Pencarian Rute Terpendek Pada Setiap Pasangan Simpul Graf dengan Menggunakan Metode Floyd

M. Faisal Baehaki (13506108)
Mahasiswa Teknik Informatika Institut Teknologi Bandung
Jl. Ganesha No. 10 Bandung 40135
e-mail: faisal.baihaki@comlabs.itb.ac.id

ABSTRAK

Graf adalah salah satu cabang matematik yang dapat digunakan dalam menyelesaikan masalah melibatkan perkiraan yang kompleks. Diantara penggunaannya yang populer adalah penganalisaan sistem rangkaian jalan raya melalui kaidah teori graf. Maklumat anggaran berkaitan dengan jarak, *cost*, dan waktu tempuh perjalanan pada rute terpendek yang akan turut disenaraikan melalui sistem yang dibangun. Algoritma Floyd Warshall digunakan untuk proses pencarian rute terpendek pada lokasi awal dan akhir yang dikehendaki pengguna sistem.
Kata kunci: *Graf, cost, algoritma, floyd, dan rute*.

1. PENDAHULUAN

Persoalan mencari lintasan terpendek di dalam graf merupakan persoalan salah satu optimasi. Graf yang digunakan dalam pencarian lintasan terpendek adalah graf berbobot (*weighted graph*), yaitu graf yang setiap sisinya diberikan suatu nilai atau bobot. Bobot pada sisi graf dapat menyatakan jarak antar kota, waktu pengiriman pesan, ongkos pembangunan, dan sebagainya. Asumsi yang kita gunakan adalah bahwa semua bobot bernilai positif. Kata "terpendek" berbeda-beda maknanya bergantung pada tipikal persoalan yang akan diselesaikan. Namun, secara umum "terpendek" berarti meminimasi bobot pada suatu lintasan di dalam graf.

Contoh-contoh terapan pencarian lintasan terpendek misalnya :

1. Misalkan simpul pada graf dapat merupakan kota, sedangkan sisi menyatakan rute yang menghubungkan dua buah kota. Bobot sisi graf dapat menyatakan jarak antara dua buah kota atau rata-rata waktu tempuh antara dua buah kota. Apabila terdapat lebih dari satu lintasan dari kota A ke kota B, maka persoalan lintasan terpendek di sini adalah menentukan jarak terpendek atau waktu tersingkat dari kota A ke kota B.
2. Misalkan simpul pada graf dapat merupakan terminal komputer atau simpul komunikasi dalam suatu jaringan, sedangkan sisi menyatakan saluran komunikasi yang menghubungkan dua buah terminal. Bobot pada graf dapat menyatakan biaya pemakaian saluran komunikasi antara dua buah terminal, jarak antara dua buah terminal, atau waktu pengiriman pesan antara dua buah terminal. Persoalan lintasan terpendek di sini adalah menentukan jalur komunikasi terpendek antara dua buah teminal komputer. Lintasan terpendek akan menghemat waktu pengiriman pesan dan biaya komunikasi.

2. METODE

Sebelum menghadirkan metode penyelesaian ini, saya jelaskan terlebih dahulu beberapa notasi-notasi yang dibutuhkan. Nomor-nomor simpul : 1, 2, .. , N. Panjang rute terpendek dari simpul i ke simpul j dinotasikan dengan d_{ij}^{m}, dimana hanya simpul m pertama yang diizinkan menjadi simpul tengah (Untuk diingat bahwa simpul tengah adalah setiap simpul pada rute kecuali simpul awal dan akhir). Jika tidak ada sebuah rute pun maka nilai $d_{ij}^{m} = \infty$. Dari definisi d_{ij}^{m} didapat ketika d_{ij}^{0} artinya panjang rute terpendek dari i ke j tidak memiliki simpul tengah.

D^m menotasikan matriks NxN di yang elemen i dan j-nya didapat dari d_{ij}^{m}. Jika kita mengetahui panjang setiap sisi pada sebuah graf, maka kita dapat menentukan matriks D^0. Intinya, kita berharap dapat menentukan D^N sebagai matriks dari panjang rute terpendek.

