Contoh Penerapan Program Dinamis dalam Algoritma Floyd-Warshall

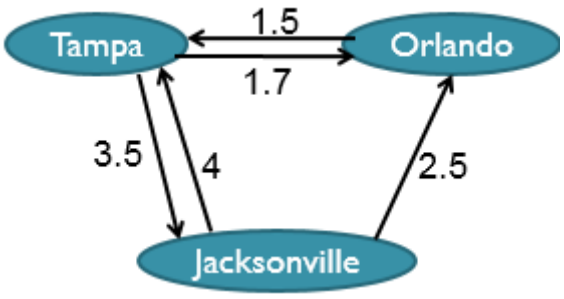
Letivany Aldina/13514067
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514067@std.stei.itb.ac.id

***Abstract*—Algoritma Floyd-Warshall merupakan salah satu jenis algoritma dalam program dinamis yang menyelesaikan permasalahan pencarian jalur terpendek. Prinsip penyelesaian masalah dalam algoritma ini sama dengan algoritma pencarian jalur terpendek lainnya dalam program dinamis, namun algoritma ini menggunakan graf berbobot berarah dengan bobot sisi positif atau negatif, tanpa memiliki siklus negatif. Dalam makalah ini, akan dibahas bagaimana detail langkah strategi yang digunakan dalam pencarian jalur terpendek menggunakan algoritma Floyd-Warshall ini.**

Kata kunci — Algoritma Floyd-Warshall, Matriks ketetanggaan, Pencarian jalur terpendek, Program dinamis.

I. PENDAHULUAN

Algoritma Floyd-Warshall merupakan algoritma pencarian jalur terpendek menggunakan graf berbobot berarah dengan bobot sisi positif atau negatif, namun tidak memiliki siklus dengan bobot negatif. Algoritma ini menghitung jarak terpendek dari seluruh pasangan simpul dalam graf berbobot berarah. Salah satu contoh penerapan algoritma Floyd-Warshall ini adalah dalam peta jalan.



Sumber: www.cs.ucf.edu/~sarahb/.../DynProg_FloydWarshall.ppt

Algoritma ini mengembalikan bobot terkecil untuk setiap pasangan simpul. Pada dasarnya, algoritma ini tidak mengembalikan urutan jalur yang ditempuh untuk mendapatkan bobot terkecil dari satu simpul ke simpul lainnya, namun dengan modifikasi tertentu dapat didapatkan urutan jalur yang ditempuh untuk mendapatkan lintasan terpendek dari pasangan simpul yang dibutuhkan atau untuk setiap pasangan simpul.

Penyelesaian masalah dalam algoritma Floyd-Warshall menggunakan matriks ketetanggaan antar simpul. Algoritma ini juga menggunakan simpul-simpul *intermediate* untuk memproses matriks ketetanggaan sehingga didapatkan seluruh jarak terpendek untuk setiap pasangan simpul. Apabila terdapat n simpul, maka matriks tersebut diproses dari $k = 1$, $k = 2$, sampai $k = n$. Kemudian untuk setiap matriks, diproses kembali berdasarkan simpul yang dijadikan simpul *intermediate*.

Pemrosesan satu matriks menggunakan rumus sebagai berikut:

$$A^k[i,j] = \min \{A^{k-1}[i,j],\ A^{k-1}[i,k] + A^{k-1}[k,j]\}$$

A = matriks ketetanggaan berbobot
i = baris matriks
j = kolom matriks
k = simpul yang menjadi simpul *intermediate*
min = fungsi yang meminimumkan bobot sisi

