BAB II
TINJAUAN PUSTAKA

2.1 Graf

A. Definisi Graf

Graf adalah kumpulan simpul (*titik*) yang dihubungkan satu sama lain melalui sisi / busur (*edges*) (Zakaria, 2006). Suatu Graf terdiri dari dua himpunan.

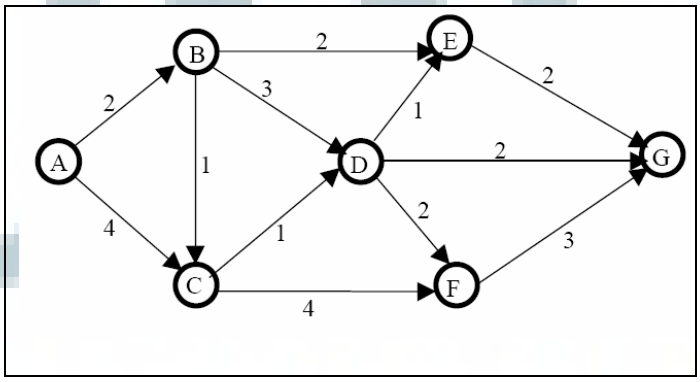
Simpul – simpul pada graf dapat merupakan obyek sembarang seperti kota, nama anak, jenis buah dan sebagainya. Busur dapat menunjukkan hubungan sembarang seperti rute penerbangan, sambungan telepon dan lain – lain.

B. Macam – Macam Graf

Menurut arah dan bobotnya, graf dibagi menjadi empat bagian, yaitu (Muttakhiroh, 2007) :

1. Graf Berarah dan Berbobot

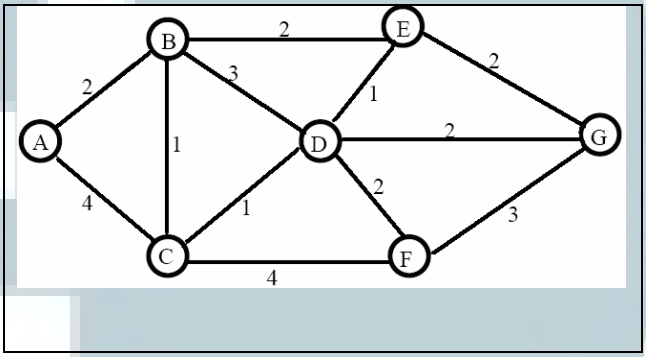
Tiap busur mempunyai anak panah dan bobot. Gambar 2.1 menunjukkan graf berarah dan berbobot yang terdiri dari tujuh titik yaitu titik A,B,C,D,E,F,G. Titik menujukan arah ke titik B dan titik C, titik B menunjukkan arah ke titik D dan titik C, dan seterusnya. Bobot antar titik A dan titik B pun telah diketahui.



Gambar 2.1 Graf berarah dan berbobot (Muttakhiroh, 2007)

2. Graf tidak berarah dan berbobot

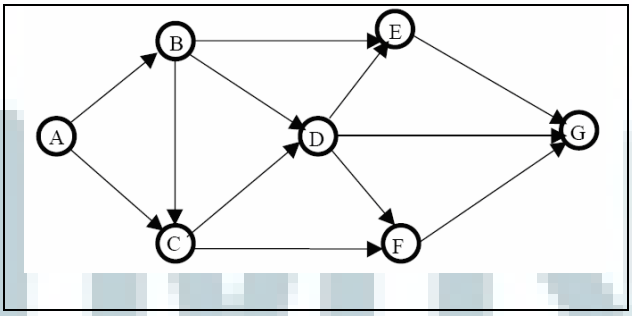
Tiap busur tidak mempunyai anak panah tetapi mempunyai bobot. Gambar 2.2 menunjukkan graf tidak berarah dan berbobot. Graf terdiri dari tujuh titik yaitu titik A,B,C,D,E,F,G. Titik A tidak menunjukkan arah ke titik B atau C, namun bobot antara titik A dan titik B telah diketahui. Begitu juga dengan titik yang lain.



