Praktikum 14

Graph

(Algoritma Multipath)

A. TUJUAN PEMBELAJARAN

Setelah melakukan praktikum dalam bab ini, mahasiswa diharapkan mampu:

1. Memahami struktur data *graph.*
2. Mampu mengimplementasikan algoritma pencarian jalur terpendek

B. DASAR TEORI

B.1 Pengertian *Graph*

Graph adalah kumpulan *node* (simpul) di dalam bidang dua dimensi yang dihubungkan dengan sekumpulan garis (sisi). *Graph* dapat digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Representasi visual dari graph adalah dengan menyatakan objek sebagai *node*, bulatan atau titik (*Vertex*), sedangkan hubungan antara objek dinyatakan dengan garis (*Edge*).

$$G = (V, E) \quad (14.1)$$

dimana

G = Graph

V = Simpul atau Vertex, atau Node, atau Titik

E = Busur atau Edge, atau arc

Jalur pada graph dinotasikan sebagai

$$P = (V_0, V_1, \ldots, V_n) \quad (14.2)$$

dimana

P : jalur

V_i : titik jalur

n : jumlah titik jalur

Graph merupakan suatu cabang ilmu yang memiliki banyak terapan. Banyak sekali struktur yang bisa direpresentasikan dengan *graph*, dan banyak masalah yang bisa diselesaikan dengan bantuan *graph*. Seringkali *graph* digunakan untuk merepresentasikan suaru jaringan. Misalkan jaringan jalan raya dimodelkan *graph* dengan kota sebagai simpul (*vertex/node*) dan jalan yang menghubungkan setiap kotanya sebagai sisi (*edge*) yang bobotnya (*weight*) adalah panjang dari jalan tersebut.

Ada beberapa cara untuk menyimpan *graph* di dalam sistem komputer. Struktur data bergantung pada struktur *graph* dan algoritma yang digunakan untuk memanipulasi *graph*. Secara teori salah satu dari keduanya dapat dibedakan antara struktur *linked list* dan matriks (*array* dimensi 2), tetapi dalam penggunaannya struktur terbaik yang sering digunakan adalah kombinasi keduanya.

B.1.1 Istilah Pada *Graph*

Terdapat beberapa istilah yang berkaitan dengan *graph* yaitu:

a. *Vertex*

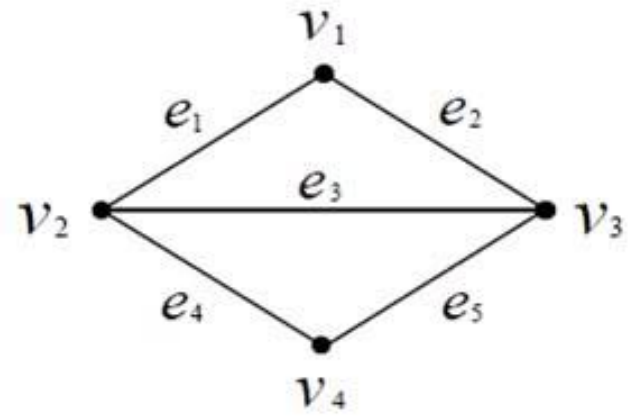
   Adalah himpunan node / titik pada sebuah *graph*.

b. *Edge*

   Adalah himpunan garis yang menghubungkan tiap *node* / *vertex*.

c. *Adjacent*

   Adalah dua buah titik dikatakan berdekatan (*adjacent*) jika dua buah titik tersebut terhubung dengan sebuah sisi. Adalah Sisi $e_3 = v_2v_3$ insident dengan titik v_2 dan titik v_3, tetapi sisi $e_3 = v_2v_3$ tidak insident dengan titik v_1 dan titik v_4. Titik v_1 adjacent dengan titik v_2 dan titik v_3, tetapi titik *v1* tidak adjacent dengan titik v_4.



Gambar 14.2 Adjacent pada Graph

d. Weight

Adalah Sebuah *graph G = (V, E)* disebut sebuah *graph* berbobot (*weight graph*), apabila terdapat sebuah fungsi bobot bernilai real *W* pada himpunan *E*,

$$W : E \rightarrow R \qquad (14.3)$$

nilai *W(e)* disebut bobot untuk sisi *e*, $\forall\ e \in$ E. Graf berbobot tersebut dinyatakan pula sebagai *G = (V, E, W).*

