Perbandingan Algoritma Dijkstra dan Algoritma Floyd-Warshall dalam Penentuan Lintasan Terpendek (*Single Pair Shortest Path*)

Raden Aprian Diaz Novandi

Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jalan Ganeça 10, Kotamadya Bandung 40135
e-mail: if15102@students.if.itb.ac.id

ABSTRAK

Penentuan lintasan terpendek dari satu titik ke titik yang lain adalah masalah yang sering ditemui dalam kehidupan sehari-hari. Berbagai kalangan menemui permasalahan serupa dengan variasi yang berbeda, contohnya seorang pengemudi yang mencari jalur terpendek dari tempat asal ke tempat tujuan, pengantar pesanan makanan cepat saji yang juga mencari jalur terpendek dari tempat asal ke tempat tujuan, dan juga seorang desainer jaringan komputer yang harus mendesain skema perutean pada jaringan yang dia tangani agar memaksimalkan performa jaringan dan meminimalkan beban yang harus ditangani oleh jaringan tersebut.
Seiring dengan waktu yang berjalan dan juga perkembangan ilmu pengetahuan dan teknologi, permasalahan pencarian lintasan terpendek ini telah terpecahkan dengan berbagai algoritma. Beberapa algoritma populer yang memecahkan persoalan pencarian lintasan terpendek tersebut adalah algoritma Dijkstra, Bellman-Ford, A-*star*, Floyd-Warshall, dsb.
Batasan makalah ini adalah mengenai perbandingan antara Algoritma Dijkstra dan Algoritma Floyd-Warshall (Roy-Floyd) dalam penentuan lintasan terpendek dari satu titik asal ke satu titik tujuan (*single pair shortest path*) yang biasa dimodelkan dalam suatu graf berbobot.

Kata dan frase kunci: *Dijkstra*, greedy, *Floyd-Warshall*, *pemrograman dinamis*, single pair shortest path

1. PENDAHULUAN

Salah satu persoalan optimasi yang sering ditemui dalam kehidupan sehari-hari adalah pencarian lintasan terpendek (*shortest path*). Persoalan ini bisa dimodelkan ke dalam suatu graf berbobot dengan nilai pada masing-masing sisi yang merepresentasikan persoalan yang akan dipecahkan.

Kata "terpendek" pada kata "lintasan terpendek" ini tidak berarti hanya bisa diartikan jarak secara fisik, namun hal itu tergantung dari tipe persoalan yang akan dipecahkan. Bisa jadi kata tersebut memiliki makna tingkat aksesibilitas suatu simpul dalam graf dari simpul yang lain.

Persoalan lintasan terpendek yang akan dibahas di makalah ini adalah lintasan terpendek antara dua buah simpul tertentu di dalam graf (*single pair shortest path*). Persoalan ini bisa jadi merupakan analogi dari beberapa persoalan populer yang tercantum di abstrak makalah.

2. PENJELASAN MENGENAI ALGORITMA DIJKSTRA DAN FLOYD-WARSHALL

2.1 Algoritma Dijkstra

Algoritma Dijkstra merupakan salah satu varian dari algoritma greedy, yaitu salah satu bentuk algoritma populer dalam pemecahan persoalan yang terkait dengan masalah optimasi. Sifatnya sederhana dan lempang (*straightforward*). Sesuai dengan artinya yang secara harafiah berarti tamak atau rakus – namun tidak dalam konteks negatif –, algoritma greedy ini hanya memikirkan solusi terbaik yang akan diambil pada setiap langkah tanpa memikirkan konsekuensi ke depan. Prinsipnya, ambillah apa yang bisa Anda dapatkan saat ini (*take what you can get now!*), dan keputusan yang telah diambil pada setiap langkah tidak akan bisa diubah kembali. Intinya algoritma *greedy* ini berupaya membuat pilihan nilai optimum lokal pada setiap langkah dan berharap agar nilai optimum lokal ini mengarah kepada nilai optimum global.

Elemen-elemen penyusun algoritma *greedy* adalah:

1. Himpunan kandidat, C
   Himpunan ini berisi elemen-elemen yang memiliki peluang untuk membentuk solusi. Pada persoalan lintasan terpendek dalam graf, himpunan kandidat ini adalah himpunan simpul pada graf tersebut.