Gambar 2.2 Graf tidak berarah dan berbobot (Muttakhiroh, 2007)

3. Graf berarah dan tidak berbobot

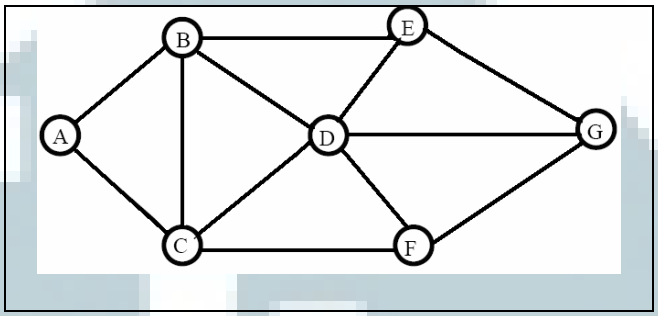
Tiap busur memiliki arah akan tetapi tidak memiliki bobot. Gambar 2.3 menunjukkan graf berarah dan tidak berbobot.



Gambar 2.3 Graf berarah dan tidak berbobot (Muttakhiroh, 2007)

4. Graf tidak berarah dan tidak berbobot

Tiap busur tidak memiliki anak panah dan tidak memiliki bobot. Gambar 2.4 menunjukkan graf tidak berarah dan tidak berbobot.



Gambar 2.4 Graf tidak berarah dan tidak berbobot (Muttakhiroh, 2007)

2.2 Permasalahan Optimasi

2.2.1 Permasalahan Optimasi

Secara umum, penyelesaian masalah pencarian jalur terpendek dapat dilakukan dengan menggunakan dua metode, yaitu metode konvensional dan metode heuristik (Muttakhiroh, 2007). Metode konvensional diterapkan dengan perhitungan matematis biasa, sedangkan metode heuristik diterapkan dengan perhitungan kecerdasan buatan.

a. Metode Konvensional

Metode konvensional adalah metode yang menggunakan perhitungan matematis biasa. Ada beberapa metode konvensional yang biasa digunakan untuk melakukan pencarian jalur terpendek, diantaranya: algoritma Dijkstra, algoritma Dijkstra, dan algoritma Bellman-Ford (Muttakhiroh, 2007).

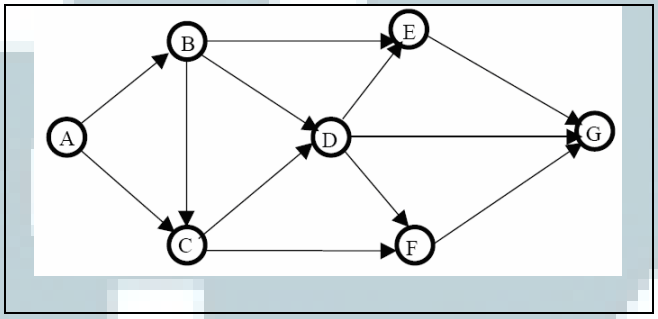
b. Metode Heuristik

Metode Heuristik adalah sub bidang dari kecerdasan buatan yang digunakan untuk melakukan pencarian dan optimasi. Ada beberapa algoritma pada metode

heuristik yang biasa digunakan dalam permasalahan optimasi, diantaranya algoritma genetika, algoritma semut, logika fuzzy, jaringan syaraf tiruan, pencarian tabu, *simulated annealing*, dan lain-lain (Muttakhiroh, 2007).

2.2.2 Permasalahan Jalur Terpendek

Jalur terpendek adalah suatu jaringan pengarahan perjalanan dimana seorang pengarah jalan ingin menentukan jalur terpendek antara dua kota, berdasarkan beberapa jalur alternative yang tersedia, dimana titik tujuan hanya satu (Muttakhiroh, 2007).