Graf berbobot *G = (V, E, W)* dapat menyatakan

- suatu sistem perhubungan udara, di mana
  - ✓ V = himpunan kota-kota
  - ✓ E = himpunan penerbangan langsung dari satu kota ke kota lain
  - ✓ W = fungsi bernilai real pada E yang menyatakan jarak atau ongkos atau waktu
- suatu sistem jaringan komputer, di mana
  - ✓ V = himpunan komputer
  - ✓ E = himpunan jalur komunikasi langsung antar dua komputer
  - ✓ W = fungsi bernilai real pada E yang menyatakan jarak atau ongkos atau waktu

e. Path

Adalah jalur dengan setiap *vertex* berbeda. Contoh, P = D5B4C2A Sebuah jalur (*W*) didefinisikan sebagai urutan (tidak nol) *vertex* dan *edge.* Diawali *origin vertex* (vertex awal) dan diakhiri *terminus vertex* (vertex akhir). Dan setiap 2 garis berurutan adalah *series*. Contoh, W = A1B3C4B1A2.

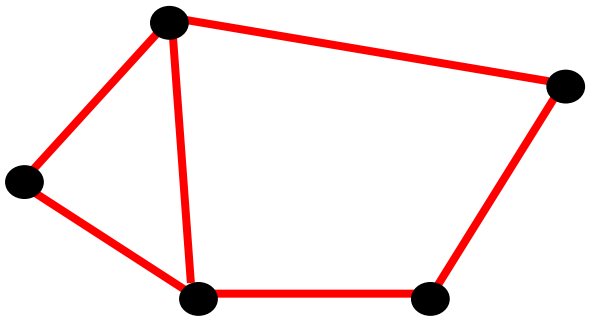
f. Cycle

Adalah Siklus (*Cycle*) atau Sirkuit (*Circuit)* yaitu lintasan yang berawal dan berakhir pada simpul yang sama .

B.1.2 Jenis-Jenis *Graph*

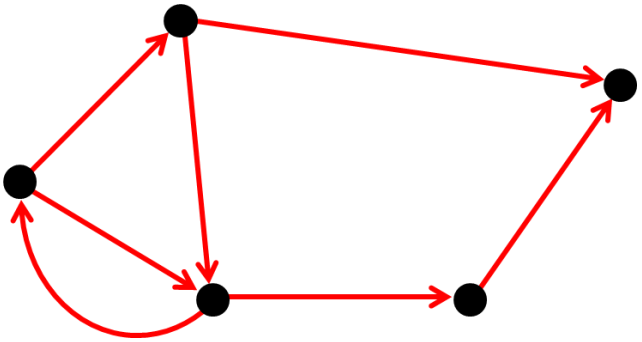
Terdapat beberapa jenis *graph* yaitu

- *Graph* tak berarah (*undirected graph* atau *non-directed graph*) dimana urutan simpul dalam sebuah busur tidak dipentingkan. Misal busur e_1 dapat disebut busur AB atau BA.



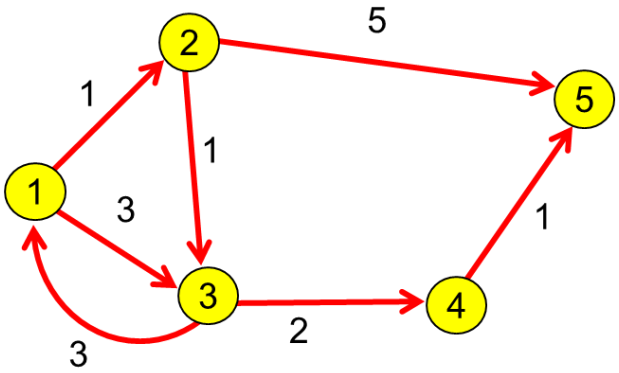
Gambar 14.2 *Graph* tak berarah

- *Graph* berarah (*directed graph*) dimana urutan simpul mempunyai arti. Misal busur AB adalah e_1 sedangkan busur BA adalah e_8.



Gambar 14.3 *Graph* berarah

- *Graph* Berbobot (*Weighted Graph*)
  - ✓ Jika setiap busur mempunyai nilai yang menyatakan hubungan antara 2 buah simpul, maka busur tersebut dinyatakan memiliki bobot.
  - ✓ Bobot sebuah busur dapat menyatakan panjang sebuah jalan dari 2 buah titik, jumlah rata-rata kendaraan perhari yang melalui sebuah jalan, dll.



Gambar 14.4 *Graph* berbobot

B.2 Representasi Graph dengan Matriks (Array Dimensi 2)

Dalam pemrograman, agar data yang ada dalam graph dapat diolah, maka graph harus dinyatakan dalam suatu struktur data yang dapat mewakili graph tersebut. Dalam hal ini *graph* perlu direpresentasikan kedalam bentuk array dan dimensi yang sering disebut matrix.