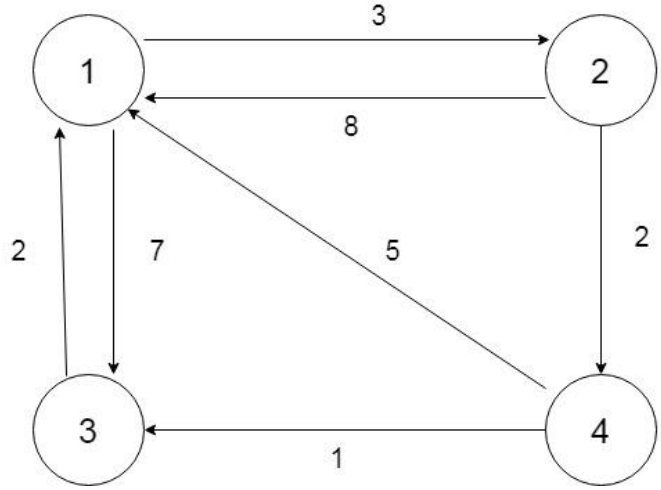
Secara umum, langkah-langkah dalam pemecahan masalah pencarian jalur terpendek menggunakan algoritma Floyd-Warshall adalah sebagai berikut:

1. Misalkan terdapat graf berbobot berarah $G = (V, E)$, dimana V adalah simpul dan E adalah sisi yang bersisian dengan simpul tersebut.
2. $|V| = n$. Dimana n merupakan jumlah simpul yang ada di dalam graf tersebut.
3. Membuat matriks ketetanggaan dari persoalan graf G.
4. Misalkan simpul *intermediate* adalah k, maka dimulai dari $k = 1$, $k = 2$, sampai $k = n$, matriks tersebut diproses menghasilkan elemen-elemen baru.
5. Untuk setiap proses di dalam sebuah matriks, seluruh elemen matriks pada baris k dan kolom k mempunyai nilai yang sama dengan matriks sebelumnya. Pada pemrosesan pertama, elemen matriks yang digunakan adalah elemen matriks saat matriks belum diproses.
6. Untuk setiap proses di dalam sebuah matriks, elemen diagonal matriks juga tetap memiliki nilai yang sama, yakni nol, karena tidak terdapat sisi dari dan ke simpul yang sama.
7. Elemen yang masih belum diproses atau masih kosong, diproses menggunakan rumus algoritma Foyd-Warshall.
8. Setelah seluruh matriks diproses, elemen-elemen pada

proses matriks terakhir menunjukkan bobot terkecil sebagai jalur terpendek yang dapat ditempuh antar pasangan simpul.

II. CONTOH PENYELESAIAN MASALAH

Misalkan terdapat graf bobot berarah sebagai berikut:



Dari graf tersebut, dapat dibuat matriks ketetanggaan A^0 yang menggambarkan bobot sisi dari setiap antar simpul tersebut, yakni sebagai berikut:

A^0		Kolom			
		1	**2**	**3**	**4**
Baris	**1**	0	3	∞	7
	2	8	0	2	∞
	3	5	∞	0	1
	4	2	∞	∞	0

Proses pertama yang dilakukan dari matriks A^0 tersebut adalah dimulai dari $k = 1$. Pada proses ini, dibuat matriks baru A^1, dengan simpul *intermediate* adalah simpul 1. Kemudian, seluruh elemen pada baris 1 dan kolom 1 matriks A^1 adalah elemen yang sama dengan elemen pada baris 1 dan kolom di matriks A^0. Selain itu, elemen diagonal yang semuanya juga nol juga memiliki nilai yang sama karena tidak ada jalur dari dan ke simpul yang sama.

	Kolom

A^1		**1**	**2**	**3**	**4**
Baris	**1**	0	3	∞	7
	2	8	0		
	3	5		0	
	4	2			0

Untuk setiap elemen matriks yang masih kosong, maka diterapkan rumus algoritma Floyd-Warshall dengan $k = 1$, yakni sebagai berikut:

$$A^1[i,j] = \min \{A^0[i,j],\ A^0[i,1] + A^0[1,j]\}$$

Untuk setiap elemen matriks yang masih kosong, maka proses yang dilakukan adalah sebagai berikut:

Elemen matriks [2,3]:

$$A^1[2,3] = \min \{A^0[2,3],\ A^0[2,1] + A^0[1,3]\}$$
$$A^1[2,3] = \min \{2,\ 8 + \infty\}$$
$$A^1[2,3] = 2$$