Gambar 2.5 Graf ABCDEFG (Muttakhiroh, 2007)

Pada gambar di atas, misalkan dari kota A ingin menuju ke kota G. Untuk menuju kota G, dapat dipilih beberapa jalur yang tersedia :

A → B → C → D → E → G

A → B → C → D → F → G

A → B → C → D → G

A → B → C → F → G

A → B → D → E → G

A → B → D → F → G

A → B → D → G

A → B → E → G

A → C → D → E → G

A → C → D → F → G

A → C → D → G

A → C → F → G

Berdasarkan data di atas, dapat dihitung jalur terpendek dengan mencari jarak antara jalur-jalur tersebut. Apabila jarak antar jalur belum diketahui, jarak dapat dihitung berdasarkan koordinat kota-kota tersebut, kemudian menghitung jalur terpendek yang dapat dilalui.

2.3 Pengenalan Algoritma Pencarian Rute

Pencarian jalur terpendek merupakan suatu permasalahan yang sering timbul pada pengguna transportasi, karena dalam melakukan perjalanan, pengguna transportasi membutuhkan solusi rute atau jarak yang paling minimum (terkecil) sehingga efisiensi waktu dapat terpenuhi (Sitanggang, 2011).

Dalam melakukan pemilihan terhadap rute terpendek, dapat dilakukan dengan metode algoritma. Algoritma merupakan kumpulan instruksi atau perintah yang dibuat secara jelas dan sistematis berdasarkan urutan yang logis untuk menyelesaikan suatu masalah. Sedangkan algoritma pencari rute adalah algoritma yang menentukan bagaimana memilih rute optimal antara tempat asal dan tujuan dengan memperhitungkan jarak terpendek.

Ada beberapa algoritma pencarian rute yang sebelumnya sudah dikembangkan, antara lain Algoritma Djikstra, Algoritma Dijkstra dan Algoritma Bellman-Ford. Algoritma yang akan dipakai dalam skripsi ini adalah algoritma Dijkstra. Algoritma Djikstra merupakan algoritma yang paling sering digunakan

dalam menentukan rute terpendek, sederhana (sifrat *greedy* atau rakus dalam pemilihan graf) dalam penggunaannya dengan hanya menggunakan vertex – vertex sederhana pada jaringan jalan yang tidak rumit (Chameri, 2006). Pada beberapa kasus algoritma Djikstra bersifat *greedy* (tidak memikirkan konsekuensi yang akan terjadi pada saat memilih keputusan) tidak memberikan solusi yang terbaik, maka dalam hal ini digunakan algoritma Dijkstra. Prinsip dari algoritma ini adalah *"jika solusi total optimal, maka bagian solusi sampai suatu tahap (misalnya tahap ke-i) juga optimal"*, yang mempunyai pengertian bahwa selain diperolehnya suatu rute terpendek dari simpul awal ke simpul akhir, juga akan diperoleh nilai – nilai rute antar simpul.

2.4 Algoritma Dijkstra

Algoritma Dijkstra, dinamai menurut penemunya, Edsger Dijkstra adalah sebuah algoritma rakus (*greedy algorithm*) dalam memecahkan permasalahan jarak terpendek (*shortest path problem*) untuk sebuah graf berarah (*directed graph*) dengan bobot-bobot sisi (*edge weights*) yang bernilai positif (Sitanggang, 2011). Misalnya, bila vertices dari sebuah graf melambangkan kota-kota dan bobot sisi (*edge weights*) melambangkan jarak antara kota-kota tersebut, maka algoritma Dijkstra dapat digunakan untuk menemukan jarak terpendek antara dua kota. Algoritma Dijkstra merupakan salah satu varian bentuk algoritma populer dalam pemecahan persoalan yang terkait dengan masalah optimasi. Sifatnya sederhana dan lempang (*straightforward*). Sesuai dengan arti *greedy* 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 didapatkan saat ini (*take what you can get now*!), dan keputusan yang telah diambil pada setiap langkah tidak akan bisa diubah kembali.

Input algoritma ini adalah sebuah graf berarah yang berbobot (*weighted directed graph*) G dan sebuah sumber vertex s dalam G dan V adalah himpunan semua vertices dalam graph G (Pu Jian, 2004). Setiap sisi dari graf ini adalah pasangan vertices (u,v) yang melambangkan hubungan dari vertex u ke vertex v. Himpunan semua tepi disebut E.

Algoritma Dijkstra merupakan algoritma pencarian rute tradisional dengan mencari node dengan fungsi F terkecil. Proses ini diulang-ulang terus hingga tujuan dicapai.

