Jurnal Penelitian Sains ; hal 23 - 28
No. 11. April 2002 ISSN: 1410-7058

ANALISIS ALGORITMA KRUSKAL GREADY DAN FASE PEMILIHAN BUSUR DALAM MENENTUKAN POHON RENTANGAN MINUMUM

Eddy Roflin
Jurusan Matematika FMIPA Universitas Sriwijaya

ABSTRAK

Algoritma Kruskal Gready merupakan salah satu algoritma untuk mendapatkan bobot pohon rentangan minimal yang dibentuk dari suatu graf terhubung terboboti. Algoritma ini mempunyai kompleksitas $\Theta(n \log n)$,dengan n adalah jumlah item yang diproses. Sebagai alternatif, algoritma Fase Pemilihan Busur memiliki kompleksitas $\Theta(m \log n)$, dengan m adalah jumlah busur dan n adalah jumlah simpul pada graf. Untuk $n < m^2$, $\Theta(m \log n) = \Theta(n \log n)$. Dengan kata lain, pada kasus terburuk, kedua algoritma tersebut sama cepatnya.

PENDAHULUAN

Dalam kehidupan sehari-hari, kita sering dihadapkan pada situasi yang dapat digambarkan dengan graf. Misalnya hubungan antara kota-kota secara abstrak dapat digambarkan dengan graf. Kota-kota digambarkan dengan simpul (*vertexs*) dan jalan yang menghubungkan antara kota tersebut digambarkan dengan busur (*edges*). Pohon (*tree*) merupakan salah satu bentuk graf. Secara sederhana pohon adalah graf terhubung yang tidak mengandung putaran (*cycle*). Dari satu graf terhubung dapat dibuat banyak pohon. Pohon yang dibentuk dari satu graf dan mengandung semua simpul pada graf tersebut disebut pohon rentangan (*spanning tree*). Dari satu graf terhubung terboboti dapat dibentuk banyak pohon rentangan yang beberapa diantaranya memiliki bobot minimum.

Salah satu cara untuk menentukan pohon rentangan minimal (*minimum spanning tree*) adalah dengan menggunakan algoritma Kruskal Gready. Dimulai dengan mengurutkan busur-busur pada graf berdasarkan bobot setiap busur dalam urutan tak turun, selanjutnya dipilih busur-busur yang akan membentuk pohon rentangan minimal.

Permasalahan utama algoritma Kruskal Gready adalah proses pengurutan busur. Proses ini memerlukan langkah yang cukup banyak. Proses pengurutan yang paling baik memiliki kompleksitas $\Theta(n \log n)$ dengan n adalah ukuran item yang diproses (stinson, 1985). Ukuran kompleksitas

menunjukkan seberapa efisien suatu algoritma dapat menyelesaikan masalah untuk sejumlah item yang menjadi input algoritma tersebut. Permasalahan berikutnya dari algoritma Kruskal Gready adalah pemeriksaan adanya putaran. Pemeriksaan putaran mempunyai kompleksitas $\Theta(\log m)$ dengan m adalah jumlah item yang diperiksa (stinson, 1985).

Tujuan Penelitian

Penelitian ini bertujuan untuk mempelajari / menganalisis algoritma Kruskal Gready dan algoritma Fase Pemilihan Busur sebagai alternatif dalam menentukan pohon perentang minimal.

TINJAUAN PUSTAKA

Graf adalah himpunan busur dan simpul yang banyaknya berhingga dimana busur-busurnya menghubungkan sebagian atau keseluruhan pasangan dari simpul-simpulnya. Dengan demikian graf $G(S,B)$ terdiri atas himpunan simpul (*vertex*) yang dinyatakan dengan $S = \{s_1, s_2, s_3, ..., s_n\}$ dan himpunan busur (*edges*) yang dinyatakan dengan $B = \{b_1, b_2, b_3, ..., b_n\}$ dengan $b_i = (s_i, s_j)$ merupakan busur yang menghubungkan simpul s_i dan simpul s_j.

Busur $b_i = (s_i, s_j)$ disebut busur berarah jika terdapat aliran dari simpil s_i menuju ke simpul s_j. Jika tidak terdapat aliran dari simpul s_i ke simpul s_j, maka busur $b_i = (s_i, s_j)$ disebut busur tidak berarah. Graf yang semua busurnya berarah disebut graf berarah (*directed graph, digraph*) dan graf yang semua busurnya tidak berarah disebut graf tak berarah (*undirected graph*).

