Algoritma *Greedy* untuk Menentukan Lintasan Terpendek

Irvan Prama Defindal[1], Boyke Ariesanda[2], Christoforus[3]

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

E-mail: if13018@students.if.itb.ac.id[1], if13036@students.if.itb.ac.id[2],
If13094@students.if.itb.ac.id[3]

Abstrak

Kemacetan yang sering terjadi selama perjalanan, sering mengganggu kegiatan kita sehari-hari. Setiap manusia ingin sampai ke tujuan dengan tepat waktu. Tetapi, sering kali kemacetan menyebabkan keinginan manusia terganggu. Oleh karena itu, dibutuhkan suatu cara untuk menanggulangi gangguan tersebut. Untuk mencapai suatu tempat dengan waktu yang lebih cepat, kita akan mencari lintasan terpendek dari tempat asal ke tempat tujuan. Lintasan terpendek ini juga memperhitungkan waktu-waktu dimana kemacetan sering terjadi. Contoh lintasan terpendek adalah: ketika kita ingin ke gedung olahraga (GOR) Trilomba di Jl. Padjajaran pukul 07.00 hari senin. Kita bisa melewati Cihampelas, menuju Jl. dr. Otten dan memotong ke Jl. Padjajaran tanpa harus melewati kemacetan yang terjadi di Wastu Kencana. Tentu saja ini akan menghemat waktu kita. Pada makalah ini akan dibahas pencarian lintasan terpendek berdasarkan jarak dan waktu-waktu terjadinya kemacetan.

Kata kunci: algoritma *greedy*, lintasan terpendek, graf berbobot

1. Pendahuluan

Kebutuhan akan informasi yang akurat dibutuhkan saat ini. Perkembangan teknologi sangat pesat dalam beberapa tahun ini, sehingga kebutuhan manusia semakin meningkat. Sehingga dibutuhkan waktu yang cepat untuk mencapai kebutuhan tersebut. Algoritma *Greedy* merupakan suatu program yang tepat untuk meningkatkan efisiensi waktu.

2. Algoritma *Greedy*

2.1 Definisi Algoritma *Greedy*

Algoritma *greedy* adalah algoritma yang memecahkan masalah langkah demi langkah, pada setiap langkah:
1. mengambil pilihan yang terbaik yang dapat diperoleh saat itu
2. berharap bahwa dengan memilih optimum loklal pada setiap langkah akan mencapai optimum global.
Algoritma *greedy* mengasumsikan bahwa optimum lokal merupakan bagian dari optimum global.

Prinsip algoritma *greedy* adalah: "*take what you can get now!"*. Ambil apa yang anda peroleh sekarang! Prinsip ini juga dipakai dalam pemecahan masalah optimasi. Dalam kehidupan sehari-hari, kita juga pernah menggunakan prinsip *greedy*, misalnya:
a. Memilih jurusan di Perguruan Tinggi
b. Memilih jalur tersingkat dari Bandung ke Jakarta.

2.2 Skema Umum Algoritma *Greedy*

Persoalan optimasi dalam konteks algoritma *greedy* disusun oleh elemen-elemen sebagai berikut:
1. Himpunan kandidat, *C*.
Himpunan ini berisi elemen-elemen pembentuk solusi. Pada setiap langkah, satu buah kandidat diambil dari himpunannya.

2. Himpunan solusi, *S*.
Merupakan himpunan dari kandidat-kandidat yang terpilih sebagai solusi persoalan. Himpunan solusi adalah himpunan bagian dari himpunan kandidat.
3. Fungsi seleksi – dinyatakan sebagai predikat SELEKSI – merupakan fungsi yang pada setiap langkah memilih kandidat yang paling mungkin untuk mendapatkan solusi optimal. Kandidat yang sudah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya.

4. Fungsi kelayakan (*feasible*) – dinyatakan dengan predikat LAYAK – merupakan fungsi yang memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendaara yang ada.

5. Fungsi obyektif, merupakan fungsi yang memaksimumkan atau meminimumkan nilai solusi.

Kita berharap optimum global merupakan solusi optimum dari persoalan. Namun, adakalanya

optimum global belum tentu merupakan solusi optimum (terbaik), tetapi dapat merupakan solusi sub-optimum atau pseudo-optimum. Hal ini dapat dijelaskan dari dua faktor berikut:
1. algoritma *greedy* tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada.
2. pemilihan fungsi SELEKSI: fungsi SELEKSI biasanya didasarkan pada fungsi obyektif (fungsi SELEKSI bisa saja identik dengan fungsi obyektif). Jika fungsi SELEKSI tidak identik dengan fungsi obyektif, kita harus memilih fungsi yang tepat untuk menghasilkan nilai yang optimum.

Karena itu, pada sebagian masalah algoritma *greedy* tidak selalu berhasil memberikan solusi yang benar-benar optimum. Tetapi, algoritma *greedy* pasti memberikan solusi yang mendekati (*approximation*) nilai optimum.

3. Lintasan Terpendek (*Shortest Path*)

3.1 Definisi Lintasan Terpendek

Lintasan terperndek adalah lintasan minimum yang diperlukan untuk mencapai suatu tempat dari tempat tertentu. Lintasan minimum yang dimaksud dapat dicari dengan menggunakan graf. Graf yang digunakan adalah graf yang berbobot, yaitu graf yang setiap sisinya diberikan suatu nilai atau bobot. Dalam kasus ini, bobot yang dimaksud berupa jarak dan waktu kemacetan terjadi.