Secara singkat algoritma Dijkstra dapat dijelaskan sebagai berikut (Gross And Yallen, 1998):

Masukkan : Graf berbobot.

Proses :

- Inisialisasi verteks.
- Inisialisasi jarak antar verteks.
- Tentukan verteks awal (s) dan verteks tujuan (t).
- Beri label permanen : 0 ke verteks awal (s) dan label sementara : .o ke verteks lainnya.
- Untuk setiap verteks V, yang belum mendapat label permanen, mendapatkan label sementara: min {label lama V1,(label lama V,+ D,,)}
- Cari harga minim diantara semua verteks yang masih berlabel sementara.

- Jadikan verteks minimum yang berlabel sementara menjadi verteks dengan label pefinanen, jika lebih dari satu verteks dipilih sembarang.
- Ulangi langkah 5 sampai 7 hingga verteks tujuan mendapat label permanen.
- Simpan hasil perhitungan.
- Tampilkan hasil pencarian.

Berikut merupakan source code untuk algoritma dijkstra yang diambil dari http://rosettacode.org/wiki/Dijkstra%27s_algorithm

```php
<?php
function dijkstra($graph_array, $source, $target) {
    $vertices = array();
    $neighbours = array();
    foreach ($graph_array as $edge) {
        array_push($vertices, $edge[0], $edge[1]);
        $neighbours[$edge[0]][] = array("end" => $edge[1], "cost" => $edge[2]);
    }
    $vertices = array_unique($vertices);

    foreach ($vertices as $vertex) {
        $dist[$vertex] = INF;
        $previous[$vertex] = NULL;
    }

    $dist[$source] = 0;
    $Q = $vertices;
    while (count($Q) > 0) {

        // TODO - Find faster way to get minimum
        $min = INF;
        foreach ($Q as $vertex){
            if ($dist[$vertex] < $min) {
                $min = $dist[$vertex];
                $u = $vertex;
            }
        }

        $Q = array_diff($Q, array($u));
        if ($dist[$u] == INF or $u == $target) {
            break;
        }
         if (isset($neighbours[$u])) {
             foreach ($neighbours[$u] as $arr) {
                 $alt = $dist[$u] + $arr["cost"];
                 if ($alt < $dist[$arr["end"]]) {
                     $dist[$arr["end"]] = $alt;
                     $previous[$arr["end"]] = $u;
                 }
             }
         }
     }
     $path = array();
     $u = $target;
     while (isset($previous[$u])) {
         array_unshift($path, $u);
         $u = $previous[$u];
     }
     array_unshift($path, $u);
     return $path;
}

$graph_array = array(
                    array("a", "b", 7),
                    array("a", "c", 9),
                    array("a", "f", 14),
                    array("b", "c", 10),
                    array("b", "d", 15),
                    array("c", "d", 11),
                    array("c", "f", 2),
                    array("d", "e", 6),
                    array("e", "f", 9)
                );

$path = dijkstra($graph_array, "a", "e");

echo "path is: ".implode(", ", $path)."\n";
```

Gambar 2. 6 Source Code Algoritma Dijkstra

Algoritma ini akan membentuk *array* $vertices yang berisi titik – titik (*node*), serta *array* $neighbours yang berisi tetangga dari *node* tersebut dan juga jarak dari setiap *node* tersebut. Kemudian setiap *node* akan diberi nilai jarak awal ($dist) berupa *infinite* dan isi dari *node* sebelumnya ($previous) berupa NULL. Setelah semua jarak *node* telah diberi nilai *infinite* dan semua node sebelumnya telah diberi nilai NULL, selanjutnya node awal akan diberi jarak 0 (nol). Selama jumlah *node* masih lebih besar dari nol, maka akan terus dilakukan pencarian jarak dari setiap *node* dengan *node* tetangganya dan juga menandai setiap *node* sebelum dari setiap *node* yang telah dikunjungi.

Setelah semua node telah berhasil dikunjungi dan telah memiliki jarak serta node sebelummya, maka algoritma ini akan menampilkan hasil yang telah didapat dengan mengambil jarak serta *node* sebelumnya yang ada pada setiap *node*.