2. Himpunan solusi, S
   Himpunan ini berisi solusi dari permasalahan yang diselesaikan dan elemennya terdiri dari elemen dalam himpunan kandidat namun tidak semuanya atau dengan kata lain himpunan solusi ini adalah upabagian dari himpunan kandidat.

3. Fungsi seleksi
   Fungsi seleksi adalah fungsi yang akan memilih setiap kandidat yang yang memungkinkan untuk menghasilkan solusi optimal pada setiap langkahnya.

4. Fungsi kelayakan
   Fungsi kelayakan akan memeriksa apakah suatu kandidat yang telah terpilih (terseleksi) melanggar *constraint* atau tidak. Apabila kandidat melanggar constraint maka kandidat tidak akan dimasukkan ke dalam himpunan solusi.

5. Fungsi objektif
   Fungsi objektif akan memaksimalkan atau meminimalkan nilai solusi. Tujuannya adalah memilih satu saja solusi terbaik dari masing-masing anggota himpunan solusi.

2.1.1 Analisis Algoritma Dijkstra

Ada beberapa kasus pencarian lintasan terpendek yang diselesaikan menggunakan algoritma Dijkstra, yaitu: pencarian lintasan terpendek antara dua buah simpul tertentu (*a pair shortest path*), pencarian lintasan terpendek antara semua pasangan simpul (*all pairs shortest path*), pencarian lintasan terpendek dari simpul tertentu ke semua simpul yang lain (*single-source shortest path*), serta pencarian lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu (*intermediate shortest path*).

Penggunaan strategi *greedy* pada algoritma Dijkstra adalah:
Pada setiap langkah, ambil sisi berbobot minimum yang menghubungkan sebuah simpul yang sudah terpilih dengan sebuah simpul lain yang belum terpilih. Lintasan dari simpul asal ke simpul yang baru haruslah merupakan lintasan yang terpendek di antara semua lintasannya ke simpul-simpul yang belum terpilih.

2.1.2 Pseudokode

```
procedure Dijkstra(INPUT m: matriks, a : simpul awal)
{  Mencari lintasan terpendek dari simpul awal a
   ke semua simpul lainnya.
   Masukan: matriks ketetanggaan (m) dari graf
   berbobot G dan simpul awal a
   Keluaran: lintasan terpendek dari a ke semua
   simpul lainnya
}
```

```
Kamus:
   s : array [1..n] of integer
   d : array [1..n] of integer
   i : integer

Algoritma:
     { Langkah 0 (inisialisasi: }
     traversal [1..n]
        s_i ← 0
        d_i ← m_ai { Langkah 1: }
     s_a ← 1
     d_a ← ∞

     { Langkah 2, 3, ..., n-1: }
     traversal [2..n-1]
         cari j  sedemikian sehingga s_j= 0 dan
         d_j = min {d1, d2, ..., dn}
         s_j ← 1 {simpul j sudah terpilih}
         perbarui d_i, untuk i = 1, 2, 3, s.d. n
dengan:
           d_i(baru) = min{d_i(lama),d_j+ m_ji}
```

Kompleksitas algoritma Dijkstra adalah $O(n^2)$, dengan n adalah jumlah simpul pada graf. Kompleksitas ini bisa diperbaiki dengan penggunaan struktur data senarai ketetanggaan (*adjacency list*) atau antrian prioritas (*priority queue*) untuk memperoleh kompleksitas $O((m+n) \log n)$.

2.2 Algoritma Floyd-Warshall

Algoritma Floyd-Warshall adalah salah satu varian dari pemrograman dinamis, yaitu suatu metode yang melakukan pemecahan masalah dengan memandang solusi yang akan diperoleh sebagai suatu keputusan yang saling terkait. Artinya solusi-solusi tersebut dibentuk dari solusi yang berasal dari tahap sebelumnya dan ada kemungkinan solusi lebih dari satu.