Elemen matriks [2,4]:

$$A^1[2,4] = \min \{A^0[2,4],\ A^0[2,1] + A^0[1,4]\}$$
$$A^1[2,4] = \min \{\infty,\ 8 + 7\}$$
$$A^1[2,4] = 15$$

Elemen matriks [3,2]:

$$A^1[3,2] = \min \{A^0[3,2],\ A^0[3,1] + A^0[1,2]\}$$
$$A^1[3,2] = \min \{\infty,\ 5 + 3\}$$
$$A^1[3,2] = 8$$

Elemen matriks [3,4]:

$$A^1[3,4] = \min \{A^0[3,4],\ A^0[3,1] + A^0[1,4]\}$$
$$A^1[3,4] = \min \{1,\ 5 + 7\}$$
$$A^1[3,4] = 1$$

Elemen matriks [4,2]:

$$A^1[4,2] = \min \{A^0[4,2],\ A^0[4,1] + A^0[1,2]\}$$
$$A^1[4,2] = \min \{\infty,\ 2 + 3\}$$
$$A^1[4,2] = 5$$

Elemen matriks [4,3]:

$$A^1[4,3] = \min \{A^0[4,3],\ A^0[4,1] + A^0[1,3]\}$$
$$A^1[4,3] = \min \{\infty,\ 2 + \infty\}$$
$$A^1[4,3] = \infty$$

Sehingga untuk $k = 1$, didapatkan elemen-elemen matriks A^1 sebagai berikut:

A^1 $k = 1$		Kolom			
		1	**2**	**3**	**4**
Baris	**1**	0	3	∞	7
	2	8	0	2	15
	3	5	8	0	1
	4	2	5	∞	0

Selanjutnya adalah memproses matriks A^1 dengan $k = 2$. Pada proses ini, dibuat kembali matriks baru A^2, dengan simpul *intermediate* adalah simpul 2. Proses pada langkah pertama diulang kembali dengan perbedaan elemen yang sama adalah elemen pada baris kedua dan kolom kedua dari matriks A^1.

A^2 $k = 2$		Kolom			
		1	**2**	**3**	**4**
Baris	**1**	0	3		
	2	8	0	2	15
	3		8	0	
	4		5		0

Rumus algoritma Floyd-Warshall yang digunakan pada proses ini adalah sebagai berikut:

$$A^2[i,j] = \min \{A^1[i,j],\ A^1[i,2] + A^1[2,j]\}$$

Untuk setiap elemen matriks yang masih kosong, maka proses yang dilakukan adalah sebagai berikut:

Elemen matriks [1,3]:

$$A^2[1,3] = \min \{A^1[1,3],\ A^1[1,2] + A^1[2,3]\}$$
$$A^2[1,3] = \min \{\infty,\ 3 + 2\}$$
$$A^2[1,3] = 5$$

Elemen matriks [1,4]:

$$A^2[1,4] = \min \{A^1[1,4],\ A^1[1,2] + A^1[2,4]\}$$
$$A^2[1,4] = \min \{7,\ 3 + 15\}$$
$$A^2[1,4] = 7$$

Elemen matriks [3,1]:

$$A^2[3,1] = \min \{A^1[3,1],\ A^1[3,2] + A^1[2,1]\}$$
$$A^2[3,1] = \min \{5,\ 8 + 8\}$$
$$A^2[3,1] = 5$$

Elemen matriks [3,4]:

$$A^2[3,4] = \min \{A^1[3,4],\ A^1[3,2] + A^1[2,4]\}$$
$$A^2[3,4] = \min \{1,\ 8 + 15\}$$
$$A^2[3,4] = 1$$

Elemen matriks [4,1]:

$$A^2[4,1] = \min \{A^1[4,1],\ A^1[4,2] + A^1[2,1]\}$$
$$A^2[4,1] = \min \{2,\ 5 + 8\}$$
$$A^2[4,1] = 2$$