Graf yang busur-busurnya diberi bobot disebut graf terboboti (*weighted graph*) dinotasikan dengan $G(S,B,W)$. Pembobot suatu busur biasanya berupa bilangan, atau besaran apapun yang ingin diberikan kepada busur.

Graf $G(S,B)$ disebut graf terhubung (*connected graph*) jika untuk sembarang dua simpul yang berbeda terdapat paling sedikit satu busur yang menghubungkan kedua simpul tersebut, dan disebut graf tak terhubung (*disconnected graph*) jika terdapat simpul yang tidak terhubung.

Suatu lintasan (*path*) adalah sebarisan (tidak kosong) busur-busur atau simpul-simpul atau kombinasi simpul dan busur. Suatu lintasan yang tertutup disebut rangkaian atau sirkuit.

Analisis Algoritma

Analisis algoritma dimaksudkan untuk mempelajari banyaknya langkah yang

dibutuhkan oleh suatu algoritma dalam menyelesaikan suatu masalah. Analisis algoritma dengan melibatkan berbagai variasi input program akan memberikan suatu pola waktu yang dibutuhkan oleh program tersebut dalam memecahkan suatu permasalahan.

Misalkan *T(n)* sebagai waktu yang dibutuhkan oleh suatu program dalam menyelesaikan suatu masalah yang berukuran n. Berkaitan dengan masalah pembentukan pohon rentangan minimal dari graf *G(S,B,W)*, ukuran masalah ditentukan oleh jumlah anggota himpunan *S* dan himpunan *B*.

Kompleksitas algoritma dapat menggambarkan kecepatan waktu *T(n)* terhadap perubahan *n*. Suatu algoritma lebih efisien dari algoritma lain, jika algoritma tersebut mempunyai waktu pertumbuhan *T(n)* yang lebih lambat dalam menyelesaikan suatu masalah tertentu.

Misal, didefinisikan suatu fungsi $f : Z^+ \rightarrow R$, dan ada *m* yang tergantung pada *f*, dengan *f(n) > 0*, untuk setiap *n > m*. Fungsi tersebut disebut sebagai fungsi waktu eksekusi (*running time function*). Misalkan *f* dan *g* fungsi waktu eksekusi. *f(n)* adalah $\Theta(g(n))$ jika terdapat bilangan riil positif c_1 dan c_2 serta bilangan bulat positif n_0 sehingga $0 \le c_1 g(n) \le f(n) \le c_2 g(n) \quad \forall n \ge n_0$.

Selanjutnya *f(n)* adalah $O(g(n))$ jika ada bilangan riil positif *c* dan bilangan bulat positif n_0 sehingga $0 \le f(n) \le cg(n) \quad \forall n \ge n_0$. Demikian pula *f(n)* adalah $\Omega(g(n))$ jika ada bilangan riil positif n_0 sehingga $0 \le cg(n) \le f(n) \quad \forall n \ge n_0$ [Cormen, 1994].

Misal didefinisikan suatu relasi terhadap himpunan semua fungsi waktu eksekusi. Jika *f(n)* adalah $\Theta(g(n))$, maka *f(n) ~ g(n)*. Dalam hal ini ~ menunjukkan relasi equivalensi. Setiap kelas equivalensi mengandung semua fungsi dengan kecepatan pertumbuhan yang spesifik. Dengan demikian, dapat dinyatakan bahwa fungsi *f* dan *g* mempunyai kecepatan pertumbuhan yang sama, jika dan hanya jika *f(n) ~ g(n)* [Biggs, 1989].

Salah satu relasi antara Θ, O, dan Ω : *f(n)* adalah $\Theta(g(n))$ jika dan hanya jika *f(n)* adalah $O(g(n))$ dan *f(n)* adalah $\Omega(g(n))$. Jika *f(n)* adalah $O(g(n))$, maka kecepatan pertumbuhan *f* tidak melebihi kecepatan pertumbuhan *g*. Jika *f(n)* adalah

$\Omega(g(n))$, maka kecepatan pertumbuhan f tidak lebih rendah dari g. Jika $f(n)$ adalah $\Theta(g(n))$ maka kecepatan pertumbuhan f sama dengan kecepatan pertumbuhan g. [cormen, 1994].

