ALGORITMA
MENCARI LINTASAN TERPENDEK

Indra Fajar[1], Gustian Siregar[2], Dede Tarwidi[3]

Laboratorium Ilmu dan Rekayasa Komputasi
Departemen Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : fs_indra@yahoo.com[1], guzztian@yahoo.com[2], cyberdeta@yahoo.com[3]

Abstrak

Pencarian lintasan terpendek menjadi hal yang sangat diperlukan. Dalam kehidupan sehari-hari, pencarian jalur terpendek digunakan misalkan oleh seorang pengendara kendaraan bermotor, perutean pada jaringan komputer. Para pengendara kendaraan bermotor mencari lintasan terpendek antara satu tempat ke tempat lain, perutean pada jaringan komputer mencari delai yang paling minimum untuk mengirimkan data. Terdapat berbagai algoritma yang membahas pencarian lintasan terpendek, antara lain algortima *exhaustive search*, algoritma *Greedy*, program dinamis. Algoritma *exhaustive search* mencari lintasan terpendek dengan mengenumerasi semua rangkaian keputusan yang mungkin dari lintasan yang ada dan memilih rangkaian keputusan yang terbaik. Algoritma *Djikstra* dengan menggunakan prinsip *Greedy* mencari lintasan terpendek dari satu simpul ke semua simpul lain yang terhubung. Sedangkan program dinamis yaitu dengan mengurangi pengenumerasian dengan prinsip optimalitas.

***Kata kunci**: lintasan terpendek, algoritma exhaustive search, algoritma Greedy, algoritma Djikstra, program dinamis.*

1. Pendahuluan

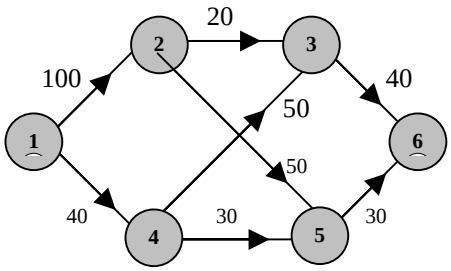
Seiring dengan perkembangan teknologi yang makin pesat termasuk dalam aplikasi dari algoritma pencarian lintasan terpendek, maka sangat penting untuk memilih algoritma yang tepat agar diperoleh hasil yang optimal. Ada banyak pilihan algoritma untuk mencari lintasan terpendek diantaranya dengan *algoritma Exhaustive search, algoritma Djikstra*, dan *Program Dinamis.* Masing-masing algoritma mempunyai keuntungan dan kelemahan tergantung dari persoalan yang dihadapi.

2. Pencarian Lintasan Terpendek

2.1 Algoritma *Exhaustive search*

Mencari lintasan terpendek dengan *exhaustive search* yaitu dengan mengenumerasi setiap lintasan yang mungkin dengan cara yang sistematis. Dari setiap kemungkinan tersebut dievaluasi satu persatu, selanjutnya bandingkan setiap lintasan yang telah dievaluasi, lintasan yang memberikan nilai terkecil merupakan lintasan terpendek yang kita cari.

Contoh pemecahan masalah dari graf :



Gambar 1 Graf berarah dengan 5 buah simpul

Misalkan akan dicari lintasan terpendek dari simpul 1 ke simpul 6. Banyaknya kemungkinan lintasan dari simpul 1 ke simpul 6 ada 2x2x1=4 kemungkinan lintasan. Kemudian enumerasi dan evaluasi setiap kemungkinan tersebut.

No.	Lintasan	Jarak	Keterangan
1.	1-2-3-6	160	-
2.	1-2-5-6	180	-
3.	1-4-3-6	130	-
4.	1-4-5-6	100	Solusi

Tabel 1. Pencarian lintasan terpendek dengan algoritma exhaustive search.

Algoritma ini akan memberikan solusi yang paling optimal untuk semua kasus graf dengan bobot yang tidak negatif. Akan tetapi, untuk jumlah simpul yang

banyak algoritma ini tidak mangkus karena membutuhkan waktu komputasi yang lama.

2.2 Algoritma *Dijkstra* dengan Menggunakan Prinsip *Greedy*

Secara harfiah g*reedy* artinya tamak atau rakus, secara bahasa, tamak dan rakus mengandung pengertian yang negatif, yaitu mengambil semua kemungkinan yang ada tanpa memikirkan kosekuensi ke depan. Prinsip dari *greedy* adalah ambil apa yang diperoleh sekarang. Prinsip ini digunakan untuk memecahkan solusi optimum tapi dalam konteks yang baik.

Algoritma *Greedy* menyelesaikan masalah langkah per langkah. Dari berbagai langkah, diambil keputusan yang terbaik dalam menentukah langkah mana yang terbaik.