Elemen matriks [4,3]:

$$A^2[4,3] = \min \{A^1[4,3],\ A^1[4,2] + A^1[2,3]\}$$
$$A^2[4,3] = \min \{\infty,\ 5 + 2\}$$
$$A^2[4,1] = 7$$

Sehingga untuk $k = 2$, didapatkan elemen-elemen matriks A^2 yakni sebagai berikut:

A^2 $k = 2$		Kolom			
		1	**2**	**3**	**4**
Baris	**1**	0	3	5	7
	2	8	0	2	15
	3	5	8	0	1
	4	2	5	7	0

Proses ketiga adalah untuk $k = 3$. Dibuat matriks baru A^3. Proses yang sama dilakukan seperti pada langkah 1 dan 2, dengan elemen yang sama adalah elemen pada baris ketiga dan kolom ketiga.

A^3 $k = 3$		Kolom			
		1	**2**	**3**	**4**
Baris	**1**	0		5	
	2		0	2	
	3	5	8	0	1

	4			7	0

Rumus yang digunakan pada proses ini adalah sebagai berikut:

$$\mathbf{A^3[i,j] = min \{A^2[i,j],\ A^2[i,3] + A^2[3,j]\}}$$

Untuk setiap elemen yang masih kosong, maka proses yang dilakukan adalah sebagai berikut:

Elemen matriks [1,2]:

$$\mathbf{A^3[1,2] = min \{A^2[1,2],\ A^2[1,3] + A^2[3,2]\}}$$
$$\mathbf{A^3[1,2] = min \{3, 5 + 8\}}$$
$$\mathbf{A^3[1,2] = 3}$$

Elemen matriks [1,4]:

$$\mathbf{A^3[1,4] = min \{A^2[1,4],\ A^2[1,3] + A^2[3,4]\}}$$
$$\mathbf{A^3[1,4] = min \{7, 5 + 1\}}$$
$$\mathbf{A^3[1,4] = 6}$$

Elemen matriks [2,1]:

$$\mathbf{A^3[2,1] = min \{A^2[2,1],\ A^2[2,3] + A^2[3,1]\}}$$
$$\mathbf{A^3[2,1] = min \{8, 2 + 5\}}$$
$$\mathbf{A^3[2,1] = 7}$$

Elemen matriks [2,4]:

$$\mathbf{A^3[2,4] = min \{A^2[2,4],\ A^2[2,3] + A^2[3,4]\}}$$
$$\mathbf{A^3[2,4] = min \{15, 2 + 1\}}$$
$$\mathbf{A^3[2,4] = 3}$$

Elemen matriks [4,1]:

$$\mathbf{A^3[4,1] = min \{A^2[4,1],\ A^2[4,3] + A^2[3,1]\}}$$
$$\mathbf{A^3[4,1] = min \{2, 7 + 1\}}$$
$$\mathbf{A^3[4,1] = 2}$$

Elemen matriks [4,2]:

$$\mathbf{A^3[4,2] = min \{A^2[4,2],\ A^2[4,3] + A^2[3,2]\}}$$
$$\mathbf{A^3[4,2] = min \{5, 7 + 8\}}$$
$$\mathbf{A^3[4,2] = 5}$$

Sehingga didapatkan elemen-elemen matriks A^3 yakni sebagai berikut:

A^3 $k = 3$		**Kolom**			
		1	**2**	**3**	**4**
Baris	**1**	0	3	5	6
	2	7	0	2	3
	3	5	8	0	1
	4	2	5	7	0

Proses terakhir adalah untuk $k = 4$. Pada proses ini,dibuat matriks A^4 dengan elemen yang sama dengan elemen pada matriks A^3 adalah seluruh elemen pada baris keempat dan kolom keempat.

