Penerapan Teori Graf Pada Algoritma *Routing*

Indra Siregar - 13508605

Program Studi Teknik Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10, Bandung 40132
e-mail: if18605@students.if.itb.ac.id

ABSTRAK

Jaringan komputer adalah suatu kumpulan komputer yang saling berkomunikasi satu sama lain dengan menggunakan cara-cara (protokol) tertentu. Komputer pada jaringan komputer dapat berupa *router*, *workstation*, *modem*, *printer*, dan perangkat-perangkat lainnya. Jaringan komputer dapat dimodelkan dengan menggunakan graf. Makalah ini khusus membahas pemodelan keterhubungan antar *router* dan algoritma *routing* yang digunakan, pada suatu jaringan komputer, dengan memanfaatkan teori graf. Pada bagian awal dijabarkan secara ringkas beberapa definisi terkait dengan teori graf. Bagian selanjutnya adalah pembahasan beberapa algoritma *routing* pada suatu jaringan komputer. Algoritma *routing* yang dibahas adalah algoritma Breadth-First, algoritma Dijkstra dan algoritma Bellman-Ford. Untuk mendapatkan kelebihan dan kekurangan dari setiap algoritma *routing* dilakukan dengan cara menganalisis kompleksitas pada setiap algoritma tersebut.

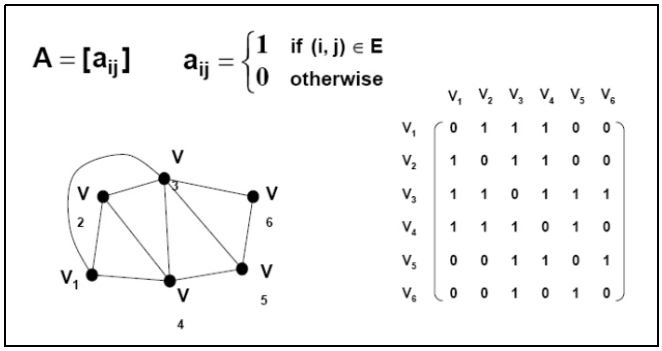
Kata kunci: graf, jaringan komputer, *router*, algoritma *routing*, *Bread-First*, Dijkstra, Bellman-Ford.

1. PENDAHULUAN

Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Graf sering digunakan untuk memodelkan jalur transportasi, penjadwalan, jaringan komputer, dan lain sebagainya.

Graph `G = (V, E)` adalah suatu graf dengan `V` adalah himpunan tidak kosong dari simpul-simpul (*vertices*), yaitu terdiri dari `n` buah simpul `{v1, v1, v3, ..., vn}` dan `E = {e1, e2, ..., en}` adalah himpunan sisi (*edges*) yang menghubungkan sepasang simpul. Jika pasangan simpul `(i, j)` saling terhubung, maka kedua simpul tersebut dikatakan bertetangga. Atau dengan kata lain jika `e` merupakan sisi yang menghubungkan simpul `i` dan `j`, `e = (i, j)`, maka sisi `e` disebut bersisian dengan simpul `i` dan `j`. Kardinalitas dari himpunan simpul dilambangkan dengan `|V|`.

Graf `G = (V, E)` dapat direpresentasikan dengan `|V| x |V|` matriks ketetanggaan. Gambar di bawah ini merupakan contoh representasi matriks ketetanggaan pada suatu graf tidak berarah.



Gambar 1 Matriks Ketetanggaan Pada Suatu Graf

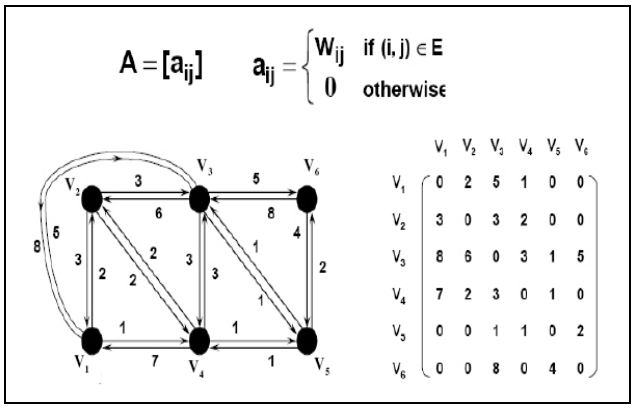
Beberapa definisi yang sering dijumpai dalam teori graf adalah sebagai berikut:

1. Jalur (*path*) antara simpul `i` dan `j` adalah sebuah urutan simpul dan sisi yang dimulai dari simpul `i` dan berakhir di simpul `j`, dengan setiap sisi bersisian dengan simpul sebelum dan sesudahnya. Sebuah jalur disebut sederhana jika setiap sisi yang dilalui pada jalur tersebut hanya sekali saja.
2. Jarak antara simpul `i` dan `j` adalah jumlah sisi minimum yang berada pada jalur dari simpul `i` ke simpul `j`.
3. Sebuah jalur sederhana disebut sebagai kalang (*cycle* atau *loop*) jika simpul awal juga merupakan simpul akhir.
4. Sebuah graf disebut terhubung jika terdapat jalur pada setiap pasang simpul `i` dan `j`.
5. Sebuah graf disebut berarah jika sisi pada graf tersebut memiliki arah. Dengan demikian simpul `(i, j)` pada sisi `e` tidak secara langsung

mengakibatkan bahwa simpul (j, i) juga ada pada e. Pada kondisi ini, sisi pada graf sering disebut dengan busur.

6. Graf disebut memiliki bobot (*weight*) jika pada setiap sisi (atau busur) diberi label dengan suatu bilangan.

Berbeda dengan graf tidak berarah pada gambar 1, representasi matriks untuk graf berarah ditunjukkan pada gambar di bawah ini. Pada gambar dapat dilihat bahwa elemen pada setiap matriks menggambarkan bobot (*weight*) pada setiap pasang simpul.



Gambar 2 Representasi Matriks Untuk Graf Berarah

1.1 Pohon (*Tree*)

Graf T merupakan sebuah pohon jika dan hanya jika ada satu jalur sederhana pada setiap pasang simpul (i, j). Jika N = |V|, maka ada N − 1 sisi (busur) dan semuanya terhubung, tanpa ada kalang (*loop*). Setiap simpul dapat menjadi akar pada pohon tersebut. Sebuah pohon dapat direpresentasikan dengan menyusun simpul-simpul dalam suatu urutan tertentu yang dimulai dari akar. Beberapa definisi pada pohon adalah sebagai berikut:

1. Setiap simpul (kecuali akar) hanya memiliki satu simpul orang tua.
2. Setiap simpul memiliki 0 atau lebih anak.
3. Sebuah sisi dengan 0 anak adalah merupakan daun.

1.2 Pohon Merentang (*Spanning Tree*)

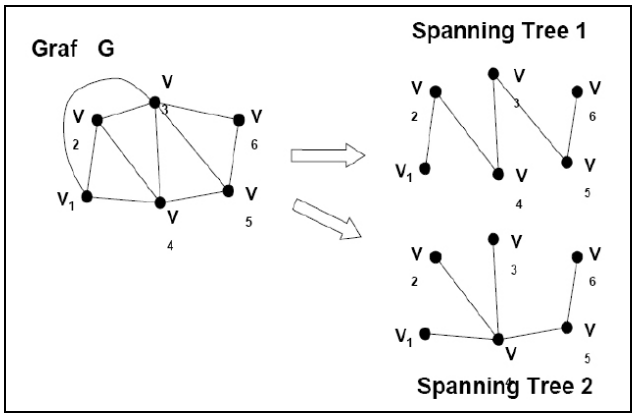
Subgraf (*subgraph*) dari sebuah graf G = (V, E) didapatkan dengan cara:

1. G' = (V', E'), dimana V' dan E' adalah himpunan bagian dari V dan E.
2. Untuk setiap sisi (busur) e pada E', simpul i dan j ada pada V'.

Sebuah subgraf T = (V', E') dari G = (V, E) adalah pohon merentang dari G jika:

1. T adalah sebuah pohon.
2. V' = V.

Gambar di bawah ini adalah dua buah pohon merentang yang didapatkan dari Graf G.



Gambar 3 Pohon Merentang Dari Graf G

1.3 Teori Graf dan Algoritma *Routing*

Pada jaringan komputer, aliran setiap paket dapat dimodelkan dengan menggunakan graf yang memiliki bobot. Simpul pada graf merepresentasikan *router*, sedangkan busur pada graf merepresentasikan *subnet*. *Routing* adalah proses untuk meneruskan paket dari suatu simpul ke simpul yang lainnya dengan berdasarkan asas jalur terpendek. Tujuannya adalah agar pengiriman paket tersebut dapat dilakukan secara efektif dan efisien. Algoritma *routing* untuk mendapatkan jalur terpendek diantaranya adalah algoritma *Bread-First*, Dijkstra, dan Bellman-Ford.