Salah satu algoritma mencari lintasan terpendek yang paling terkenal adalah algoritma *Djikstra* yang ditemukan oleh matematikawan Belanda, Edsger Wybe *Djikstra* pada tahun 1959.

Algoritma *Djikstra* menggunakan prinsip *greedy* dalam menentukan lintasan terpendeknya. Pada setiap langkah, ambil sisi yang 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 terpendek diantara semua lintasannya yang belum terpilih.

Berikut adalah langkah-langkah pencarian lintasan terpendek menggunakan algoritma *Djikstra* :

1. Graf yang dibuat direpresentasikan dalam matriks ketetanggaan M=[m_{ij}].
   m_{ij} = bobot sisi (i,j)
   m_{ii} = 0
   m_{ij} = ∞, bila tidak ada lintasan dari simpul i ke simpul j.
2. Dibuat tabel S = [s_i]
   s_i = 1, jika simpul i termasuk lintasan terpendek.
   s_i = 0, jika simpul i tidak termasuk dalam lintasan terpendek.
3. Dibuat tabel keluaran D = [d_i], yaitu jarak dari simpul awal ke simpul tujuan.

Skema umum algoritma *Djikstra* :

```
Procedure Djikstra (G: graf berbobot terhubung sederhana, dengan semua bobot positif)
{G mempunyai simpul a= v0, v1, ...vn=z dan bobot w(vi,vj), dimana w(vi,vj) =∞ jika { vi,vj} bukan merupakan sisi di G}
for i:= 1 to n
     L(vi):= ∞
L(a):= 0
S:= ∅
{Setiap simpul diinisialisasi sehingga simpul simpul a bernilai nol dan semua simpul lain bernilai ∞, dan S adalah himpunan kosong}
while z∉S
   begin
      u:= sebuah simpul bukan anggota S dengan L(u) minimum.
      S:= S∪{u}
      for semua simpul v yang bukan anggota S
         If L(u)+w(u,v) < L(v) then
            L(v):= L(u)+w(u,v)
      {menambahkan sebuah simpul ke S yang mempunyai jumlah bobot minimum, dan memperbarui simpul-simpul yang bukan di S}.
   end.      {L(z)=panjang lintasan terpendek dari simpul a ke simpul z}.
```

Contoh lihat graf pada gambar 1

1. Matriks ketetanggaan M dari graf tersebut :

	1	2	3	4	5	6
1	0	100	∞	40	∞	∞
2	∞	0	20	∞	50	∞
3	∞	∞	0	∞	∞	40
4	∞	∞	50	0	∞	∞
5	∞	∞	∞	30	0	30
6	∞	∞	∞	∞	∞	0

Tabel 1 Matriks ketetanggaan

2. Lintasan terpendek dari simpul 1 ke semua simpul dalam graf

Simpul asal	Simpul Tujuan	Lintasan Terpendek	Jarak
1	2	1-2	100
1	3	1-4-3	90
1	4	1-4	70
1	5	1-4-5	40
1	6	1-4-5-6	100

Tabel 2 Lintasan terpendek

2.3 Program Dinamis

Untuk menentukan semua pasangan lintasan terpendek dalam graf berbobot dapat dipakai program dinamis. Misalkan G=(V,E) adalah graf dengan n buah simpul, dan C adalah bobot untuk G sedemikian sehingga C(i,i)=0, $1 \le i \le n$, dan C(i,j) adalah bobot dari sisi <i,j> jika <i,j> ∈E(G) dan C(i,j)= ∞jika i≠j dan <i,j>∉ E(G).

Masalah pasangan lintasan terpendek adalah menentukan matriks A sedemikian sehingga A(i,j) adalah panjang lintasan terpendek dari i ke j. Definisikan A^k(i,j) adalah panjang lintasan terpendek dari i ke j yang tidak melalui simpul yang lebih besar dari k. A(i,j) dapat diperoleh :

$$A(i,j) = \min\{\min_{1\le k\le n}\{A^{k-1}(i,k) + A^{k-1}(k,j)\}, C(i,j)\} \quad (1)$$

Oleh karena itu $A^0(i,j) = C(i,j)$, $1 \le i \le n$, $1 \le j \le n$.
Rumus (1) dapat dituliskan :

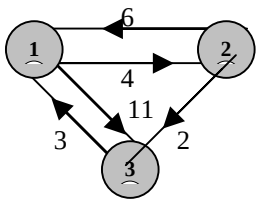
$$A^k(i,j) = \min\{A^{k-1}(i,j), A^{k-1}(i,k) + A^{k-1}(k,j)\}, \quad k \ge 1 \ldots (2)$$