2.5 Website

World Wide Web secara luas dikenal dengan istilah *Web*. *Web* pertama kali diperkenalkan pada tahun 1992 (Mulyanto, 2008). Hal ini sebagai hasil usaha pengembangan yang dilakukan CERN di Swiss. Internet dan *web* adalah dua hal yang berbeda. Internet menggunakan TCP/IP sebagai protocol operasionalnya, sedangkan *web* menggunakan HTTP (*Hyper Text Transfer Protocol*).

2.5.1 Standar Teknologi Web

Secara umum teknologi desain *web* terbagi menjadi beberapa *layer* (lapisan), yaitu *structural layer*, *presentation layer* dan *behavioral layer*.

A. Structural Layer

Layer ini berhubungan dengan struktur dokumen dokumen *web.* Bagaimana sebuah dokumen tersusun, format apa yang dipakai, tanda atau mark up apa yang digunakan merupakan bagian dari layer ini. Standar teknologi yang direkomendasikan saat ini adalah Extensible Hypertext Markup Language (XHTML) dan Extensible Markup Language (XML). XHTML adalah HTML versi terakhir (4.01) yang ditulis ulang dengan dengan aturan-aturan yang lebih ketat mengacu pada XML. Sedangkan XML adalah sekumpulan aturan untuk menyusun bahasa markup.

B. Presentation Layer

Layer ini berhubungan dengan bagaimana mengatur tampilan dokumen pada layar, suara yang keluar, atau bagaimana format pencetakan dokumen. Pada teknologi web lama bagian ini menyatu dengan structural layer. Tapi pada standar baru, layer ini disarankan untuk dipisah. Yang termasuk teknologi ini adalah *Cascading Style Sheets* (CSS).

C. Behavioral Layer

Layer ini berhubungan dengan masalah penggunaan bahasa skrip dan pemrogramannya untuk tujuan meningkatkan sisi interaktif dan dinamis halaman web. Yang termasuk dalam layer ini adalah Document Object Model (DOM) dan JavaScript. DOM memungkinkan suatu dokumen atau skrip untuk mengakses atau meng-update isi, struktur, dan style dari dokumen. JavaScript merupakan teknologi yang cukup lama dan tetap digunakan untuk menambah dokumen menjadi lebih interaktif.

2.5.2 Web Statis dan Web Dinamis

Halaman *web* dapat digolongkan menjadi *web* statis dan *web* dinamis. Pengertian *web* statis dan *web* dinamis seringkali mengundang perdebatan. Sebagian pengguna internet menyatakan jika pada halaman-halaman *web* dilengkapi dengan animasi yang bergerak maka disebut *web* dinamis sedangkan jika halaman-halaman *web* tersebut hanya berisi teks dan gambar yang tidak bergerak maka disebut *web* statis. Namun berdasarkan kesepakatan maka pengertian statis dan dinamis tidak ditentukan oleh ada atau tidaknya animasi bergerak pada halaman-halaman *web*, tetapi ditentukan oleh isi atau informasi yang ada pada halaman-halaman tersebut.

Data dan informasi yang ada pada *web* statis tidak berubah-ubah. Dokumen web yang dikirim kepada *client* akan sama isinya dengan apa yang ada di *web* server. Sedangkan *web* dinamis, memiliki data dan informasi yang berbeda-beda tergantung *input* apa yang disampaikan *client*. Dokumen yang sampai di *client* akan berbeda dengan dokumen yang ada di web server.

2.5.3 Web Mobile

Web Mobile adalah sebuah teknologi baru telah mengakomodasi kebutuhan akan akses internet melalui perangkat *mobile* (bergerak). Jika sebelumnya *web* atau internet hanya dapat diakses melalui komputer (PC /*Personal Computer*), maka dengan adanya teknologi *web mobile*, sebuah *web* akan dapat diakses melalui perangkat bergerak seperti telepon seluler (*mobile phone*) dan atau PDA/*Pocket PC* . *Wireless web* atau internet *web mobile*

memungkinkan pengguna untuk mencari informasi melalui peralatan *wireless* atau *mobile device* miliknya.