Algoritma Floyd untuk pencarian rute terpendek dimulai dengan D^0 dan mengalkulasi D^1 dari D^0. Selanjutnya, algoritma ini mengalkulasi D^2 dari D^1. Proses ini diulang sampai D^N dihitung dari D^{N-1}. Ide yang mendasari dituangkan ke dalam beberapa perhitungan di bawah ini :

a) Sebuah rute terpendek berasal dari simpul i ke simpul m yang mengizinkan hanya simpul m-1 pertama yang dijadikan simpul tengah.
b) Sebuah rute terpendek berasal dari simpul m ke simpul j yang mengizinkan hanya simpul m-1 pertama yang dijadikan simpul tengah.
c) Sebuah rute terpendek berasal dari simpul i ke simpul j yang mengizinkan hanya simpul m-1

pertama yang dijadikan simpul tengah. Karena tidak ada sirkuit dengan panjang negatif maka pemendek dari rute diberikan pada (d) dan (e) merupakan rute terpendek dari i ke j yang hanya mengizinkan simpul m pertama sebagai simpul tengah.

d) Gabungan dari rute-rute pada item a dan b

e) Rute pada item 3.

Dengan demikian,

$d_{ij}^{m} = \min (d_{im}^{m-1} + d_{mj}^{m-1}, d_{ij}^{m-1})$ (1)

Dari persamaan (1), kita dapat melihat bahwa hanya elemen dari matriks D^{m-1} yang dibutuhkan untuk menghitung elemen matriks D^m. Lebih jauh lagi, penghitungan ini dapat diselesaikan tanpa referensi dari graf itu sendiri.

2.1 Algoritma Floyd

Selanjutnya saya akan menjelaskan solusi pencarian rute terpendek dari setiap pasangan simpul dengan menggunakan algoritma Floyd ini. Langkah-langkahnya sebagai berikut :

Langkah 1 : Nomor setiap simpul pada graf : 1, 2, .. , N. Tentukan matriks D^0 yang elemen i dan j-nya sama dengan panjang terpendek sisi dari simpul i ke simpul j. Jika tidak ada sisi yang eksis, maka $d_{ij}^{0} = \infty$, $d_{ii}^{0} = 1$ untuk setiap i.

Langkah 2 : Untuk m = 1, 2, …, N, berturut-turut tentukan elemen pada D^m yang berasal dari D^{m-1} menggunakan formula rekursif di bawah ini

$$d_{ij}^{m} = \min (d_{im}^{m-1} + d_{mj}^{m-1}, d_{ij}^{m-1})$$

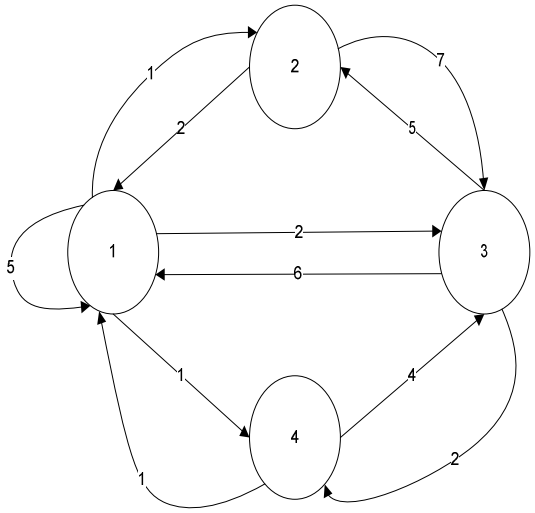
selama setiap elemen ditentukan, simpan setiap rute yang menunjukkannya. Ketika terminasi, elemen i dan j dari matriks D^N mewakili panjang rute terpendek dari simpul i ke simpul j.