Berikut ini adalah prosedur untuk memperoleh semua pasangan lintasan terpendek

```
Procedure Semua_Lintasan(COST,A,n)
{COST(n,n) adalah bobot dari graf dengan n buah simpul, A(i,j) adalah bobot lintasan terpendek dari simpul i ke simpul j}
{COST(i,i)=0, 1≤i≤n}

for i:=1 to n do
     for j:=1 to n do
         A(i,j):=COST(i,j)
     end
end
for k:=1 to n do
  for i:=1 to n do
        {untuk setiap pasangan simpul yang mungkin}
    for j:=1 to n do
      A(i,j):=min{A(i,j),A(i,k)+A(k,j)}
    end
  end
end
end.
```

Contoh mencari semua pasangan lintasan terpendek dengan menggunakan program dinamis dari graf :



Gambar 2 Graf berarah dengan 3 buah simpul

Tabel yang dibangkitkan berdasarkan program dinamis :

$A^{(0)}$	1	2	3
1	0	2	11
2	6	0	2
3	3	∞	0

(a)

$A^{(1)}$	1	2	3
1	0	2	11
2	6	0	2
3	3	7	0

(b)

$A^{(2)}$	1	2	3
1	0	2	11
2	6	0	2
3	3	7	0

(c)

$A^{(3)}$	1	2	3
1	0	2	11
2	6	0	2
3	3	∞	0

(d)

Tabel 3 Pembangkitan lintasan terpendek dengan program dinamis.

3. Perbandingan Algoritma untuk Mencari Lintasan Terpendek

Berdasarkan penjelasan pada pembahasan sebelumnya maka dapat dibandingkan dari ketiga algoritma pencari lintasan terpendek sebagai berikut:
:

3.1 Algoritma *Brute-force*

Algoritma *brute-force* sebagai pencari lintasan terpendek hanya cocok untuk jumlah simpul yang sedikit saja. Untuk jumlah simpul yang banyak akan dibutuhkan waktu komputasi yang lama.

Keadaan paling buruk dari suatu graf jika suatu simpul mempunyai sisi ke semua simpul lain dalam graf tersebut, maka banyaknya kemungkinan lintasan dari satu simpul ke satu simpul lain pada graf tersebut adalah

$$P(n-2,n-2) + P(n-2,n-3) + P(n-2,n-4) + \ldots P(n-2,0)$$
$$= \frac{(n-2)!}{0!} + \frac{(n-2)!}{1!} + \frac{(n-2)!}{2!} + \frac{(n-2)!}{3!} + \ldots + 1 < n!$$

P(n,k) = Permutasi dari n-buah diambil k-buah.

Sedangkan waktu untuk menghitung bobot setiap lintasan adalah n. Sehingga total waktu komputasi yang dibutuhkan dari algoritma *brute-force* tersebut adalah $T(n) = O(n \cdot n!)$.

3.2 Algoritma *Dijkstra*

a. Instruksi dalam kalang for pertama
```
for i:= 1 to n ...
```
Kompleksitas algoritmanya O(n).
b.kalang bersarang
kalang `while z∉S...` dan kalang for semua simpul v... kompleksitas algoritmanya $O(n^2)$.
Sehingga kompleksitas algoritma *Dijkstra* adalah $O(n)+O(n^2)= O(n^2)$.
Algoritma *Dijkstra* hanya mencari lintasan terpendek dari suatu simpul tertentu ke semua simpul lainnya.
Sehingga untuk mencari semua pasangan simpul terpendek, total waktu komputasinya adalah
$T(n)=n.O(n^2)=O(n^3)$.

3.3 Program Dinamis

a. Instruksi pada kalang **for** bersarang pertama :
```
for i:=1 to n do
     for j:=1 to n do
         ...
     end
end
```
Memberikan kompleksitas algortima $O(n^2)$.
b. Instruksi pada kalang **for** bersarang kedua :
```
for k:=1 to n do
   for i:=1 to n do
      ...
    for j:=1 to n do
        ...
    end
  end
end
```
Memberikan kompleksitas algoritma $O(n^3)$.

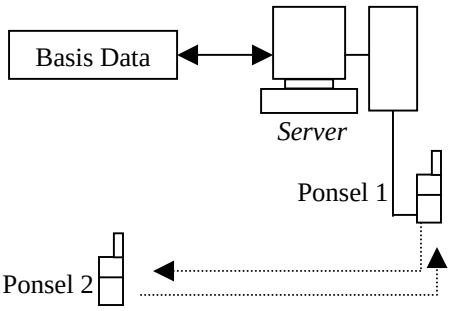
Sehingga total waktu komputasi (kompleksitas algoritmanya) adalah $O(n^2) + O(n^3) = O(n^3)$.