Hal yang membedakan pencarian solusi menggunakan pemrograman dinamis dengan algoritma *greedy* adalah bahwa keputusan yang diambil pada tiap tahap pada algoritma greedy hanya berdasarkan pada informasi yang terbatas sehingga nilai optimum yang diperoleh pada saat itu Jadi pada algoritma *greedy*, kita tidak memikirkan konsekuensi yang akan terjadi seandainya kita memilih suatu keputusan pada suatu tahap.

Dalam beberapa kasus, algoritma *greedy* gagal memberikan solusi terbaik karena kelemahan yang dimilikinya tadi. Di sinilah peran pemrograman dinamis yang mencoba untuk memberikan solusi yang memiliki pemikiran terhadap konsekuensi yang ditimbulkan dari pengambilan keputusan pada suatu tahap. Pemrograman dinamis mampu mengurangi pengenumerasian keputusan yang tidak mengarah ke solusi. Prinsip yang dipegang oleh pemrograman dinamis adalah prinsip optimalitas, yaitu *jika solusi total optimal, maka bagian solusi sampai suatu tahap (misalnya tahap ke-i) juga optimal*.

2.2.1 Karakteristik Program Dinamis

Beberapa karakteristik yang dimiliki oleh program dinamis antara lain:

1. Persoalan dibagi atas beberap tahap, yang stiap tahapnya hanya akan diambil satu keputusan.

2. Masing-masing tahap terdiri atas sejumlah status yang saling berhubungan dengan status tersebut. Status yang dimaksud di sini adalah berbagai kemungkinan masukan yang ada pada tahap tersebut.

3. Ketika masuk ke suatu tahap, hasil keputusan akan transformasi.

4. Ongkos (beban) pada suatu tahap akan meningkat secara teratur seiring bertambahnya jumlah tahapan.

5. Ongkos yang ada pada suatu tahap tergantung dari ongkos tahapan yang telah berjalan dan ongkos pada tahap itu sendiri.

6. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan pada tahap sebelumnya.

7. Terdapat hubungan rekursif yang menyatakan bahwa keputusan terbaik dalam setiap status pada tahap k akan memberikan keputusan terbaik untuk setiap status pada tahap k + 1

8. Prinsip optimalitas berlaku pada persoalan yang dimaksud.

Dalam proses penyelesaian menggunakan program dinamis, pendekatan yang dilakukan bisa jadi ada dua macam, yaitu pendekatan maju (*forward*) dan mundur (*backward*), dan perlu untuk diketahui pula bahwa solusi yang dihasilkan dari kedua pendekatan itu adalah sama. Solusi dari program dinamis bisa jadi lebih dari satu macam.

2.2.2 Analisis Algoritma Floyd-Warshall

Algoritma Floyd-Warshall membandingkan semua kemungkinan lintasan pada graf untuk setiap sisi dari semua simpul. Menariknya, algoritma ini mampu mengerjakan proses perbandingkan ini sebanyak V^3 kali (bandingkan dengan kemungkinan jumlah sisi sebanyak V^2 (kuadrat jumlah simpul) pada graf, dan setiap kombinasi sisi diujikan). Hal tersebut bisa terjadi karena adanya perkiraan pengambilkan keputusan (pemilihan jalur terpendek) pada setiap tahap antara dua simpul, hingga perkiraan tersebut diketahui sebagai nilai optimal,

Misalkan terdapat suatu graf G dengan simpul-simpul V yang masing-masing bernomor 1 s.d. N (sebanyak N buah). Misalkan pula terdapat suatu fungsi `shortestPath(i, j, k)` yang mengembalikan kemungkinan jalur terpendek dari i ke j dengan hanya memanfaatkan simpul 1 s.d. k sebagai titik perantara.

Tujuan akhir penggunaan fungsi ini adalah untuk mencari jalur terpendek dari setiap simpul i ke simpul j dengan perantara simpul 1 s.d. k+1.

Ada dua kemungkinan yang terjadi:
1. Jalur terpendek yang sebenarnya hanya berasal dari simpul-simpul yang berada antara 1 hingga k.
2. Ada sebagian jalur yang berasal dari simpul-simpul i s.d. k+1, dan juga dari k+1 hingga j

Perlu diketahui bahwa jalur terpendek dari i ke j yang hanya melewati simpul 1 s.d. k telah didefinisikan pada fungsi shortestPath(i, j, k) dan telah jelas bahwa jika ada solusi dari i s.d. k+1 hingga j, maka panjang dari solusi tadi adalah jumlah (konkatenasi) dari jalur terpendek dari i s.d. k+1 (yang melewati simpul-simpul 1 s.d. k), dan jalur terpendek dari k+1 s.d. j (juga menggunakan simpul-simpul dari 1 s.d. k).

