BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka.

Acuan tinjauan pustaka penelitian terletak pada objek, pemodelan, studi kasus, dan bahasa pemrograman. Dalam penelitian yang dilakukan oleh Imron Fauzi tahun 2011 dengan objek pencarian rute tercepat dan rute terpendek studi kasus pada jalan raya antara wilayah Blok M dan Kota.Pada penelitian yang dilakuan oleh Siti nandiroh,Haryanto tahun 2009 dengan objek penentuan rute terpendek jalan dan lokasi pariwisata di Kota Surakarta menggunakan Algoritma *Dijkstra*. Penelitian yang dilakukan oleh Deiby T.Salaki tahun 2011 dengan objek penentuan lintasan terpendek dari Fmipa ke Rektorat dan Fakultas lain di Unsrat Manado.Penelitian yang sama dilakukan oleh Kalsum Mustika tahun 2012 dengan objek aplikasi sistem informasi geografis penentuan lintasan terpendek pengantaran barang menggunakan algoritma *A** (Studi Kasus pada Cv.BKL Express untuk Wilayah Kota Medan). Penelitian yang di dilakukan Syarah Sukmadria S tahun 2014 dengan objek perancangan aplikasi pencarian rute terpendek dengan metode *Floyd* pada taksi. Sedangkan penelitian yang dilakukan oleh Supriyanto tahun 2016 dengan objek penerapan algoritma *dijkstra* untuk menentukan jalur terpendek lokasi Bengkel AHASS di Kabupaten Bantul.

2.2 Landasan Teori

2.2.1 SIG/GIS

Sistem Informasi Georafis (*Georaphic Information System*) adalah merupakan suatu sistem informasi yang berbasis komputer, yang dirancang untuk bekerja dengan menggunakan data yang memiliki informasi spasial (bereferensi keruangan). Sistem ini mengcapture, mengecek, mengintegrasikan, memanipulasi, menganalisa, dan menampilkan data yang secara spasial mereferensikan kepada kondisi bumi.

2.2.2 Sistem

2.2.2.1 Pengertian Sistem

Sistem adalah sekelompok elemen-elemen yang terintegrasi dengan maksud yang sama untuk mencapai tujuan (McLeod: 2010). Menurut Satzinger, Jackson, dan Burd (2010:6) sistem merupakan sekumpulan komponen yang saling berhubungan dan bekerja bersama untuk mencapai suatu tujuan. Menurut Murdik (2002) bahwa sistem adalah seperangkat elemen yang membentuk kegiatan atau suatu prosedur atau bagian pengolahan yang mencari suatu tujuan-tujuan bersama dengan mengoperasikan data atau barang pada waktu tertentu untuk menghasilkan informasi atau energi atau barang. Bedasarkan ketiga pengertian di atas, maka dapat disimpulkan bahwa pengertian sebuah sistem adalah sekumpulan elemen yang terintegasi dan bekerja bersama guna mencapai suatu tujuan tertentu dengan mengoperasikan data atau barang pada waktu tertentu.

2.2.2.2 Karakteristik Sistem

Suatu sistem mempunyai karakteristik atau sifat-sifat yang tertentu, yaitu mempunyai komponen, batas sistem, lingkungan luar sistem, penghubung, masukan, keluaran, tujuan (Jogiyanto, 1999:3). Adapun pengertian dari masing-masing karakteristik Sistem tersebut adalah sebagai berikut:

1. Komponen Sistem Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerja sama membentuk satu kesatuan.
2. Batasan Sistem Batasan sistem (*boundary*) merupakan daerah yang membatasi antara suatu dengan Sistem yang lainnya atau dengan lingkungan luarnya.
3. Lingkunagan Luar Sistem Lingkungan luar sistem (*envronment*) dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi oprerasi sistem.
4. Penghubung Sistem Penghubung (*interface*) merupakan media penghubung antara satu sub sistem dengan sub sistem yang lainya.