A^4 $k = 4$		**Kolom**			
		1	**2**	**3**	**4**
Baris	**1**	0			6
	2		0		3
	3			0	1
	4	2	5	7	0

Rumus yang digunakan pada proses ini adalah sebagai berikut:

$$\mathbf{A^4[i,j] = min \{A^3[i,j],\ A^3[i,4] + A^3[4,j]\}}$$

Untuk setiap elemen yang masih kosong, maka proses yang dilakukan adalah sebagai berikut:

Elemen matriks [1,2]:

$$\mathbf{A^4[1,2] = min \{A^3[1,2],\ A^3[1,4] + A^3[4,2]\}}$$
$$\mathbf{A^4[1,2] = min \{3, 6 + 5\}}$$
$$\mathbf{A^4[1,2] = 3}$$

Elemen matriks [1,3]:

$$\mathbf{A^4[1,3] = min \{A^3[1,3],\ A^3[1,4] + A^3[4,3]\}}$$
$$\mathbf{A^4[1,3] = min \{5, 6 + 7\}}$$
$$\mathbf{A^4[1,2] = 5}$$

Elemen matriks [2,1]:

$$\mathbf{A^4[2,1] = min \{A^3[2,1],\ A^3[2,4] + A^3[4,1]\}}$$
$$\mathbf{A^4[2,1] = min \{7, 3 + 2\}}$$
$$\mathbf{A^4[2,1] = 5}$$

Elemen matriks [2,3]:

$$\mathbf{A^4[2,3] = min \{A^3[2,3],\ A^3[2,4] + A^3[4,3]\}}$$
$$\mathbf{A^4[2,3] = min \{2, 3 + 7\}}$$
$$\mathbf{A^4[2,1] = 2}$$

Elemen matriks [3,1]:

$$\mathbf{A^4[3,1] = min \{A^3[3,1],\ A^3[3,4] + A^3[4,1]\}}$$
$$\mathbf{A^4[3,1] = min \{5, 1 + 2\}}$$
$$\mathbf{A^4[3,1] = 3}$$

Elemen matriks [3,2]:

$$A^4[3,2] = \min \{A^3[3,2],\ A^3[3,4] + A^3[4,2]\}$$
$$A^4[3,2] = \min \{8,\ 1 + 5\}$$
$$A^4[3,2] = 6$$

Sehingga untuk k = 4, elemen-elemen matriks A^4 adalah sebagai berikut:

A^4 $k = 4$		Kolom			
		1	**2**	**3**	**4**
Baris	**1**	0	3	5	6
	2	5	0	2	3
	3	3	6	0	1
	4	2	5	7	0

Karena jumlah simpul pada graf contoh berjumlah 4 dan k sudah mencapai proses keempat, maka elemen-elemen matriks pada A4 merupakan bobot terkecil sebagai jalur terpendek antar pasangan simpul yang dapat ditempuh.

Pasangan Simpul	Jarak Terpendek
1 - 2	3
1 - 3	5
1 - 4	6
2 - 1	5
2 - 3	2
2 - 4	3
3 - 1	3
3 - 2	6
3 - 4	1
4 - 1	2
4 - 2	5
4 - 3	7

III. *PSEUDO-CODE*

Dari pembahasan contoh persoalan tersebut, implementasi algoritma Floyd-Warshall pada dasarnya cukup sederhana.Untuk setiap k = 1, k = 2, sampai k = n,setiap baris dan kolom matriks diproses menggunakan rumus algoritma Floyd-Warshall. Secara umum, berikut adalah *pseudo code* dari algoritma Floyd-Warshall.

```
for k = 1 to n
    for i = 1 to n
        for j = 1 to n
            A[i,j] = min(A[i,j], A[i,k] + A[k,j])
```

IV. PEMBANGUNAN JALUR TERPENDEK

Pada contoh persoalan yang telah dibahas pada bagian sebelumnya, hasil yang didapatkan hanya berupa daftar bobot terkecil untuk setiap pasangan simpul, namun tidak disebutkan bagaimana lintasan yang ditempuh agar didapatkan bobot terkecil tersebut. Terdapat dua cara agar didapatkan lintasan yang ditempuh untuk mendapatkan jarak terpendek, yaitu sebagai berikut.

