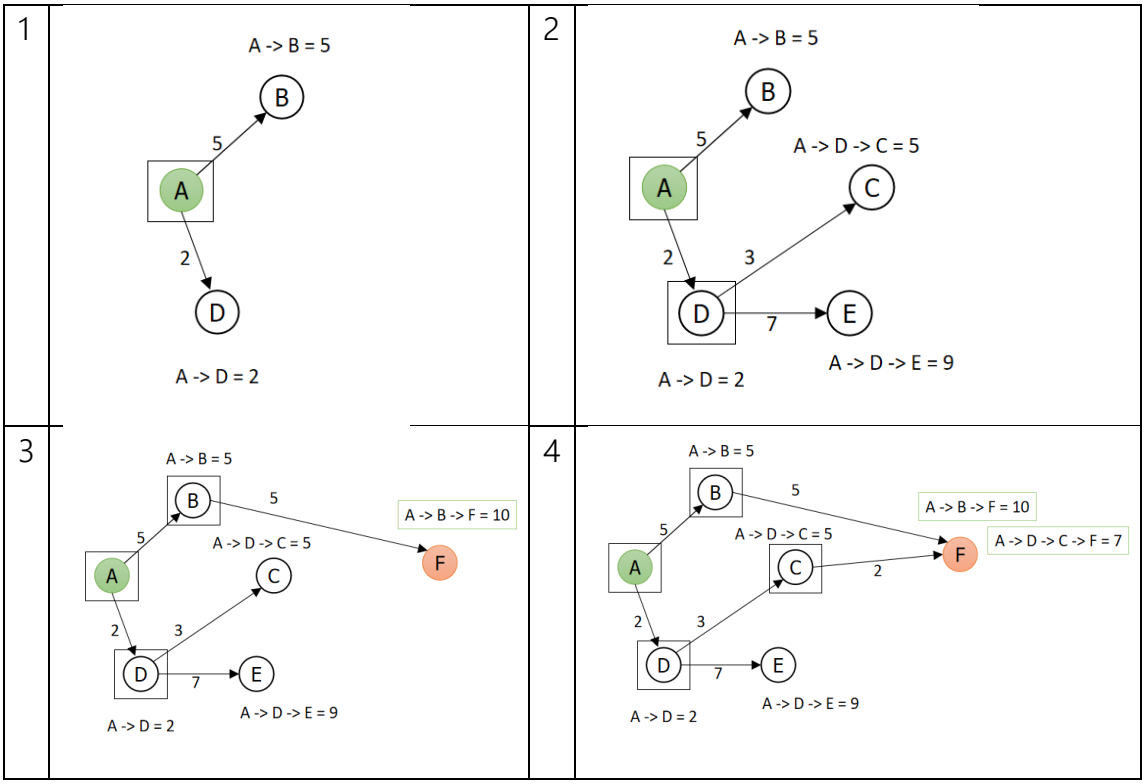
1. Berikan nilai bobot untuk setiap titik ke titik lainnya, berikan nilai 0 pada node awal dan nilai tak hingga terhadap node lain (yang belum terisi).
2. Atur semua node dengan label “Belum dikunjungi” dan atur node awal sebagai “Node keberangkatan”
3. Dari node keberangkatan, pertimbangkan node tetangga 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.

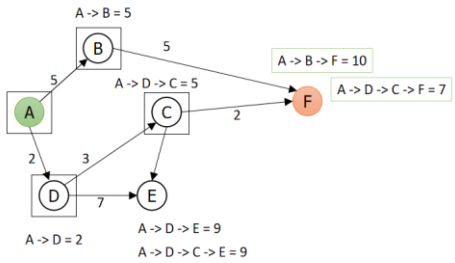
4. Saat kita selesai mempertimbangkan setiap jarak terhadap node tetangga, tandai node yang telah terjamah sebagai "Node dikunjungi". Node yang telah dikunjungi tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.
5. Set "node belum dikunjungi" dengan jarak terkecil (dari node asal) sebagai "node asal" selanjutnya, kemudian lanjutkan dengan mengulang langkah 3.



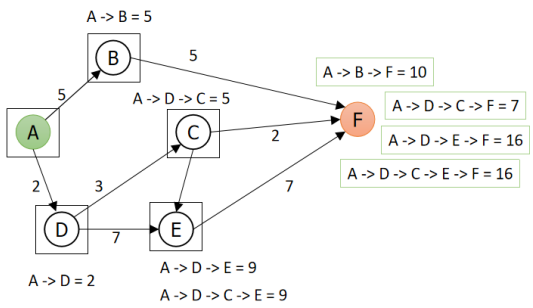
Gambar 01. Contoh Pencarian Lintasan Terpendek dari A ke F



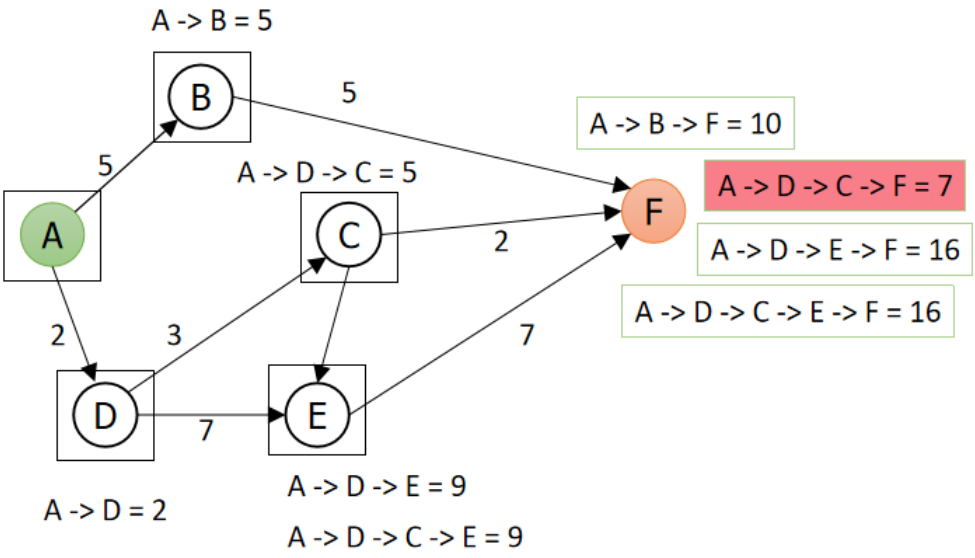
5



6



7



2.2 Algoritma Flyod Warshall

Algoritma pencarian 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 Djikstra 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];
            }
        }
    }
}
```

Keterangan :

k = Dimulai 0 -> jumlah simpul pada graph

baris = Dimulai 0 -> jumlah simpul pada graph

kolom = Dimulai 0 -> jumlah simpul pada graph

III. Guided

3.1 Guided I

Berikut adalah program pencarian lintasan terpendek dengan menerapkan algoritma Djikstra. Silahkan dijalankan, amati, dan lakukan analisis.

```
#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;
}
```

3.2 Guided II

Berikut adalah program pencarian lintasan terpendek dengan menerapkan algoritma Flyod Warshall. Silahkan dijalankan, amati, dan lakukan analisis..

```
#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;
}
```

IV. 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.

2. 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.

Catatan : Sertakan sintak program dan screenshoot hasil running program untuk jawaban Guided maupun Unguided.

V. Sumber

Sjukani, Moh. 2007. Struktur Data (Algoritma & Struktur Data 2) dengan C, C++. Jakarta : Penerbit Mitra Wacana Media.