TUGAS VII

JARINGAN KOMPUTER

DISUSUN OLEH:

NAMA : ARUM CANTIKA PUTRI

NIM : 09011181419022

KELAS : SK 5 A

FAKULTAS ILMU KOMPUTER

JURUSAN SISTEM KOMPUTER

UNIVERSITAS SRIWIJAYA

2016

CARA ROUTER MENENTUKAN RUTE TERPENDEK DENGAN ALGORITMA

ALGORITMA DIJKSTRA

Dijkstra, adalah algoritma greedy yang menyelesaikan permasalahan mencari jarak terpendek dari suatu graf berarah dengan bobot sisi yang tidak negatif. Contoh analogi: jika simpul pada graf melambangkan suatu entitas komputer dalam jaringan dan bobot sisi melambangkan besaran beban (metric cost) antarentitas, algoritma Dijkstra dapat dipakai untuk mencari rute terbaik antara dua komputer dalam jaringan komputer.

Pseudukode

```
function Dijkstra (input M : matriks, a : simpul
   awal) → tabel
{ Mencari lintasan terpendek dari impul awal a
   ke semua simpul alinnya
   Masukan: matriks ketetanggaan (M) dari graf
   berbobot G dan simpul awal a
   Keluaran: tabel D yang berisi panjang
   lintasan terpendek dari a ke semua simpul

   lainnya
}
Deklarasi
   D, S : tabel
   i : integer
Algoritma:
   {Langkah 0 inisialisasi}
   for i ← 1 to n do
      S[i] ← 0
      D[i] ← m[a,i]
   endfor

   {Langkah 1}
   S[a] ← 1   {karena simpul a adalah simpul
               asal lintasan terpendek, jadi
               simpul a sudah pasti terpilih
               dalam lintasan terpendek}
   D[a] ← ∞   {tidak ada lintasan terpendek
               dari simpul a ke a}

   {Langkah 2, 3, ..., n-1}
   for i ← 2 to n-1 do
      Cari j sedemikian hingga S[j] = 0 dan D[j]
         = Minimum{D[1], D[2], ..., D[n]}
      S[j] ← 1 {Simpul j sudah terpilih ke dalam
                lintasan terpendek}
      Hitung D[i] yang baru dari a ke simpul i ∉
         S dengan cara sebagai berikut:
         D[i] ← Minimum{D[i], (D[j] + M[j,i])}
   endfor
   return D
```

Misalkan sebuah graf berbobot dengan n buah simpul dinyatakan degan matriks ketetanggan M = [mij], yang dalam hal ini,

mij = bobot sisi (i, j)

mii = 0

mij = ∞, jika tidak ada sisi dari simpul i ke simpul j

Selain matriks M, kita juga menggunakan tabel S = [si] yang dalam hal ini,

si = 1, jika simpul i termasuk ke dalam lintasan terpendek

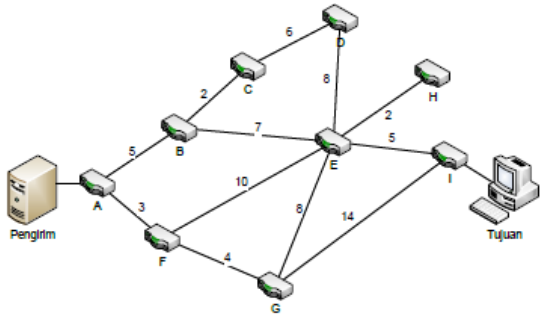
si = 0, jika simpul i tidak termasuk ke dalam lintasan terpendek

dan keluarannya adalah sebuah tabel D = [di], yang dalam hal ini,

di = panjang lintasan dari simpul awal ke simpul i

Pada akhir algoritma, di akann berisi panjang lintasan terpendek dari simpul asal ke simpul i.