Dari ketiga algoritma pencari lintasan terpendek di atas, algoritma-algoritma tersebut mempunyai kekurangan dan kelebihan masing-masing.
a. Algoritma *brute-force* selalu menghasilkan solusi yang optimal. Akan tetapi, algoritma ini hanya cocok untuk simpul yang sedikit, karena waktu komputasinya yang besar yaitu O(n.n!).
b. Algoritma *Dijkstra* sama halnya dengan algoritma *brute-force* selalu menghasilkan solusi optimal. Yang membedakannya adalah bahwa algoritma ini waktu komputasinya lebih kecil yaitu $O(n^2)$. Sehinnga untuk mencari lintasan terpendek dari semua simpul total waktu komputasinya adalah $n.O(n^2)=O(n^3)$.
c. Program dinamis selalu menghasilkan solusi yang optimal dengan waktu komputasi $O(n^3)$. Jika pada algortima *Dijkstra* hanya mencari lintasan terpendek dari satu ke semua simpul lain, maka pada program dinamis semua pasangan lintasan terpendek dicari.

4. Aplikasi Algortima Mencari Lintasan Tercepat Menggunakan Teknologi *Mobile*

Algoritma pencari lintasan terpendek dapat diaplikasikan dalam kehidupan sehari-hari. Salah satu aplikasi yang menguntungkan adalah mencari lintasan terpendek dan tercepat antar kota dalam suatu pulau.

Permasalahan tersebut dapat dimodelkan sebagai berikut :



Gambar 3 Skema pemrosesann lintasan terpendek dengan teknologi *mobile*

Misalkan kota-kota tersebut berada di Pulau Jawa. Dalam basis data terdapat jarak dan waktu tempuh dari kota-kota tersebut. Di dalam *server* berisi program pencari lintasan terpendek dengan menggunakan salah satu dari ketiga jenis algoritma yang telah dijelaskan sebelumnya, misalkan algoritma *Dijkstra. Server* juga terhubung dengan sebuah ponsel (ponsel 1 pada gambar 3) untuk menerima permintaan dari pengguna (ponsel 2) kemudian mengirimkan data berupa lintasan tercepat yang didapat dengan menggunakan algoritma *Dijkstra.*

Teknologi yang digunakan di sini adalah teknologi SMS *(Short Message Service).* Ponsel 2 mengirimkan permintaan berupa SMS yang berisi kota asal dan kota tujuan ke ponsel 1. Ponsel 1 akan mengirimkan data ke *server*, kemudian *server* akan mengambil data yang berhubungan dengan permintaan, berupa semua kota yang terhubung dengan kota asal dan kota tujuan yang diminta. Data yang telah didapatkan akan diolah oleh *server* dengan algoritma *Dijkstra. Server* melalui ponsel 1 mengirimkan data keluaran berupa lintasan tercepat kota yang diminta ke ponsel 2 .

Basis data yang telah ada setiap saat dapat diubah sesuai dengan kondisi yang ada. Misalkan terdapat jalur baru yang menghubungkan dua buah kota, waktu tempuh yang mengalami perubahan, dan jarak. Sehingga dengan basis data yang dinamis dihasilkan keluaran yang lebih akurat.

Aplikasi ini sangat cocok dipakai oleh para pengemudi yang akan bepergian dari satu kota ke

kota lain, sehingga efektivitas baik dari segi waktu maupun biaya perjalanan akan lebih optimal.

5. Kesimpulan

Penggunaan ketiga algoritma yang telah dijelaskan sebelumnya, yaitu algoritma *brute-force*, algoritma *Dijkstra*, dan program dinamis disesuaikan dengan keadaan. Misalkan jumlah simpul yang sedikit walaupun dengan ketiga algortima dapat dipakai, akan tetapi lebih baik digunakan algoritma *brute-force*, karena algoritmanya mudah dibuat. Jika suatu graf mempunyai simpul yang banyak, maka algoritma *brute-force* tidak cocok digunakan karena memerlukan waktu komputasi yang lama, sehingga algoritma *Dijkstra* dan program dinamis dapat dipergunakan dengan waktu komputasi yang sama yaitu $O(n^3)$.

Banyak sekali aplikasi dari ketiga algoritma ini yang sangat dibutuhkan dalam kehidupan sehari-hari, antara lain mencari jarak tercepat dari satu kota ke kota lain, perutean pada jaringan komputer, dll.

Daftar Pustaka

[1] http://www.cs.usak.ca/resources/tutorial. Diakses tanggal 18 Mei 2005.

[2] Horowitz, Ellis and Sahni, Sartaj. *Fundamentals of Computer Algorithms.* Computer Science Press, Inc. USA: 1978.

[3] Munir, Rinaldi. *Diktat Kuliah IF 2251 Strategi Algoritmik.* Laboraorium Ilmu dan Rekayasa Komputasi Departemen Teknik Informatika ITB. Bandung: 2005.

[4] Rosen, H. Kenneth. *Discrete Mathematics and its Applications.* McGraw-Hill. USA: 1999.

[5] Tugas Akhir Ika Swastika (10100063) *Pencarian Jalur Tercepat pada Rute antar Kota di Pulau Jawa dengan Teknologi Mobile.*