A. *Path reconstruction*

Pada metode ini, kita dapat menginisialisasi terlebih dahulu sebuah matriks yang menyimpan jalur yang ditempuh antara dua simpul *i* dan *j*. Simpan bobot lintasan terpendek baru antara simpul *i* dan *j*. Kemudian simpan fakta bahwa lintasan terpendek antara *i* dan *j* melewati simpul *intermediate k*. Berikut adalah *pseudo-code* sederhana tentang bagaimana *path reconstruction* ini diimplementasikan.

```
if (A[i,k] + A[k,j] < A[k,j]) then
    vertex
    A[i,j] = A[i,k] + A[k,j]
    path[i,j] = path[k,j]
```

Sehingga apabila digabungkan, maka *pseudo-code* sederhana untuk pencarian jalur terpendek menggunakan algoritma Foyd-Warshall ini adalah sebagai berikut.

```
procedure [array] FloydWarshall(A, P)
    for k in 1 to n do
        for i in 1 to n do
            for j in 1 to n do
                if A[i][j] > A[i][k] + A[k][j] then
                    A[i][j] = A[i][k] + A[k][j]
                    P[i][j] = P[k][j]
    return P
```

B. *Transitive Closure*

Pada *transitive closure*, pemrosesan graf akan memberikan informasi yang lengkap mengenai seluruh lintasan untuk seluruh pasangan simpul dengan bobot terkecil. Sederhananya, berikut adalah *pseudo-code* untuk *transitive closure*.

```
Transitive-Closure(W[1..n][1..n])
01 T ← W     // T(0)
02 for k ←1 to n do // compute T(k)
03    for i ←1 to n do
04       for j ←1 to n do
05          T[i][j] ← T[i][j] ∨ (T[i][k] ∧ T[k
06 return T
```

Sumber: www.cs.ucf.edu/~sarahb/.../DynProg_FloydWarshall.ppt

V. CONCLUSION

Algoritma Foyd-Warshall merupakan salah satu metode dalam program dinamis yang menyelesaikan *all pairs shortest path*. Dengan algoritma ini, didapatkan seluruh bobot terkecil untuk setiap pasangan simpul dalam sebuah graf lengkap berarah. Namun, untuk mendapatkan lintasan terpendek dari suatu pasangan simpul tertentu atau seluruh pasangan simpul, dibutuhkan modifikasi pada dasar algoritma Foyd-Warshall atau menggunakan *transitive closure*.

VII. ACKNOWLEDGMENT

Dalam penyusunan makalah Strategi Algoritma ini, penulis ingin menyampaikan rasa syukur kepada Tuhan Yang Maha Esa atas berkah dan rahmat yang telah dilimpahkannya. Penulis juga ingin menyampaikan terima kasih kepada orang tua penulis yang selalu memberikan dukungan dalam berbagai keadaan dan situasi. Penulis juga ingin menyampaikan terima kasih kepada dosen-dosen pengajar mata kuliah Strategi Algoritma yang telah mengajarkan pokok-pokok dan pondasi dari desain dan analisis dalam mata kuliah Strategi Algoritma ini. Penulis juga tidak lupa ingin menyampaikan terima kasih kepada asisten-asisten mata kuliah Strategi Algoritma serta rekan-rekan penulis yang telah banyak membantu dan memberi arahan selama perkuliahan Strategi Algoritma ini berlangsung.

REFERENCES

[1] http://www.cs.ucf.edu/~dmarino/ucf/cop3503/lectures/

[2] http://www.cs.umd.edu/~meesh/351/mount/lectures/lect24-floyd-warshall.pdf

[3] http://home.cse.ust.hk/faculty/golin/COMP271Sp03/Notes/MyL15.pdf

[4] https://web.stanford.edu/class/cs97si/07-shortest-path-algorithms.pdf

[5] http://math.mit.edu/~rothvoss/18.304.1PM/Presentations/1-Chandler-18.304lecture1.pdf

[6] Levitin, Anany. *Introduction to The Design and Analysis of Algorithms 3rd Edition*. United States of America: Pearson, 2012.

[7] Munir, Rinaldi. Diktat Kuliah IF2211 Strategi Algoritma. Bandung: Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, Januari 2009.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 14 Mei 2018

Letivany Aldina/13514067