Misalnya kita mempinyai graf seperti ini:



Pada graf tersebut, proses routing dilakukan dari simpul A menuju simpul I. Kemudian graf tersebut kita representasi dengan menggunakan matriks ketetanggaan sehingga menjadi:

Tabel 1. Matriks Ketetanggaan dari Graf Masalah

Simpul	A	B	C	D	E	F	G	H	I
A	∞	5	∞	∞	∞	3	∞	∞	∞
B	5	∞	2	∞	7	∞	∞	∞	∞
C	∞	2	∞	6	∞	∞	∞	∞	∞
D	∞	∞	6	∞	8	∞	∞	∞	∞
E	∞	7	∞	8	∞	10	8	2	5
F	3	∞	∞	∞	10	∞	4	∞	∞
G	∞	∞	∞	∞	8	4	∞	∞	14
H	∞	∞	∞	∞	2	∞	∞	∞	∞
I	∞	∞	∞	∞	5	∞	14	∞	∞

Dengan menggunakan pseudo-code algoritma dijkstra pada bab dasar teori, kita dapat memecahkan permasalahan ini dengan langkah-langkah sebagai berikut:

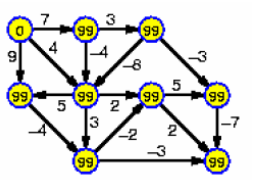
Tabel 2. Langkah Pencarian Jalur Terpendek

Langkah	Simpul yang dipilih	Lintasan	S								D								
			A	B	C	D	E	F	G	H	I	A	B	C	D	E	F	G	H	I
0	-	-	0	0	0	0	0	0	0	0	0	0	5 (A, B)	∞	∞	∞	3 (A, F)	∞	∞	∞
1	A	A	1	0	0	0	0	0	0	0	0	∞	5 (A, B)	∞	∞	∞	3 (A, F)	∞	∞	∞
2	F	AF	1	0	0	0	0	1	0	0	0	∞	5 (A, B)	∞	∞	13 (A,F ,E)	3 (A, F)	7 (A,F ,G)	∞	∞
3	G	AFG	1	0	0	0	0	1	1	0	0	∞	5 (A, B)	∞	∞	12 (A,F ,E)	3 (A, F)	7 (A,F ,G)	∞	21 (A,F, G,I)
4	E	AFE	1	0	0	0	1	1	1	0	0	∞	5 (A, B)	∞	21 (A,F, E,D)	13 (A,F ,E)	3 (A, F)	7 (A,F ,G)	15 (A,F, E,H)	18 (A,F, E,I)
5	B	AB	1	1	0	0	1	1	1	0	0	∞	5 (A, B)	7 (A,B ,C)	21 (A,F, E,D)	12 (A, B,E)	3 (A, F)	7 (A,F ,G)	15 (A,F, E,H)	18 (A,F, E,I)
6	C	ABC	1	1	1	0	1	1	1	0	0	∞	5 (A, B)	7 (A,B ,C)	13 (A,B, C,D)	12 (A, B,E)	3 (A, F)	7 (A,F ,G)	15 (A,F, E,H)	18 (A,F, E,I)
7	D	ABC D	1	1	1	1	1	1	1	0	0	∞	5 (A, B)	7 (A,B ,C)	13 (A,B, C,D)	12 (A, B,E)	3 (A, F)	7 (A,F ,G)	15 (A,F, E,H)	18 (A,F, E,I)
8	E	ABE	1	1	1	1	1	1	1	0	0	∞	5 (A, B)	7 (A,B ,C)	13 (A,B, C,D)	12 (A, B,E)	3 (A, F)	7 (A,F ,G)	14 (A,B, E,H)	17 (A,B, E,I)
8	H	ABE H	1	1	1	1	1	1	1	1	0	∞	5 (A, B)	7 (A,B ,C)	13 (A,B, C,D)	12 (A, B,E)	3 (A, F)	7 (A,F ,G)	14 (A,B, E,H)	17 (A,B, E,I)
9	I	ABE I	1	1	1	1	1	1	1	1	0	∞	5 (A, B)	7 (A,B ,C)	13 (A,B, C,D)	12 (A, B,E)	3 (A, F)	7 (A,F ,G)	14 (A,B, E,H)	17 (A,B, E,I)