2. ALGORITMA ROUTING

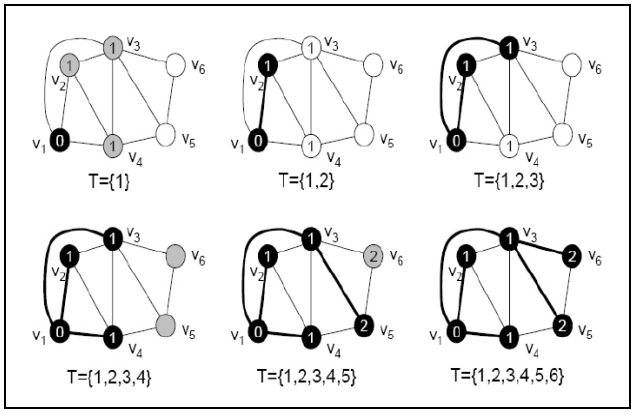
2.1 Bread-First

Algoritma ini mencari solusi dari pohon merentang yang berasal dari suatu graf. Ide dasar dari algoritma *Bread-First* adalah dengan melakukan eksplorasi terhadap simpul-simpul yang dimulai dari akar. Setiap simpul yang dijumpai akan diperiksa apakah merupakan solusi atau tidak.

Secara umum algoritma *Bread-First* akan melakukan langkah-langkah sebagai berikut:

1. Menentukan simpul yang berperan sebagai akar (misalnya simpul x).
2. Melakukan penjelajahan terhadap simpul yang bertetangga dengan simpul x (level 1).
3. Untuk setiap simpul yang berada pada level 1, dilakukan penjelajahan terhadap semua simpul yang bertetangga dengan semua simpul pada level 1 yang belum dijelajahi pada langkah sebelumnya (level 2).
4. Untuk setiap simpul yang dijelajahi, diperiksa apakah merupakan simpul tujuan atau tidak. Jika merupakan simpul tujuan maka penjelajahan akan berhenti dengan solusi merupakan simpul-simpul yang dijelajahi dari awal sampai kepada simpul akhir tujuan.
5. Jika bukan merupakan simpul tujuan makan perulangan akan berlaku selama belum semua simpul dijelajahi.

Gambar di bawah ini adalah contoh penerapan algoritma *Bread-First* pada sebuah pohon merentang yang berasal dari suatu graf.



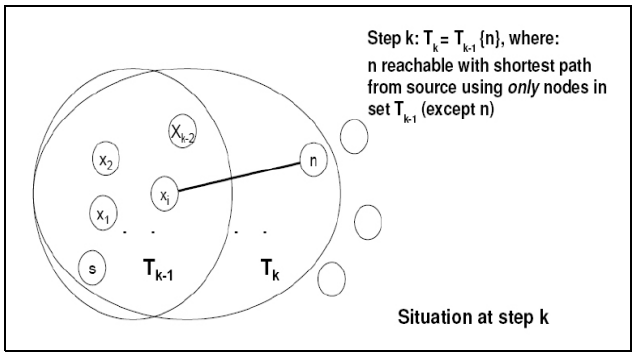
Gambar 4 Algoritma *Bread-First*

Kompleksitas algoritma Bread-First adalah $O(|V|^3)$.

2.2 Dijkstra

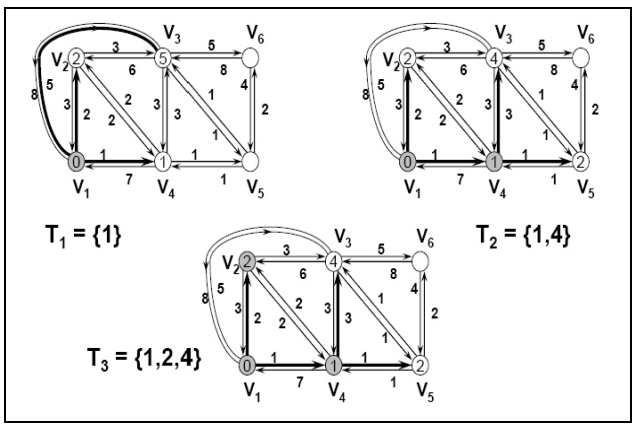
Ide dasar dari algoritma Dijkstra adalah memeriksa simpul dengan bobot terkecil dan memasukkannya ke dalam himpunan solusi. Pada setiap iterasi himpunan solusi ini akan berubah jika terdapat sisi dengan bobot lebih kecil dari sisi pada himpunan solusi. Secara umum langkah-langkah pada algoritma Dijkstra adalah sebagai berikut:

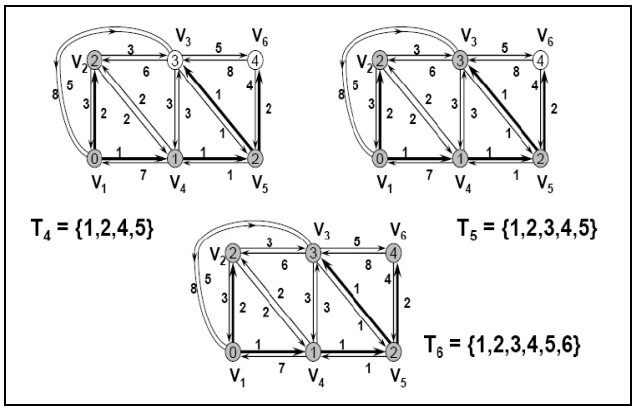
1. Pada langkah `k`: cari sisi `k` yang dapat dijangkau dari `s` dengan bobot minimum, dilambangkan dengan T_k.
2. Pada langkah `k + 1`: cari simpul `v` dengan jarak minimum dari `s` dan dapat dijangkau dengan simpul-simpul yang ada pada T_k (kecuali `v` sendiri).
3. T_{k+1} = Tk U {v}
4. Algoritma berhenti ketika semua simpul sudah dikunjungi.



Gambar 5 Algoritma Dijkstra

Untuk lebih jelasnya, gambar di bawah ini merupakan langkah-langkah yang dilakukan dengan menggunakan algoritma Dijkstra.





Gambar 6 Langkah-langkah Pada Algoritma Dijkstra

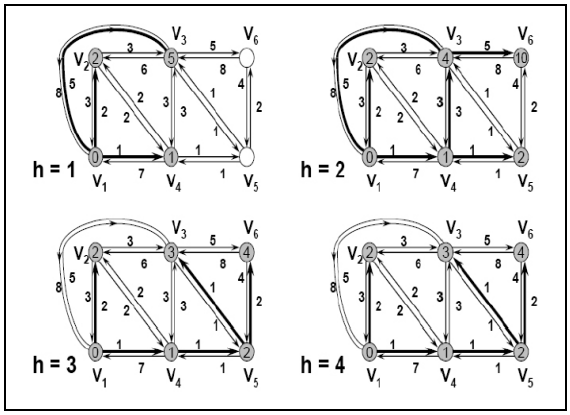
Kompleksitas algoritma Dijkstra adalah $O(|V|^2)$. Dengan implementasi yang efisien dapat diperoleh kompleksitas $O(|V| \cdot \log |V|)$.

2.3 Bellman-Ford

Misalnya simpul sumber adalah s, dan akan dicari jalur terpendek dari s ke semua simpul yang lain pada graf G. Langkah-langkah pada algoritma Bellman-Ford adalah sebagai berikut:

1. Temukan jalur terpendek antara s dan simpul lainnya, sehingga jalur ini adalah paling banyak memiliki 1 lompatan (*hop*).
2. Cari jalur terpendek antara s dan simpul lainnya dengan memiliki paling banyak 2 lompatan.
3. Lakukan iterasi sampai jalur terpendek memiliki jumlah lompatan paling banyak berjumlah diameter dari graf. Diameter graf adalah jarak maksimum antara pasangan simpul pada graf, diukur dengan lompatan (*hop*).

Gambar di bawah ini merupakan langkah-langkah yang dilakukan dengan menggunakan algoritma Bellman-Ford.



Gambar 7 Langkah-langkah Pada Algoritma Bellman-Ford

Kompleksitas yang diperoleh adalah $O(|V| . |E|)$, jadi untuk kasus terburuk adalah ketika $|E| = |V|^2$, adalah $O(|V|^3)$.

Algoritma Bellman-Ford juga dapat dilakukan secara rekursif, yaitu:

$$L(s,n) = \min_{j \in A_n} [L(s,j) + w(j,n)]$$

A_n: set of vertices that are predecessors of n in G

Dimana jalur terpendek antara s dan n diberikan dengan melakukan konkatenasi terhadap jalur terpendek antara s dengan satu *predecessor* simpul j pada n dan link antara j dan n.