3.2 *Single-source shortest path*

Ada beberapa macam persoalan lintasan terpendek, antara lain:
a. Lintasan terpendek antara dua buah simpul tertentu (*a pair shortets path*).
b. Lintasan terpendek antara semua pasangan simpul (*all pairs shortest path*).
c. Lintasan terpendek dari simpul tertentu ke semua simpul yang lain (*single-source shoertest path*).
d. Lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu (*intermediate shortest path*).

Dalam makalah ini, persoalan yang digunakan adalah *single-source shortest path*. Diberikan sebuah persoalan:
"Diberikan sebuah graf berbobot G(V, *E*). Tentukan lintasan terpendek dari simpul awal, *a*, ke setiap simpul lainnya di *G*. Asumsi bahwa bobot semua sisi bernilai positif."

Algoritma *greedy* untuk mencari lintasan terpendek dapat dirumuskan sebagai berikut:
1. Perikasa semua sisi yang langsung bersisian dengan simpul *a*. Pilih sisi yang bobotnya terkecil. Sisi ini menjadi lintasan terpendek pertama, sebut saja L(1).
2. Tentukan lintasan terpendek kedua dengan cara berikut:

> (i) hitung: $d(i)$ = panjang $L(1)$ + bobot sisi dari simpul akhir $L(1)$ ke simpul i yang lain
> (ii) pilih $d(i)$ yang terkecil
> Bandingkan $d(i)$ dengan bobot sisi (a, i). Jika bobot sisi (a, i) lebih kecil daripada $d(i)$, maka
> $L(2) = L(1)$ U (sisi dari simpul akhir $L(i)$ ke simpul i)

3. Dengan cara yang sama, ulangi langkah 2 untuk menentukan lintasan terpendek berikutnya.

4. Strategi *Greedy* pada Algoritma Dijkstra

4.1 Algoritma Dijkstra

Algoritma Dijkstra ditemukan oleh Edger Wybe Dijkstra. Algoritma ini merupakan algoritma yang paling terkenal untuk mencari lintasan terpendek. Algoritma Dijkstra diterapkan pada graf berarah, tetapi selalu benar untuk graf tak-berarah.

Algoritma ini menggunakan strategi *Greedy* sebagai berikut:

> "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 yang terpendek diantara semua lintasannya ke simpul-simpul yang belum terpilih."

Langkah-langkah algoritma Dijkstra yang lebih jelas dapat dilihat di: http://kur2003.if.itb.ac.id/strategialgoritmik/rencana/algoritmagreedy.

5. Analisis Algoritma Dijkstra untuk Mencari Lintasan Terpendek.

Analisis yang dilakukan meliputi 2 aspek, yaitu:
1. berdasarkan graf berbobot yang bobotnya merupakan panjang lintasan
2. berdasarkan graf berbobot yang bobotnya merupakan waktu terjadinya kemacetan.

Tabel 1. Panjang lintasan dari satu tempat ke tempat yang lainnya.

	1	2	3	4
1	0	∞	3500 m	500 m
2	7000 m	0	4000 m	∞
3	∞	4000 m	0	3000 m
4	500 m	∞	4500 m	0

Keterangan:
1 = Dago
2 = Pasteur
3 = Cihampelas
4 = ITB

Contoh persoalan: carilah lintasan terpendek dari ITB menuju Pasteur!

Jadi, lintasan terpendek dari ITB menuju Pasteur adalah: 4 – 3 – 2, yaitu sepanjang:

3000 m + 4000 m = 7000 m

Tabel 2. Waktu kemacetan sering terjadi.

	1	2	3	4
1	0	siang, sore	siang	∞
2	pagi, sore	0	sore	sore
3	sore, malam	sore	0	siang, sore
4	∞	sore	pagi, sore	0

Keterangan sama dengan tabel 1.

Jarak yang digunakan dari satu tempat ke tempat yang lainnya sama dengan jarak yang digunakan pada tabel 1.

Contoh persoalan: carilah lintasan terpendek dari ITB menuju Pasteur pada siang hari!

Jadi, lintasan terpendek dari ITB menuju Pasteur adalah: 4 – 1 – 2, yaitu sepanjang:

500 m + 7000 m = 7500 m

Seharusnya, lintasan terpendek dari ITB menuju Pasteur adalah 7000 m, tetapi pada siang hari terjadi kemacetan di Cihampelas, maka mengambil jalan yang tidak macet, yaitu melewati Dago.

6. Kesimpulan

Pencarian lintasan terpendek sangat berguna untuk menentukan jalan tersingkat untuk menuju suatu tempat. Sehingga, kita dapat sampai tepat waktu menuju tempat tujuan.
Hasil analisis berdasarkan bobot-bobot yang berbeda, menunjukkan bahwa semakin banyak bobot yang diberikan, maka semakin akurat pula data yang dihasilkan. Sehingga menghasilkan waktu yang efisien.

Daftar Pustaka

1. Algoritma *Greedy*. *http://kur2003.if.itb.ac.id/strategialgoritmik*. Diakses tanggal 16 Mei 2005 pukul 19.00 WIB.

2. R. Munir. , *Diktat Kuliah IF2251 Strategi Algoritmik*, Bandung, 2005.

3. Mehran S. , *Exhaustive Recursion and Backtracking,* April 2005.