Pencarian Lintasan Terpendek pada *Map-based Services*

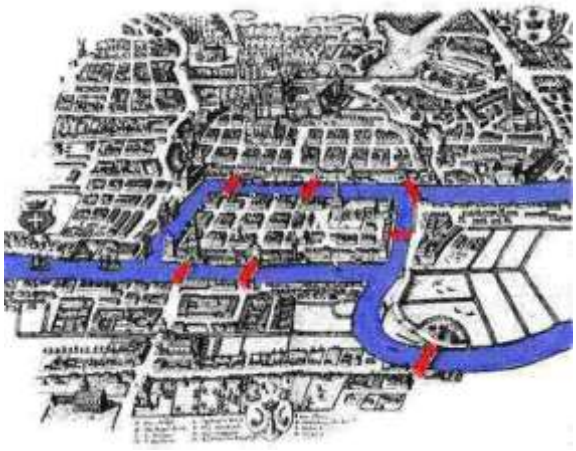
William Stefan Hartono - 13512098
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13512098@std.stei.itb.ac.id

***Abstrak* — Ilmu graf dapat diaplikasikan ke dalam berbagai hal. Salah satu penerapan yang berkaitan dengan lintasan/sirkuit graf adalah menentukan lintasan terpendek (*shortest path*). Graf yang digunakan adalah graf berbobot (*weighted graph*), yang setiap sisinya diberi suatu nilai atau bobot. Ada beberapaa macam persoalan mengenai pencarian lintasan terpendek dalam graf berbobot. Namun persoalan yang dibahas di makalah ini adalah pencarian lintasan terpendek dari simpul tertentu ke semua simpul yang lain.**

***Kata kunci* : *global positioning system* (GPS), graf berarah (*directed graph*), graf berbobot (*weighted graph*), lintasan terpendek (*shortest path*), *map-based services*, teori graf**

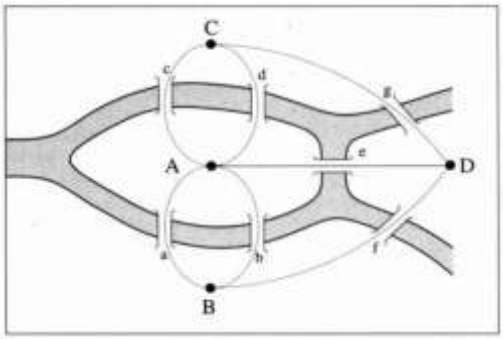
I. PENDAHULUAN

Teori graf merupakan pokok bahasan yang sudah tua usianya namun memiliki banyak penerapan sampai saat ini. Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan di antara objek-obek tersebut. Menurut catatan sejarah, pada tahun 1763, masalah jembatan Konigsberg adalah masalah yang pertama kali menggunakan graf. Konigsberg (sekarang bernama Kaliningrad) adalah sebuah kota yang berada di sebelah timur negara bagian Prussia, Jerman. Di kota tersebut terdapat sebuah sungai, bernama sungai Pregal yang mengalir mengitari pulau bernama pulai Kneiphof lalu bercabang menjadi dua buah anak sungai, seperti tampak pada gambar berikut ini :



Gambar 1 Jembatan Konigsberg

Ada tujub buah jembatan yang menghubungkan antara daratan yang satu dengan daratan yang lainnya. Masalah jembatan Konigsberg adalah : apakah mungkin untuk berjalan menyeberangi ketujuh jembatan tersebut tepat satu kali dan kembali lagi ketempat semula? Sebagian penduduk kota sepakah bahwa masalah tersebut tidak mungkin diselesaikan. Akan tetapi, mereka tidak dapat menjelaskan mengapa demikian kecuali dengan cara coba-coba. Masalah ini kemudian dipecahkan oleh seorang matematikawan yang berasal dari Swiss, Leonhard Euler. Euler berhasil menemukan jawaban dari masalah tersebut melalui sebuah pembuktian sederhana. Euler merepresentasikan masalah ini ke dalam sebuah graf dengan keempat daratan masing – masing menjadi sebuah simpul (*vertex*) dan ketujuh jembatan masing – masing menjadi sebuah sisi (*edge*).



Gambar 2 Representasi Jembatan Konigsberg dalam sebuah Graf

Solusi Euler adalah : tidak mungkin melalui ketujuh jembatan masing – masing tepat satu kali dan kembali lagi ke tempat semula jika derajat setiap simpul tidak genap. Yang dimaksud dengan derajat di sini adalah banyaknya sisi yang berhubungan dengan suatu simpul. Pada Gambar 2 tersebut dapat diketahui bahwa simpul B, C, dan D masing – masing berderajat tiga, sedangkan simpul A berderajat lima. Karena masing-masing simpul berderajat ganjil, maka menurut solusi Euler tidak mungkin masalah tersebut dapat diselesaikan.