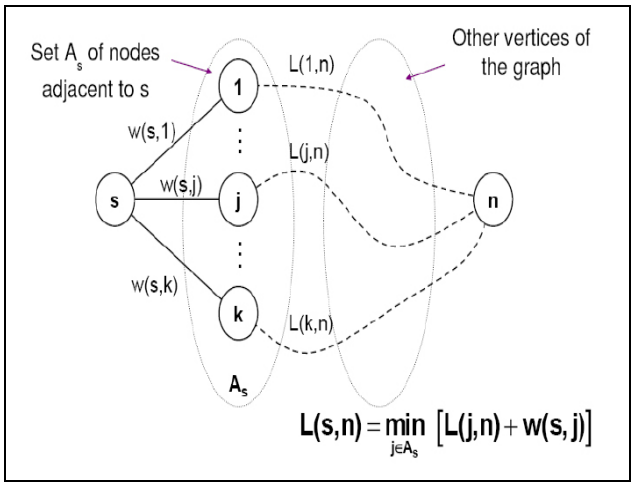
Juga berlaku:

$$L(s,n) = \min_{j \in A_s} [w(s,j) + L(j,n)]$$

A_s: set of vertices that are adjacent to s in G

Dimana jalur terpendek antara s dan n diberikan dari harga minimum dari konkatenasi jalur terpendek antara sebuah simpul j yang bertetangga dengan s dan n dan link antara s dan j.

Sifat rekursif pada algoritma Bellman-Ford dapat dilihat pada gambar di bawah ini:



Gambar 8 Algoritma Bellman-Ford Secara Rekursif

3. ALGORITMA DIJKSTRA VERSUS BELLMAN-FORD

Walaupun kedua algoritma (Dijkstra dan Bellman-Ford) sama-sama berusaha menemukan jalur terpendek, namun kedua algoritma tersebut memiliki beberapa perbedaan.

Salah satu perbedaan adalah dalam hal kompleksitas. Algoritma Bellman-Ford memiliki kompleksitas yang lebih besar dibandingkan dengan Dijkstra.

Perbedaan lainnya adalah pada algoritma Dijkstra simpul asal membutuhkan pengetahuan tentang topologi graf, yaitu informasi semua busur dan bobotnya. Oleh karena itu dibutuhkan pertukaran informasi dengan semua simpul. Sedangkan algoritma Bellman-Ford membutuhkan pengetahuan mengenai bobot dari sisi yang bertetangga dengan suatu simpul dan nilai dari jalur terpendek yang dimulai dari simpul tetangganya tersebut. Dalam hal ini hanya diperlukan komunikasi antara simpul yang bertetangga saja (implementasi terdistribusi).

Walaupun algoritma Bellman-Ford hanya membutuhkan distribusi informasi antara simpul yang bertetangga saja, namun implementasi terdistribusi pada algoritma Bellman-Ford dapat menyebabkan masalah yang disebut dengan *bad news phenomenon*, dimana iterasi yang diperlukan bisa sedemikian tinggi sehingga memperlambat distribusi informasi dari satu simpul ke simpul yang lainnya.

4. KESIMPULAN

Berdasarkan pemabahasan di atas, kesimpulan yang didapatkan dari makalah Penerapan Teori Graf Pada Algoritma *Routing* adalah sebagai berikut:

1. Keterhubungan antar *router* pada suatu jaringan komputer dapat dimodelkan dengan menggunakan graf.
2. Graf yang dihasilkan dari pemodelan tersebut dapat diproses lebih lanjut dengan menggunakan kaidah-kaidah yang berlaku pada graf.
3. Beberapa algoritma *routing* adalah algoritma *Bread-First*, Dijkstra, dan Bellman-Ford.
4. Setiap algoritma routing tersebut memiliki kompleksitas yang berbeda-beda. *Bread-First* memiliki kompleksitas $O(|V|^3)$, Dijkstra memiliki kompleksitas $O(|V|^2)$ dan Bellman-Ford adalah $O(|V|.|E|)$.

REFERENSI

[1] Becchetti, Lucas, *Computer Networks II: Graph theory and routing algorithms*, Universita degli Studi di Roma, 2008.

[2] Tanenbaum, Andrew S, *Computer Networks*, Prentice Hall PTR, 2002.

[3] Munir, Rinaldi, *Matematika Diskrit*, Edisi Ketiga, Penerbit Informatika: Bandung, 2005.