5. Masukan Sistem Masukan (*input*) energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintenance input*) dan masukan sinyal (*signal input*). Maintenance input adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. Signal input adalah energi yang diproses untuk didapatkan keluaran.
6. Keluaran Sistem Keluaran (*output*) adalah hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan.
7. Pengolahan Sistem Suatu sistem dapat mempunyai suatu bagian pengolahan yang akan merubah masukan menjadi keluaran.
8. Sasaran atau tujuan sistem sasaran dari sistem sangat menentukan sekali masukan yang dibutuhkan sistem dan keluaran yang dihasilkan sistem. Suatu sistem pasti mempunyai tujuan atau sasaran, kalau tidak mempunyai sasaran maka operasi sistem tidak ada gunanya. Sasaran dari sistem sangat menentukan sekali masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuannya

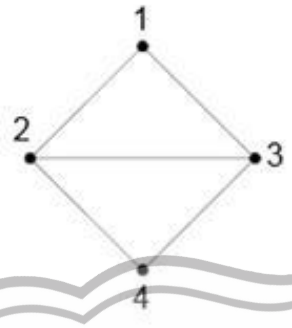
2.2.3 *Graf*

2.2.3.1 Definisi *Graf*

Teori *graf* merupakan pokok bahasan yang memiliki banyak terapan sampai saat ini. *Graf* digunakan untuk mempresentasikan objek-objek diskrit dan hubungan dengan objek-objek tersebut. Secara matematis *graf* didefinisikan sebagai pasangan himpunan (V,E), ditulis dengan notasi G = (V,E), yang dalam hal ini V adalah himpunan tidak kosong dari sipul-simpul (*vertex* atau *node*) dan E adalah himpunan sisi (*edge*) yang menghubungkan sepasang simpul (Munir, 2005).

Simpul (*vertex*) pada *graf* dapat dinyatakan dengan huruf, bilangan atau gabungan keduanya. Sedangkan sisi-sisi yang menghubungkan simpul u dengan simpul v dinyatakan dengan pasangan (u, v) atau dinyatakan dengan lambang e1,

e2, e3 dan seterusnya. Dengan kata lain, jika e adalah sisi yang menghubungkan simpul u dengan simpul v, maka e dapat dituliskan sebagai e = (u, v).

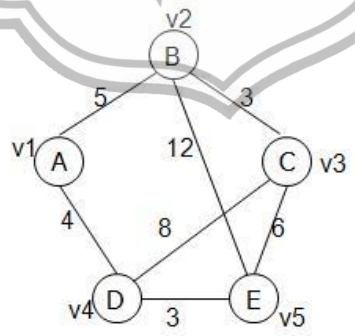


Gambar 2.1 *Graf* Sederhana

2.2.3.2 Jenis - jenis *Graf*

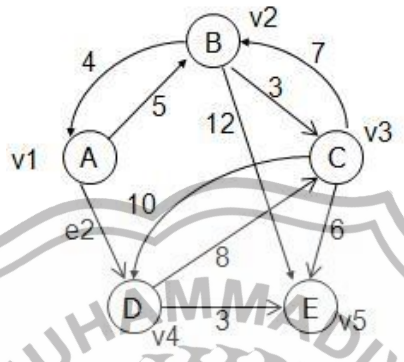
Klasifikasi pada *graf* cukup luas, klasifikasi tersebut bergantung pada faktor-faktor yang membedakannya. Berdasarkan orientasi arah pada sisinya, maka secara umum *graf* dibedakan atas dua jenis sebagai berikut :

1. *Graf* tidak berarah (*undirected graph*) *Graf* yang sisinya tidak memiliki orientasi arah disebut *graf* tidak berarah. Pada *graf* tidak berarah, urutan pasangan simpul yang dihubungkan oleh sisi tidak diperhatikan. Jadi, (u, v) = (v, u) adalah sisi yang sama



Gambar 2.2 *Graf* Tidak Berarah

2. *Graf* berarah (*directed graph*) *Graf* yang setiap sisinya diberikan orientasi arah disebut graf berarah, pada *graf* berarah (u, v) dan (v, u) menyatakan dua buah sisi yang berbeda. Dengan kata lain dapat ditulis (u, v) ≠ (v, u).