Seiring dengan berkembangnya ilmu pengetahuan, teori Graf kini merupakan suatu materi utama dalam berbagai riset yang berhubungan dengan matematika, bidang kelistrikan, ilmu kimia, ilmu komputer, sosiologi, ekonomi, komunikasi, dan sebagainya. Umumnya, banyak permasalah yang dapat direpresentasikan dengan

menggunakan graf sebagai model. Contohnya seperti : rangkaian listrik, ikatan senyawa karbon, turnamen round robin, transaksi konkuren, pengujian program dengan *flowchart*, pencarian rute terpendek, pewarnaan peta, dan lain – lain.

II. TEORI DASAR

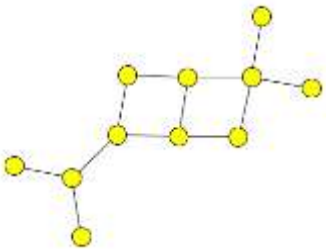
1. **Definisi Graf**

Graf adalah suatu pasangan himpunan (V, E), yang dalam hal ini :

V = himpunan tidak kosong dari simpul – simpul
= $\{v_1, v_2, \ldots, v_n\}$

E = himpunan sisi yang menghubungkan tiap simpul
= $\{e_1, e_2, \ldots, e_n\}$

Sehingga suatu graf G, dapat dituliskan dengan notasi G = (V, E). Biasanya sutu graf digambarkan sebagai kumpulan titik (simpul) yang dihubungkan oleh garis – garis (sisi).



Gambar 3 Contoh Graf

2. **Jenis Graf berdasar Orientasi Arah pada Sisi**

Sisi pada suatu graf dapat mempunyai orientasi arah ataupun tidak. Berdasarkan orientasi arah yang terdapat pada sisi, graf secara umum dibedakan menjadi dua jenis, yaitu :

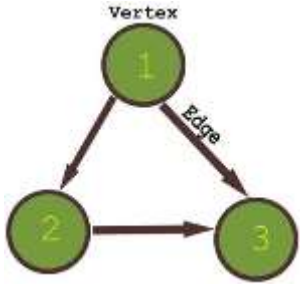
a. Graf Tak-berarah (*Undirected Graph*)

Graf tak-berarah adalah graf yang sisinya tidak mempunyai orientasi arah. Sehingga pada graf tak-berarah, urutan pasangan simpul yang dihubungkan oleh sebuah sisi tidak diperhatikan. Dengan kata lain, $(v_i, v_j) = (v_j, v_i)$.

Gambar 3 adalah sebuah contoh graf tak-berarah.

b. Graf Berarah (*Directed Graph*)

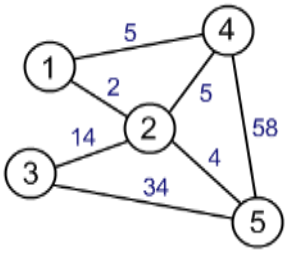
Graf yang setiap sisinya diberi orientasi arah disebut sebagai graf berarah. Sisi yang berarah tersebut sering disebut sebagai busur (*arc*). Pada graf berarah urutan pasangan simpul yang dihubungkan oleh sebuah sisi diperhatikan. Sehingga $(v_i, v_j) \neq (v_j, v_i)$. Untuk sebuah busur (v_i, v_j), simpul v_i dinamakan simpul asal (*initial vertex*) dan simpul v_j dinamakan simpul terminal (*terminal vertex*).



Gambar 4 Contoh Graf Berarah

3. **Graf Berbobot (Weighted Graph)**

Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga (bobot). Bobot pada tiap sisi dapat menyatakan apapaun. Misalnya bila tiap simpul merepresentasikan kota, maka bobot pada sisi dapat merepresentasikan jarak antar kota atau waktu tempuh, dan lain sebagainya.



Gambar 5 Contoh Graf Berbobot

III. PEMBAHASAN

***1.* *Global Positioning System* (GPS)**

GPS pertama kali dikembangkan oleh bidang militer di Amerika Serikat. Sebenarnya konsep GPS sudah ada sejak akhir tahun ‘60an namun aktualisasinya baru dimulai pada Februari 1978. Kemudian pada tahun 1989, sebuah perusahaan bernama *Magella Corp.* mengenalkan sebuah *hand-held receiver* GPS untuk pertama kalinya. Dan pada Maret 1996, presiden Amerika Serikat mengizinkan penggunaan GPS untuk kepentingan umum.