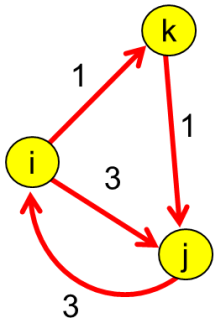
Lintasan terpendek merupakan salah satu dari masalah yang dapat diselesaikan dengan *graph*. Jika diberikan sebuah *graph* berbobot, masalah lintasan terpendek adalah bagaimana kita mencari sebuah jalur pada *graph* yang meminimalkan jumlah bobot sisi pembentuk jalur tersebut. Terdapat beberapa macam persoalan lintasan terpendek antara lain:

- Lintasan terpendek antara dua buah simpul tertentu (*a pair shortets path*).
- Lintasan terpendek antara semua pasangan simpul (*all pairs shortest path*).
- Lintasan terpendek dari simpul tertentu ke semua simpul yang lain (*single-source shoertest path*).
- Lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu (*intermediate shortest path*).

B.3 Algoritma Warshall

Algoritma Floyd-Warshall menghitung jarak terpendek (*shortest path*) untuk semua pasangan titik pada sebuah *graph*, dan melakukannya dalam waktu berorde kubik. Algoritma warshall digunakan untuk menyelesaikan permasalahan jalur terpendek *multi path.*

Algoritma Floyd-Warshall memiliki input *graph* berarah dan berbobot (V,E), yang berupa daftar titik (*node/vertex* V) dan daftar sisi (e*dge* E). Jumlah bobot sisi-sisi pada sebuah jalur adalah bobot jalur tersebut. Sisi pada E diperbolehkan memiliki bobot negatif, akan tetapi tidak diperbolehkan bagi *graph* ini untuk memiliki siklus dengan bobot negatif. Algoritma ini menghitung bobot terkecil dari semua jalur yang menghubungkan sebuah pasangan titik, dan melakukannya sekaligus untuk semua pasangan titik.



Gambar 14.5 Ilustrasi *graph*

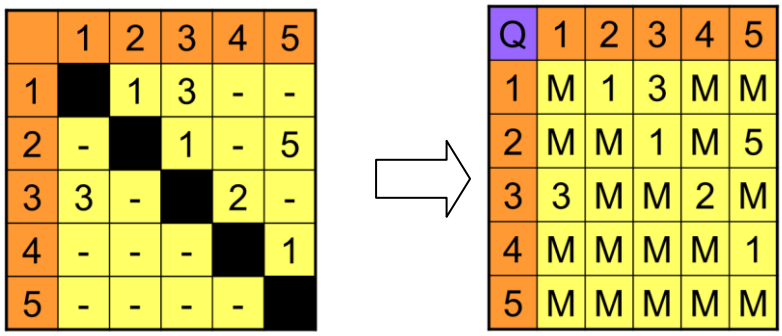
Sebagai ilustrasi pada Gambar 14.5, algoritma melakukan pengecekan apakah beban langsung Q(i, j) memang lebih kecil daripada beban melalui titik perantara Q(i,k)+Q(k,j)

if ((Q(i,k)+Q(k,j))< Q(i, j))

Q(i, j) ⇐ Q(i,k)+Q(k,j)

Misalkan pada contoh kasus Gambar 14.4 di atas, langkah-langkah penyelesaian dengan algoritma Warshall sebagai berikut:

1. Representasi matriks beban di bawah ini menjadi array dua dimensi Q.

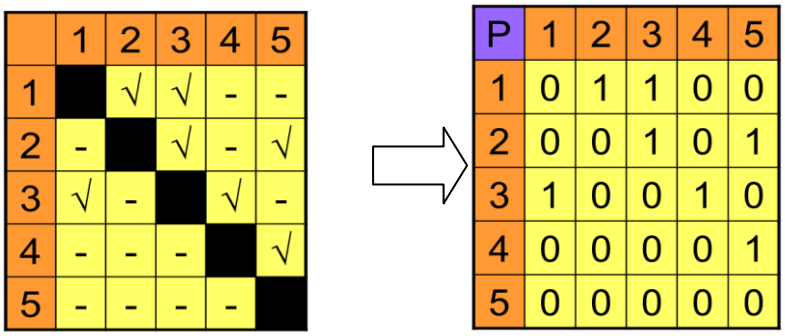


	1	2	3	4	5
1		1	3	-	-
2	-		1	-	5
3	3	-		2	-
4	-	-	-		1
5	-	-	-	-	

Q	1	2	3	4	5
1	M	1	3	M	M
2	M	M	1	M	5
3	3	M	M	2	M
4	M	M	M	M	1
5	M	M	M	M	M

Dimana M adalah big integer.