Gambar 2.3 *Graf* Berarah

2.2.4 Terminologi Dasar Graf

Dalam teori *Graf* terdapat beberapa terminologi (istilah) yang berkaitan dengan *graf* yang akan di definisikan satu persatu sebagai berikut:

a. Bertetangga (*Adjacent*)

Dua buah simpul pada *graf* dikatakan bertetangga bila keduanya terhubung langsung dengan sebuah sisi. Dengan kata lain, vj bertetangga dengan vk jika (vj, vk) adalah sebuah sisi pada *graf*.

b. Berisian (*Incident*)

Untuk sembarang sisi e = (vj, vk), sisi e dikatakan bersisian dengan simpul vj dan vk.

c. Simpul Terpencil (*Isolated Vertex*)

Simpul terpencil adalah simpul yang tidak mempunyai sisi yang bersisian dengannya. Atau dapat dinyatakan bahwa simpul terpencil adalah simpul yang tidak satupun bertetangga dengan simpul-simpul lainnya.

d. *Graf* Kosong (*Null Graph* atau *Empty Graph*)

Graf yang himpunan sisinya merupakan himpunan kosong disebut sebagai *graf* kosong dan ditulis sebagai Nn, yang dalam hal ini n adalah jumlah simpul

e. Derajat (*Degree*)

Derajat suatu simpul pada *graf* tak berarah adalah jumlah sisi yang bersisian dengan simpul tersebut. Pada *graf* berarah, derajat simpul v dinyatakan din (v) dan dout (v), yang dalam hal ini din (v) = derajat masuk = derajat busur yang masuk ke simpul. Dout (v) = derajat keluar = derajat busur yang keluar dari simpul. Untuk sembarang *graf* G, banyaknya simpul yang berderajat ganjil selalu genap.

f. Lintasan (*Path*)

Lintasan yang panjangnya n dari simpul awal v0 ke simpul tujuan vn didalam *graf* G adalah barisan berselang seling simpul-simpul dan sisi-sisi yang berbentuk v0, e1, v1, e2, v2, e3, v3..., vn-1, en, vn sedemikian sehingga e1 = (v0, v1), e2 = (v1-1, vn) adalah sisi-sisi dari *graf* G.

g. Siklus (*Cycle*) Atau Sirkuit (*Circuit*)

Lintasan yang berawal dan berakhir di simpul yang sama disebut sirkuit atau siklus.

h. Terhubung (*Connected*)

Graf tak berarah disebut *graf* terhubung (*connected graph*) jika untuk setiap pasang simpul vi dan vj di dalam himpunan V terhadap lintasan dari vi ke vj (yang berarti ada lintasan dari vi ke vj). Jika tidak maka disebut *graf* tak terhubung. *Graf* berarah dikatakan terhubung jika *graf* tak berarahnya terhubung (*Graf* tak berarahnya dari *graf* diperoleh dengan menghilangkan arahnya). *Graf* berarah disebut *graf* terhubung terkuat (*strongly connected graph*) apabila untuk setiap pasang simpul sembarang vi dan vj di G terhubung kuat. Kalau tidak, G disebut *graf* terhubung lemah.

2.2.5 Algoritma *Dijkstra*

2.2.5.1 Pengertian Algoritma *Dijkstra*

Algoritma *Dijkstra* ditemukan oleh Edsger W. *Dijkstra* pada tahun 1959 yang merupakan salah satu varian bentuk algoritma populer dalam pemecahan persoalan yang terkait dengan masalah optimasi dan bersifat sederhana. Algoritma ini menyelesaikan masalah mencari sebuah lintasan terpendek (sebuah lintasan yang mempunyai panjang minimum) dari *vertex a* ke *vertex z* dalam *graph* berbobot. Adapun bobot tersebut merupakan bilangan positif jadi tidak dapat dilalui oleh *node* negatif, namun jika terjadi demikian, maka penyelesaian yang diberikan adalah infiniti.

Umumnya sebelum melakukan iterasi, algoritma telah mengidentifikasikan jarak terdekat dari i-1 *vertex* terdekatnya. Selama seluruh *edge* berbobot tertentu yang (positif), maka *vertex* terdekat berikutnya dari *node* asal dapat ditemukan selama *vertex* berdekatan dengan *vertex* Ti. Kumpulan *vertex* yang berdekatan dengan *vertex* di Ti dapat dikatakan sebagai *"fringevertices"*. *Vertex* inilah yang merupakan kandidat dari Algoritma *Dijkstra* untuk memilih *vertex* berikutnya datri *node* asal. Algoritma *Dijkstra* mempunyai sifat sederhana (*straight forward*). Sesuai dengan arti *greedy* yang secara harfiah berarti tamak atau rakus, namun tidak dalam konteks negatif, Algortima *Greedy* ini hanya memikirkan solusi terbaik yang akan diambil pada setiap langkah tanpa memikirkan konsekuensi kedepan.