Algoritma Kruskal Gready

Misalkan $G(S,B)$ berupakan sebuah graf terhubung yang memiliki n simpul dan m busur. Langkah algoritma Kruskal Gready dalam menentukan pohon rentangan minimal adalah sebagai berikut : *Pertama*, mulai dengan graf T yang memiliki n simpul yang tidak terhubung. *Kedua*, urutkan busur-busur graf G dalam urutan yang tidak turun berdasarkan bobot busur. *Ketiga*, mulai dari urutan pertama, tambahkan busur tersebut pada T sehingga dengan penambahan busur baru tersebut tidak membentuk putaran. *Keempat*, ulangi langkah ketiga sampai $n-1$ busur ditambahkan pada T.

Proses utama algoritma Kruskal Gready adalah mengurutkan busur-busur berdasarkan bobot busur dan proses pemeriksaan putaran pada waktu memilih busur-busur yang akan menjadi busur pohon rentangan minimal. Algoritma Kruskal Gready memilih busur-busur dengan bobot minimal secara global, yang pada akhirnya melakukan proses pemeriksaan putaran pada setiap penambahan busur baru.

Sebuah himpunan yang terdiri atas n elemen dapat diurutkan dengan menggunakan fasilitas *merge sort* dalam waktu $\Theta(n \log n)$. Misalkan ditentukan waktu $a(n)$, dengan demikian waktu yang diperlukan algoritma Kruskal adalah $\Theta((n \log n)+(n-1)a(s))$. Jika didukung dengan data yang baik $a(s) = \Theta(\log n)$, sehingga waktu eksekusinya menjadi $\Theta(n \log n)$.

Algoritma Fase Pemilihan Busur

Misal, diberikan graf terhubung $G(S,B)$, dengan $S = \{s_1, s_2, s_3, ..., s_n\}$; dan $B = \{b_1, b_2, b_3, ..., b_m\}$. Akan dicari subgraf dari graf $G(S,B)$. Langkah *pertama*, daftarkan semua busur yang menghubungkan suatu simpul tertentu dengan simpul lainnya pada graf $G(S,B)$. Langkah *kedua*, berilah setiap busur dengan indeks. Indeks ini bersifat khas untuk setiap busur. Langkah *ketiga*, pilihlah untuk setiap simpul satu busur dengan bobot busur paling kecil terhadap busur-busur lain pada simpul tersebut atau busur yang bobotnya sama, pilihlah busur dengan indeks minimum. Dengan langkah tersebut akan dihasilkan suatu subgraf $G'(S,B')$, dengan S

$\{s_1, s_2, s_3, ..., s_n\}$; dan $B = \{b'_1, b'_2, b'_3, ..., b'_r\}$, untuk suatu r $(r < m)$.

Dengan menggunakan algoritma ini akan diperoleh paling sedikit [$n/2$] busur dan tidak akan menghasilkan putaran. Dengan demikian fase pemilihan busur tidak menjamin terbentuknya pohon, walaupun jumlah simpul yang terpilih sudah sama dengan jumlah simpul pada graf (n simpul), tetapi jumlah busur yang terpilih masih kurang dari $(n-1)$. Dengan demikian subgraf yang dihasilkan berupa graf yang tidak terhubung, terputus atas beberapa komponen. Oleh karena itu perlu dilakukan fase pemilihan busur kedua (fase pelengkap).

Misalkan graf $G(S,B)$ terdiri atas n simpul dan m busur. Karena setiap busur menghubungkan dua buah simpul, maka total proses pencarian busur pada fase pemilihan busur adalah $2m$. Selanjutnya, jika ada busur dengan bobot yang sama, maka yang dipilih adalah busur dengan indeks yang paling kecil. Misalkan diperlukan k kali pemeriksaan setiap busur, dengan k suatu konstanta, sehingga proses pemilihan busur memerlukan $2\ km$ langkah.

Langkah yang diperlukan sebelum melakukan fase pelengkap adalah sebagai berikut. *Pertama*, daftarkan semua subgraf yang dihasilkan sebagai simpul baru, misalkan k simpul. *Kedua*, hapus semua busur yang saling menghubungkan dua buah simpul pada subgraf yang sama. *Ketiga*, sisakan busur-busur yang menghubungkan subgraf yang satu dengan subgraf yang lain. *Keempat*, lakukan kembali fase pemilihan busur untuk k simpul. *Kelima*, ulangi langkah pertama selama jumlah busur yang terpilih masih kurang dari $(n-1)$. Permasalahan yang dihadapi dalam fase pelengkap adalah jumlah busur yang dilibatkan pada fase kedua, ketiga, dan seterusnya tidak dapat lagi diramalkan. Hal ini dikarenakan variasi busur pada suatu simpul tidak mengikuti aturan tertentu.