Maka dari itu, rumus untuk fungsi `shortestPath(i, j, k)` bisa ditulis sebagai suatu notasi rekursif sbb.:

Basis-0
```
shortestPath(i, j, 0) = edgeCost(i, j);
```

Rekurens
```
shortestPath(i, j, k) = min(shortestPath(i, j, k-1),
 shortestPath(i, k, k-1) +
 shortestPath(k, j, k-1));
```

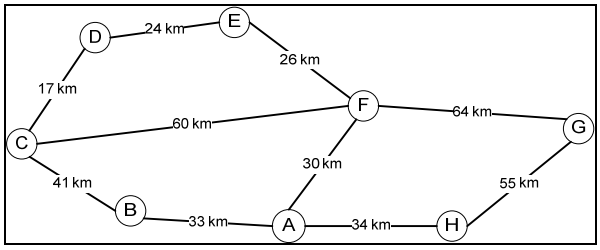
Rumus ini adalah inti dari algoritma Floyd-Warshall. Algoritma ini bekerja dengan menghitung shortestPath(i,j,1) untuk semua pasangan (i,j), kemudian hasil tersebut akan digunakan untuk menghitung shortestPath(i,j,2) untuk semua pasangan (i,j), dst. Proses ini akan terus berlangsung hingga k = n dan kita telah menemukan jalur terpendek untuk semua pasangan (i,j) menggunakan simpul-simpul perantara.

2.2.3 Pseudokode

```
//Asumsikan bahwa terdapat fungsi edgeCost(i,j) yang mengembalikan biaya (cost) di ujung dari i ke j (tak hingga jika tidak ada)
//Juga asumsikan bahwa n adalah jumlah simpul dan edgeCost(i,i)=0

path = array of integer;

//Matriks dua dimensi. Pada setiap langkah di algoritma, path[i][j] adalah jalur terpendek dari i ke j memanfaatkan nilai perantara pada (1..k-1). Setiap path[i][j] diinisialisasi ke edgeCost(i,j);

procedure FloydWarshall()
k traversal [1..n]
    foreach (i,j) pada [1..n]
        path[i][j] = min(path[i][j],
           path[i][k]+path[k][j]);
    {end foreach}
{end traversal}
```

3. STUDI KASUS

Misalkan terdapat suatu graf berbobot yang merepresentasikan kondisi keterhubungan antarkota di suatu daerah, dengan ilustrasi sebagai berikut.



Gambar 1. Representasi keterhubungan antarkota dalam graf berbobot

Misalkan seseorang akan melakukan perjalanan dari kota A ke kota C. Orang tersebut mencoba untuk menerapkan algoritma Dijkstra dan algoritma Floyd-Warshall untuk mencari jalur terpendek dari kota A ke kota C.

Berikut ini adalah penelusuran jalur apabila orang tersebut menggunakan algoritma Dijkstra (prinsip *greedy*):

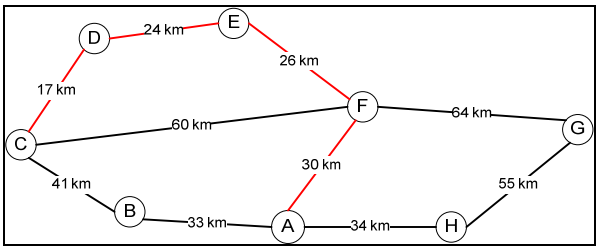
Tahap 1:
Dari kota A, orang tersebut akan memilih kota F dengan bobot minimum dari kota A (30 km).

Tahap 2:
Dari kota F, orang tersebut kemudian memilih kota E yang memiliki bobot minimum dari kota F (26 km).