ALGORITMA BELLMAN-FORD (NEGATIVE WEIGHTED PROBLEM)

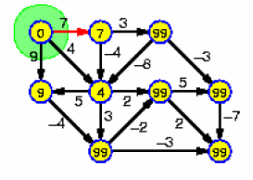
Algoritma Bellman-Ford menghitung jarak terpendek (dari satu sumber) pada sebuah digraph berbobot. Maksudnya dari satu sumber ialah bahwa ia menghitung semua jarak terpendek yang berawal dari satu titik node. Maka Algoritma Bellman-Ford hanya digunakan jika ada sisi berbobot negatif.

Skema Pencarian Lintasan Terpendek dengan algoritma Bellman Ford:

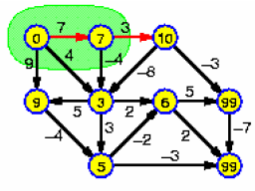


Gambar 1 Contoh graf berbobot negatif

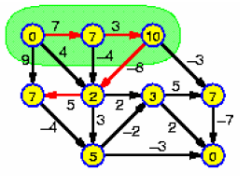
Bellman-Ford untuk penyelesaian contoh graf pada gambar 1



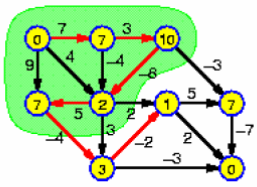
Gambar 2 Tahap pertama Algoritma Bellman-Ford untuk penyelesaian contoh graf pada gambar 1



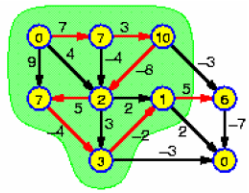
Gambar 3 Tahap kedua Algoritma



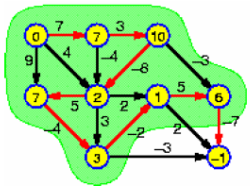
Gambar 4 Tahap ketiga Algoritma Bellman-Ford untuk penyelesaian contoh graf pada gambar 1



Gambar 5 Tahap keempat Algoritma Bellman-Ford untuk penyelesaian contoh graf pada gambar 1



Gambar 6 Tahap Kelima Algoritma Bellman-Ford untuk penyelesaian contoh graf pada gambar 1



Gambar 7 Lintasan terpendek untuk penyelesaian contoh graf pada gambar 1 sebesar -1

Algoritma

```
// Definisi tipe data dalam graf
record titik {
        list sisi2
        real jarak
```

```
        titik sebelum
}
record sisi {
        titik dari
        titik ke
        real bobot
}
function BellmanFord(list semuatitik, list
semuasisi, titik dari)
        // Argumennya ialah graf, dengan bentuk daftar titik
        // and sisi. Algoritma ini mengubah titik-titik dalam
        // semuatitik sehingga atribut jarak dan sebelum
        // menyimpan jarak terpendek.
        // Persiapan
        for each titik v in semuatitik:
                if v is dari then v.jarak = 0
                else v.jarak := tak-hingga
                v.sebelum := null
        // Perulangan relaksasi sisi
for i from 1 to size(semuatitik):
        for each sisi uv in semuasisi:
                u := uv.dari
                v := uv.ke //
uv adalah sisi dari u ke v
                if v.jarak > u.jarak + uv.bobot
                                v.jarak := u.jarak + uv.bobot
                                v.sebelum := u
        // Cari sirkuit berbobot(jarak) negative
        for each sisi uv in semuasisi:
                u := uv.dari
                v := uv.ke
                if v.jarak > u.jarak + uv.bobot
                        error "Graph
mengandung siklus berbobot total negatif"
```

Kompleksitas waktu algoritma (Running time)

Algoritma Bellman-Ford menggunakan waktu sebesar O(V.E), di mana V dan E adalah banyaknya sisi dan titik. Inisialisasi:

O(v) Loop utama: O(v.e)

Pengecekan loop negative: O(e)

Total: O(v.e)