Misalkan, akan ditinjau kasus yang setiap dua simpulnya memiliki busur persekutuan yang berbobot minimum terhadap busur lain pada kedua simpul yang bersangkutan. Misalkan graf G(S,B) memiliki n simpul dan m busur. Pada fase pertama akan ditemukan [$n/2$] busur. Dengan demikian pada fase kedua jumlah busur yang terlibat maksimal $(m - [n/2])$ busur. Pada fase ketiga akan ditemukan minimum $[n/4]$ busur. Dengan demikian jumlah busur yang terlibat pada fase berikutnya maksimal $(m - [n/2] - [n/4])$ busur. Demikian seterusnya. Secara keseluruhan jumlah busur yang terlibat

pada proses pemilihan busur, jika ditinjau dari kasus terburuk, adalah :

$m + (m-[n/2]) + (m-[n/2] - [n/4]) + \ldots + (m-[n/2] - [n/4] - 1) = m + m([\log_2 (n)] - 1) - M < m \log_2 (n)$, untuk $n > 3$.

Dari analisis di atas, dapat disimpulkan bahwa dengan menggunakan fase pelengkap untuk melengkapi jumlah busur yang dipilih sebagai busur pohon rentangan minimum diperlukan waktu $\Theta(m \log n)$,

KESIMPULAN

Algoritma Fase Pemilihan Busur menunjukkan bahwa untuk mendapatkan minimum [$n/2$] busur yang merupakan busur pembentuk pohon rentangan minimal diperlukan *4 km* langkah. Selanjutnya dengan mengulangi fase pemilihan busur sebagai fase pelengkap telah pula menunjukkan bahwa jumlah busur yang telibat pada seluruh proses pemilihan busur-busur pembentuk pohon rentangan minimal, tidak lebih dari $(m \log_2 (n))$ busur. Dengan demikian, jumlah langkah yang diperlukan untuk mendapatkan pohon rentangan minimal dari sembarang graf terhubung terboboti adalah maksimum $(4 km \log_2 (n))$. Jadi waktu eksekusi dari algoritma ini mempunyai kompleksitas $\Theta(m \log n)$. Algoritma Kruskal Gready mempunyai kompleksitas $\Theta(n \log n)$. . Secara teoritis $\Theta(m \log n)$ sama dengan $\Theta(n \log n)$ untuk $n < m^2$. Dengan kata lain, pada kasus terburuk, algoritma Fase Pemilihan Busur sama cepatnya dengan algoritma Kruskal Gready.

DAFTAR PUSTAKA

Balas, Egon. dan Yu, Chang Su., (1986), *Finding A Maximum Clique In An Arbitrary Graph*, Siam Juornal Computing Vol. 15, No. 4, Hal. 1054-1056.

Buckley, F. dan Harary, F., (1990), *Distance In Graphs*, Addison-Wesley Publishing Company, London.

Chartrand, G. dan Oelllerman, O. R., (1993), *Applied and Algorithmic Graph Theory*, McGraw- Hill, Inc, New York.

Gavril, Fanica., (1972), *Algorithms for Minimum Coloring, Maximum Clique, Minimum Covering by Cliques, An Maximum Independent Set of A Chordal Graph*, Siam Jornal Computing Vol. 1, No. 2, Hal. 180-187.

Christofides, N., 1975, *Graph Theory, an Algorithmic Approach*, London, Academic Press.

Godran, M. dan Minoux, M., (1979), *Graph and Algorithms*, John Wiley and Sons Ltd, New York.

Harary, F., (1994), *Graph Theory*, Addison-Wesley Publishing Company, London.

Martono, T. *Teori Graf dan Aplikasinya*, UPT Produksi Media Informasi LSI-IPB, 1990.

Mayeda, Wataru., (1972), *Graph Theory*, John Wiley and Sons, Inc, New York.

Kerani D, (1990), *Analisis Jaringan*, Penerbit Katunika, Jakarta.

Stinson, D.R. *An Introduction to the Design and Analysis of Algorithms*, Winnipeg, Manitoba, Canada, 1985.

Thulasiraman, K. dan Swamy, M. N. S., (1992), *Graphs: Theory and Algorithms*, John Wiley and Sons, Inc, New York.