Tahap 3:
Dari kota E, orang tersebut akan melanjutkan perjalanan ke kota D (satu-satunya simpul yang terhubung)

Tahap 4:
Dari kota D, orang tersebut lalu melanjutkan perjalanan dan sampai ke kota C.

Total jarak yang ditempuh oleh orang tersebut adalah = **97** km dengan jalur (A – F – E – D – C). Dalam representasi graf, warna merah pada sisi graf menunjuk ke jalur terpendek menurut algoritma Dijkstra.



Gambar 2. Representasi keterhubungan antarkota setelah menerapkan algoritma Dijkstra

Sekarang, orang tersebut mencoba menerapkan algoritma Floyd-Warshall dengan pendekatan pemrograman dinamis maju (*forward*).

Basis

$$f_1(s) = cx_1s$$

Rekurens

$$f_k(s) = \min x_k \{cx_ks + f_{k-1}(x_k)\},\ k = 2, 3, 4$$

Tahap 1:

$$f_1(s) = cx_1s$$

s	Solusi optimum	
	$f_1(s)$	x_1
B	33	A
F	30	A
H	34	A

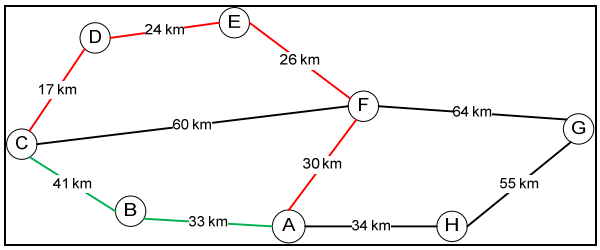
Tahap 2:

$$f_2(s) = \min s_2 \{cx_2s + f_1(x_2)\}$$

x_2	$f_2(x_2, s) = cx_2s + f_1(x_2)$			Solusi optimum	
s	B	F	H	$f_2(s)$	x_2
C	**74**	90	∞	**74**	B
E	∞	56	∞	26	F
G	∞	94	89	89	H

Dari hasil pencarian jalur terpendek dari A ke C menggunakan algoritma Floyd-Warshall (pemrograman dinamis), ditemukan bahwa jarak terpendek dari A ke C adalah **74** km dengan jalur (A – B – C).

Berikut ini representasi graf setelah menggunakan kedua algoritma tadi. Dalam representasi graf, warna merah pada sisi graf menunjuk ke jalur terpendek menurut algoritma Dijkstra, sementara warna hijau menurut algoritma Floyd-Warshall.



Gambar 3. Representasi keterhubungan antarkota setelah menerapkan algoritma Dijkstra (sisi berwarna merah), dan algoritma Floyd-Warshall (sisi berwarna hijau)

Terdapat perbedaan yang cukup signifikan untuk perbedaan penerapan kedua algoritma tadi (ca. 23 km), ini berarti algoritma Dijkstra gagal memberi solusi optimum untuk kasus di atas.

4. KESIMPULAN

1. Algoritma Dijkstra yang menerapkan prinsip *greedy* tidak selalu berhasil memberikan solusi optimum untuk kasus penentuan lintasan terpendek (*single pair shortest path*).

2. Algoritma Floyd-Warshall yang menerapkan pemrograman dinamis lebih menjamin keberhasilan penemuan solusi optimum untuk kasus penentuan lintasan terpendek (*single pair shortest path*).

REFERENSI

[1] Munir, Rinaldi, "Diktat Kuliah IF2251 Strategi Algoritmik". Program Studi Teknik Informatika, STEI ITB, 2007.

[2] en.wikipedia.org/wiki/Dijkstra_algorithm.htm
Tanggal akses: 22 Mei 2007, pukul 17.59

[3] en.wikipedia.org/wiki/Dynamic_programming.htm
Tanggal akses: 22 Mei 2007, pukul 17.34

[4] en.wikipedia.org/wiki/Floyd-Warshall_algorithm.htm
Tanggal akses: 22 Mei 2007, pukul 17.29

[5] www.cprogramming.com/computersciencetheory/dp.htm
Tanggal akses: 21 Mei 2007, pukul 14.48

[6] www.cprogramming.com/computersciencetheory/dijks.htm
Tanggal akses: 21 Mei 2007, pukul 14.47