2.2.6 Elemen - elemen Penyusun *Greedy* dalam Algoritma *Dijkstra*

2.2.6.1 Himpunan Kandidat

Himpunan ini berisi elemen-elemen yang memiliki peluang untuk membentuk solusi. Pada persoalan lintasan terpendek dalam *graf*, himpunan kandidat ini adalah himpunan simpul pada *graf* tersebut.

a. Himpunan Solusi

   Himpunan ini berisi solusi dari permasalahan yang diselesaikan dan elemennya terdiri dari elemen dalam himpunan kandidat namun tidak semuanya atau dengan kata lain himpunan solusi ini adalah bagian dari himpunan kandidat.

b. Fungsi Seleksi

   Fungsi Seleksi adalah fungsi yang akan memilih setiap kandidat yang memungkinkan untuk menghasilkan solusi optimal pada setiap langkahnya.

c. Fungsi Kelayakan

   Fungsi kelayakan akan memeriksa apakah suatu kandidat yang telah terpilih (terseleksi) melanggar *constraint* atau tidak. Apabila kandidat melanggar *constraint* maka kandidat tidak akan dimasukkan ke dalam himpunan solusi.

d. Fungsi Objektif

   Fungsi Objektif akan memaksimalkan atau meminimalkan nilai solusi. Tujuannya adalah memilih satu saja solusi terbaik dari masing-masing anggota himpunan solusi.

Jika menggunakan Algoritma *Dijkstra* untuk menentukan jalur terpendek dari suatu *graph* maka pasti akan menemukan jalur yang terbaik. Karena pada waktu penentuan jalur yang akan terpilih, akan dianalisis bobot dari *node* yang belum terpilih. Lalu dipilih *node* dengan bobot yang terkecil, jika ternyata ada bobot yang lebih kecil jika melalui *node* tertentu maka bobot dapat berubah. Algoritma ini akan berhenti jika semua *node* sudah terpilih dan dengan Algoritma *Dijkstra* ini dapat menemukan jarak terpendek dari *node* yang dipilih.

Persoalan mencari lintasan terpendek di dalam *graf* merupakan salah satu persoalan optimasi. *Graf* yang digunakan dalam mencari lintasan terpendek

dalam *graph* berbobot. Bobot pada sisi *graph* dapat menyatakan jarak antar kota, waktu, biaya dan sebagainya. Dalam hal ini bobot harus bernilai positif.

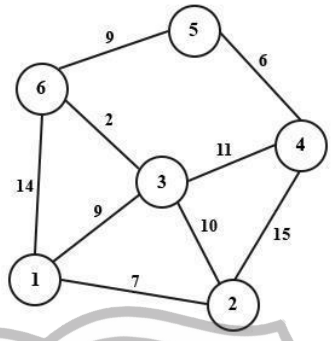
Dijkstra akan melakukan pengembangan pencarian dari satu titik ke titik lain dan ke titik selanjutnya tahap demi tahap. Adapun urutan logika dari Algoritma *Dijkstra* adalah sebagai berikut:

1. Beri nilai bobot (jarak) untuk setiap titik ke titik lainnya, lalu set nilai 0 pada *node* awal dan nilai tak hingga terhadap *node* lain (belum terisi).
2. Set semua *node* "belum terpilih" dan set *node* awal sebagai "*node* keberangkatan".
3. Dari *node* keberangkatan, pertimbangkan *node* tetangga yang belum terpilih dan hitung jaraknya dari titik keberangkatan. Sebagai contoh, jika titik keberangkatan A ke B memiliki bobot jarak 6 dan dari B ke node C berjarak 2, maka jarak ke C melewati B menjadi 6+2 =8. Jika jarak ini lebih kecil dari jarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru.
4. Saat kita selesai mempertimbangkan setiap jarak terhadap *node* tetangga, tandai *node* yang telah terpilih sebagai "*Node* terpilih". *Node* terpilih tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.
5. Set "*Node* belum terpilih" dengan jarak terkecil (dari *node* keberangkatan) sebagai "*Node* keberangkatan" selanjutnya dan lanjutkan dengan kembali ke step 3.