Optimasi dari algoritma ini menginduksi dari fakta bahwa panjang rute terpendek dari i ke j yang mengizinkan hanya simpul m pertama sebagai simpul tengah harus paling kecil dari (a) panjang rute terpendek dari i ke j yang mengizinkan hanya simpul m-1 pertama sebagai simpul tengah dan (b) panjang rute terpendek dari i ke j yang mengizinkan hanya simpul m pertama sebagai simpul tengah dan menggunakan sekali simpul m_{th} sebagai simpul tengah.

Perlu diingat bahwa $d_{ii}^{m} = 0$ untuk setiap i dan m. Karenanya, elemen diagonal dari matriks D^1, D^2, .. , D^N tidak perlu dikalkulasi. Lebih jauh lagi, $d_{im}^{m-1} = d_{im}^{m}$ dan $d_{mi}^{m-1} = d_{mi}^{m}$, untuk setiap i = 1, 2, .. , N. Oleh sebab itu pada komputasi matriks D^m, baris dan kolom m_{th} tidak perlu dikalkulasi. Dengan demikian pada setiap matriks D^m hanya elemen (N-1)(N-2) yang bukan merupakan diagonal pada baris dan kolom m_{th} tidak perlu dikalkulasi.

Saya akan memberikan contoh bagaimana algoritma Floyd ini bekerja pada pencarian rute terpendek simpul graf.

2.2 Contoh Kasus



Gambar 1. Graf yang akan dicari rute terpendeknya

$$D^0 = \begin{pmatrix} 0 & 1 & 2 & 1 \\ 2 & 0 & 7 & \infty \\ 6 & 5 & 0 & 2 \\ 1 & \infty & 4 & 0 \end{pmatrix}$$

Elemen-elemen dari D^1 dan rute terpendek yang berhubungan dijelaskan di bawah ini :

$d_{ij}^{1} = \min \{d_{i1}^{0} + d_{1j}^{0}, d_{ij}^{0}\}$ Corresponding Path

$d_{11}^{1} = d_{11}^{0} = 0$

$d_{12}^{1} = d_{12}^{0} = 1$ (1,2)

$d_{13}^{1} = d_{13}^{0} = 2$ (1,3)

$d_{13}^{1} = d_{14}^{0} = 1$ (1,4)

$d_{21}^{1} = d_{21}^{0} = 2$ (2,1)

$d_{22}^{1} = 0$

$d_{23}^{1} = \min \{d_{21}^{0} + d_{13}^{0}, d_{23}^{0}\}$ (2,1)

$= \min \{2 + 2, 7\} = 4$ (1,3)

$d_{24}^{1} = \min \{d_{21}^{0} + d_{14}^{0}, d_{24}^{0}\}$ (2,1)

$= \min \{2 + 1, \infty\} = 3$ (1,4)

$d_{31}^{1} = d_{31}^{0} = 6$ (3,1)

$d_{32}^{1} = \min \{d_{31}^{0} + d_{12}^{0}, d_{32}^{0}\}$ (3,2)

$= \min \{6+1, 5\} = 5$

$d^{1}_{33} = 0$

$d^{1}_{34} = \min \{d^{0}_{31} + d^{0}_{14}, d^{0}_{34}\}$ (3,4)

$= \min \{6 + 1, 2\} = 2$

$d^{1}_{41} = d^{0}_{41} = 1$ (4,1)

$d^{1}_{42} = \min \{d^{0}_{41} + d^{0}_{12}, d^{0}_{42}\}$ (4,1)

$= \min \{1 + 1, \infty\} = 2$ (1,2)

$d^{1}_{43} = \min \{d^{0}_{41} + d^{0}_{13}, d^{0}_{43}\}$ (4,1)

$= \min \{1 + 2, 4\} = 3$ (1,3)

$d^{1}_{44} = 0$

Dengan jalan yang serupa, matriks D^2, D^3, dan D^4 dan seluruh jalan-jalan yang berhubungan dapat dihitung. Hasil dari perhitungan ini adalah:

$$D^2 = \begin{pmatrix} 0 & 1 & 2 & 1 \\ 2 & 0 & 4 & 3 \\ 6 & 5 & 0 & 2 \\ 1 & 2 & 3 & 0 \end{pmatrix}$$