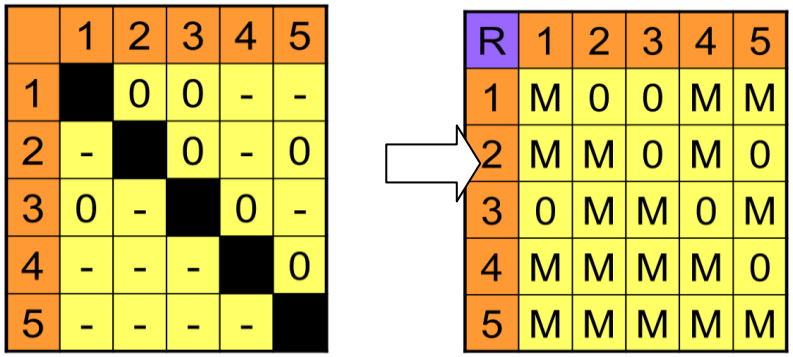
2. Representasi matriks jalur di bawah ini menjadi array dua dimensi P



	1	2	3	4	5
1		√	√	-	-
2	-		√	-	√
3	√	-		√	-
4	-	-	-		√
5	-	-	-	-	

P	1	2	3	4	5
1	0	1	1	0	0
2	0	0	1	0	1
3	1	0	0	1	0
4	0	0	0	0	1
5	0	0	0	0	0

3. Representasi matriks rute di bawah ini menjadi array dua dimensi



	1	2	3	4	5
1		0	0	-	-
2	-		0	-	0
3	0	-		0	-
4	-	-	-		0
5	-	-	-	-	

R	1	2	3	4	5
1	M	0	0	M	M
2	M	M	0	M	0
3	0	M	M	0	M
4	M	M	M	M	0
5	M	M	M	M	M

Dimana M adalah big integer.

4. Melakukan pengecekan beban langsung Q(1,3) = 3 dengan beban tak langsung

   Q(1,1) + Q(1,3) = M+3

   Q(1,2) + Q(2,3) = 2

   Q(1,3) + Q(3,3) = 3+M

   Q(1,4) + Q(4,3) = M+M

   Q(1,5) + Q(5,3) = M+M

Sehingga Beban terkecil adalah Q(1,3) = 2

Algoritma warshall untuk beban adalah sebagai berikut :

```
for k = 1 to n
  for i = 1 to n
    for j = 1 to n
      if ((Q(i,k) + Q(k,j)) < Q(i,j))
         Q(i,j) ⇐ (Q(i,k)+Q(k,j)
```

Algortitma warshall untuk jalur adalah sebagai berikut :

```
for k = 1 to n
  for i = 1 to n
    for j = 1 to n
      P(i,j) ⇐  P(i,j) OR (P(i,k) AND P(k,j))
```

Algoritma warshall untuk rute adalah sebagai berikut :

```
for k = 1 to n
  for i = 1 to n
    for j = 1 to n
        if ((Q(i,k) + Q(k,j)) < Q(i,j))
          if (R(k,j) = 0)
            R(i,j) ⇐ k
          else
            R(i,j)=R(k,j)
```

C. TUGAS PENDAHULUAN

Jawablah pertanyaan berikut ini :

1. Jelaskan representasi graph dengan matriks untuk Beban, Jalur dan Rute
2. Tuliskan algoritma Warshall

D. PERCOBAAN

Percobaan 1 : Mendeklarasikan matriks Beban, Jalur dan Rute

```
#include<stdio.h>

#define N 5
#define M 1000


void Tampil(int data[N][N], char *judul)
{
   printf("%s = \n",judul);
   for(int i=0; i<N; i++) {
         for(int j=0; j<N; j++)
            if(data[i][j] >= M)
```

```
                    printf("M ");
               else
                    printf("%d ", data[i][j]);
          printf("\n");
    }
}

void main()
{
    int Beban[N][N] = {M,1,3,M,M,
                       M,M,1,M,5,
                       3,M,M,2,M,
                       M,M,M,M,1,
                       M,M,M,M,M};

     int Jalur[N][N] = {0,1,1,0,0,
                        0,0,1,0,1,
                        1,0,0,1,0,
                        0,0,0,0,1,
                        0,0,0,0,0};

     int Rute[N][N] = {M,0,0,M,M,
                       M,M,0,M,0,
                       0,M,M,0,M,
                       M,M,M,M,0,
                       M,M,M,M,M};

    Tampil(Beban, "Beban");
    Tampil(Jalur, "Jalur");
    Tampil(Rute, "Rute");
}
```