2.2.7 Langkah - langkah dalam Algoritma *Dijkstra*

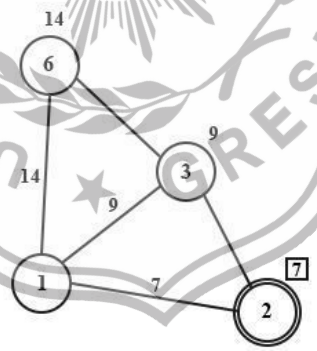
Dibawah ini penjelasan langkah per langkah pencarian jalur terpendek secara rinci dimulai dari *node* awal sampai *node* tujuan dengan nilai jarak terkecil.

1. Pada Gambar 2.4, *node* awal 1, nodetujuan 5. Setiap *edge* yang terhubung antar node telah diberi nilai



Gambar 2.4 Contoh Kasus *Dijkstra* – Langkah 1

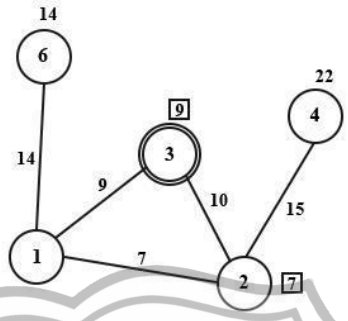
2. *Dijkstra* melakukan kalkulasi terhadap *node* tetangga yang terhubung langsung dengan *node* keberangkatan (*node* 1), terlihat pada Gambar 2.5, hasil yang didapat adalah *node* 2 karena bobot nilai *node* 2 paling kecil dibandingkan nilai pada *node* lain, nilai = 7 (0+7)



Gambar 2.5 Contoh Kasus *Dijkstra* – Langkah 2

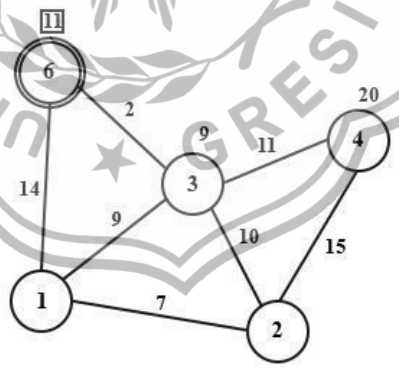
3. *Node* 2 diset menjadi *node* keberangkatan dan ditandai sebagai *node* yang terpilih. *Dijkstra* melakukan kalkulasi kembali terhadap *node-node* tetangga yang terhubung langsung dengan *node* yang terpilih. Kemudian pada Gambar 2.6, kalkulasi *Dijkstra* menunjukkan bahwa *node* 3 yang

menjadi *node* keberangkatan selanjutnya karena bobot nya yang paling kecil dari hasil kalkulasi terakhir, nilai 9 (0+9).



Gambar 2.6 Contoh Kasus *Dijkstra* – Langkah 3

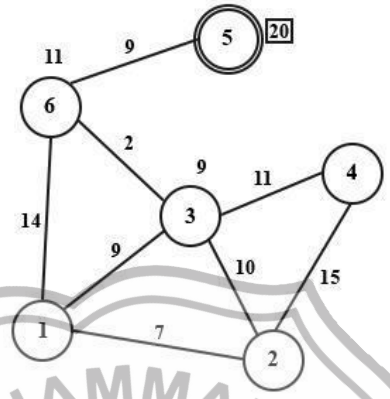
4. Perhitungan berlanjut pada gambar 2.7, dengan *node* 3 ditandai menjadi *node* yang telah terpilih. Dari semua *node* tetangga belum terlewati yang terhubung langsung dengan *node* terlewati, *node* selanjutnya yang ditandai menjadi *node* terpilih adalah *node* 6 karena nilai bobot yang terkecil, nilai 11 (9+2).



Gambar 2.7 Contoh Kasus *Dijkstra* – Langkah 4

5. *Node* 6 menjadi *node* terpilih, *Dijkstra* melakukan kalkulasi kembali, dan menemukan bahwa *node* 5 (*node* tujuan) telah tercapai lewat *node* 6. Pada Gambar 2.8, alur terpendeknya adalah 1-3-6-5, dan nilai bobot yang

didapat adalah 20 (11+9). Bila nodetujuan telah tercapai maka kalkulasi *Dijkstra* dinyatakan selesai.



Gambar 2.8 Contoh Kasus *Dijkstra* – Langkah 5