Rute terpendek untuk D^2 :

$$\begin{pmatrix} (1,2) & (1,3) & (1,4) \\ (2,1) & (2,1)(1,3) & (2,1)(1,4) \\ (3,1)(3,2) & & (3,4) \\ (4,1)(4,1)(1,2) & (4,1)(1,3) & \end{pmatrix}$$

$$D^3 = \begin{pmatrix} 0 & 1 & 2 & 1 \\ 2 & 0 & 4 & 3 \\ 6 & 5 & 0 & 2 \\ 1 & 2 & 3 & 0 \end{pmatrix}$$

Rute terpendek untuk D^3 :

$$\begin{pmatrix} (1,2) & (1,3) & (1,4) \\ (2,1) & (2,1)(1,3) & (2,1)(1,4) \\ (3,1)(3,2) & & (3,4) \\ (4,1)(4,1)(1,2) & (4,1)(1,3) & \end{pmatrix}$$

$$D^4 = \begin{pmatrix} 0 & 1 & 2 & 1 \\ 2 & 0 & 4 & 3 \\ 6 & 5 & 0 & 2 \\ 1 & 2 & 3 & 0 \end{pmatrix}$$

Rute terpendek untuk D^4 :

$$\begin{pmatrix} (1,2) & (1,3) & (1,4) \\ (2,1) & (2,1)(1,3) & (2,1)(1,4) \\ (3,4)(4,1)(3,4)(4,1)(1,2) & & (3,4) \\ (4,1) & (4,1)(1,2) & (4,1)(1,3) \end{pmatrix}$$

Catatan bahwa karena jumlah simpul-simpul dapat semaunya, maka algoritma Floyd ini akan mencari jalan terpendek pada interasi awal jika simpul dengan nomor tertutup antara simpul yang satu dengan yang lainnya mempunyai kenyataan bahwa simpul memang tertutup antara satu dengan lainnya.

Pada contoh numerikal sebelumnya, sisi aktual yang terdiri dari masing-masing jalan terpendek telah direkam sebagai bukti perilaku algoritma ini. Untuk masalah dari ukuran sebenarnya, sangat jelas bahwa prosedur ini sulit diimplementasikan. Akibatnya, kita perlu untuk mengembangkan teknik yang lebih efisien untuk menentukan sisi aktual yang membentuk jalan terpendek ini.

2.3 Analisis

Algoritma Floyd harus memperhitungkan N matriks D^1, D^2, …, D^N. Masing-masing dari matriks ini terdiri dari N^2 elemen. Oleh karena itu, total N^3 elemen yang harus dihitung jika menggunakan Algoritma ini. Masing-masing dari komputasi ini memerlukan persamaan (1) dengan satu tambahan dan satu minimasi. Total jumlah komputasi yang dibutuhkan oleh algoritma Floyd sebanding dengan $2N^3$, atau dengan terminologi teknik, algoritma ini membutuhkan $O(2N^3)$ waktu eksekusi.

III. SIMPULAN

Algoritma Floyd memang merupakan salah satu alternatif untuk menyelesaikan masalah pencarian rute terdekat pada simpul graf. Waktu komputasi algoritma ini memang tidak cukup baik karena bersifat eksponensial. Namun untuk kasus dengan N yang kecil maka algoritma ini cukup baik jika digunakan. Waktu komputasi dari algoritma ini sebanding dengan $2N^3$ dimana N adalah jumlah simpul. Untuk menyelesaikan kasus-kasus pencarian rute terpendek digunakan formula rekursif pada persamaan (1).

REFERENSI

[1] Ir. Rinaldi Munir, M.T. “Strategi Algoritmik”, Lab Ilmu dan Rekayasa Komputasi Teknik Informatika ITB, 2005.

[2] Ir. Rinaldi Munir, M.T. “Matematika Diskrit”, Lab Ilmu dan Rekayasa Komputasi Teknik Informatika ITB, 2005.

[3] http://en.wikipedia.org/wiki/Floyd

[4] Edward Minieka, “Optimization Algorithms for Networks and Graphs” Marcell Deker Inc