Percobaan 2 : Algoritma Warshall untuk pencarian jalur terpendek multipath

```
#include<stdio.h>

#define N 5
#define M 1000


void Tampil(int data[N][N], char *judul)
{
    printf("%s = \n",judul);
    for(int i=0; i<N; i++) {
          for(int j=0; j<N; j++)
             if(data[i][j] >= M)
                    printf("M ");
             else
                    printf("%d ", data[i][j]);
          printf("\n");
    }
}

void Warshall(int Q[N][N], int P[N][N], int R[N][N])
{
    for(int k=0; k<N; k++)
          for (int i=0; i<N; i++)
                for (int j=0; j<N; j++){
                      P[i][j] = P[i][j] | (P[i][k] & P[k][j]);
```

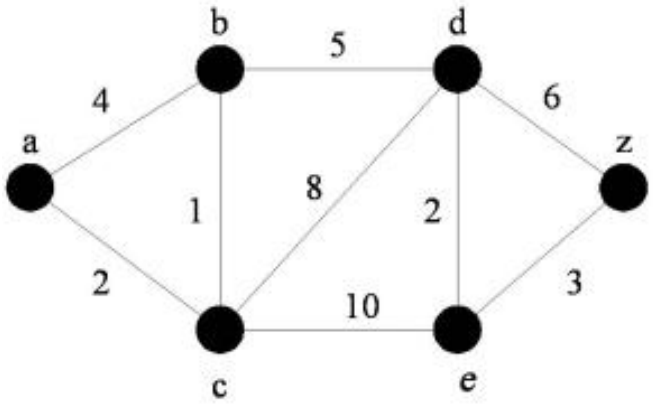
```
                        if ((Q[i][k] + Q[k][j]) < Q[i][j]) {
                         Q[i][j] = Q[i][k] + Q[k][j];
                           if (R[k][j] == 0)
                              R[i][j] = k+1;
                           else
                              R[i][j] = R[k][j];
                        }
                  }
}

void main()
{
   int Beban[N][N] = {M,1,3,M,M,
                      M,M,1,M,5,
                      3,M,M,2,M,
                      M,M,M,M,1,
                      M,M,M,M,M};

    int Jalur[N][N] = {0,1,1,0,0,
                       0,0,1,0,1,
                       1,0,0,1,0,
                       0,0,0,0,1,
                       0,0,0,0,0};

    int Rute[N][N] = {M,0,0,M,M,
                      M,M,0,M,0,
                      0,M,M,0,M,
                      M,M,M,M,0,
                      M,M,M,M,M};

   Tampil(Beban, "Beban");
   Tampil(Jalur, "Jalur");
   Tampil(Rute, "Rute");
   Warshall(Beban, Jalur, Rute);
   printf("Matriks setelah Algoritma Warshall : \n");
   Tampil(Beban, "Beban");
   Tampil(Jalur, "Jalur");
   Tampil(Rute, "Rute");
}
```

E. LATIHAN

1. Dari percobaan 2 diatas, tambahkan fungsi untuk menampilkan rute dari satu titik ke titik berikutnya berdasarkan matriks rute. Input berupa titik awal dan titik akhir. Gunakan struktur data Stack untuk mencari rute seperti langkah-langkah di bawah ini.
   1. Rute 1-5?
   2. Ambil nilai di baris 1, kolom 5 = 4 $\rightarrow$ push
   3. Ambil nilai di baris 1, kolom 4 = 3 $\rightarrow$ push
   4. Ambil nilai di baris 1, kolom 3 = 2 $\rightarrow$ push
   5. Ambil nilai di baris 1, kolom 2 = 0 (stop) $\rightarrow$ pop sampai stack kosong
   6. Sehingga rutenya menjadi 1-2-3-4-5 dengan beban minimal 5.

Beban minimal diperoleh dari matriks beban dengan nilai baris, kolom sama dengan 1,5. Lakukan hal yang sama untuk input titik awal dan titik akhir yang lain.

2. Berdasarkan *graph* di bawah ini, representasikan matriks, gunakan algoritma warshall untuk mencari rute terpendek dan rute seperti pada Latihan 1.



3. Rancanglah sendiri sebuah graph dan lakukan hal yang sama dengan Latihan 2.

F. LAPORAN RESMI

1. Kerjakan hasil percobaan(D) dan latihan(E) di atas dan tambahkan analisa.
2. Tuliskan kesimpulan dari percobaan dan latihan yang telah anda lakukan.