GPS adalah suatu sistem navigasi yang menyediakan informasi mengenai lokasi dan waktu pada segala kondisi cuaca. GPS bekerja dnegan menggunakan satelit – satelit yang mengorbit bumi untuk memancarkan sinyal yang kemudian akan diterima oleh sebuah *receiver*. Sinyal yang diterima oleh *receiver* kemudian akan diproses dan dikirimkan kembali ke satelit. Data yang sudah diproses tersebut disebut dengan data ephemeris.

Setelah menerima dan memroses sinyal dari satelit, *receiver* kemudian menggunakan data ephemeris tersebut untuk mengukur jaraknya dengan satelit. Dengan mengukur jaraknya dengan satelit, maka akan didapat koordinat lokasi dimana *receiver* tersebut berada.

2. *Map-based Services*

Seiring dengan berkembangnya zaman dan teknologi, maka layanan untuk mengakses berbagai kebutuhan pun semakin berkembang. Di zaman modern ini, tidak mengherankan jika kita bepergian dengan menggunakan berbagai alat transportasi yang tersedia, baik pribadi maupun umum. Masalahnya adalah jika kita ingin bepergian dengan kendaraan pribadi dan kita tidak mengetahui rute yang harus kita tempuh untuk mencapai tempat tujuan. Untuk mengatasi masalah itu, lahirlah *map-based service*.

Map-based service adalah sebuah layanan mengenai peta, navigasi, rute, dan lokasi. Pada umumnya *map-based services* dikelola disebuah jaringan yang harus diakses melalui internet.

Tampilan antarmuka pada sebuah *map-based service* biasanya berupa sebuah HTTP simpel yang menampilkan gambar peta (dengan ekstensi JPEG, PNG) yang bisa ditampilkan dalam sebuah browser. Program antarmuka biasanya juga mendukung kemampuan untuk memilih apakah gambar yang ditampilkan seharusnya ber-layer atau tidak.

Hingga saat ini sudah banyak bermunculan *map-based services* dalam berbagai bentuk seperti aplikasi untuk mobile maupun *mobile devices*. Sebut saja beberapa map-based services yang terkenal, seperti : Google Earth, Google Map, dan Waze.

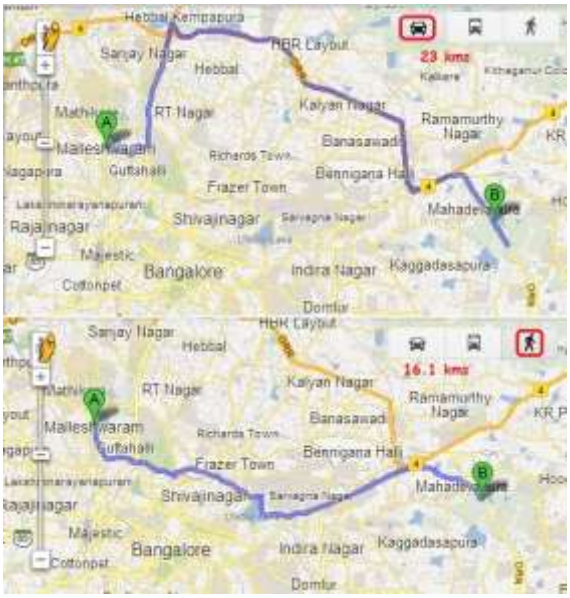
Gambar 6 Contoh *Map-based Services*

c. Implementasi Teori Graf dalam *Map-based Services*

Ketika kita menggunakan sebuah map-based services, apa pun itu, kita akan melihat titik – titik (simpul – simpul) tempat yang dapat kita tuju. Jika garis – garis yang menghubungkan titik – titik itu kita anggap sebagai sisi, maka kita akan mendapatkan sebuah graf.

Gambar 7 Contoh *Map-based Services* di kehidupan nyata

Walaupun mungkin kita tidak dapat melihat bobot yang terdapat pada tampilan antarmuka *map-based services*, setiap jalan yang direpresentasikan oleh sisi tersebut akan mempunyai sebuah bobot tersendiri. Secara otomatis, aplikasi *map-based services* akan mengirimkan sinyal secara terus menerus kepada satelit untuk meminta koordinat dimana dia berada melalui GPS. Kemudian, saat kita menentukan tempat yang akan kita tuju, *map-based service* akan melakukan sebuah perhitungan terhadap bobot pada tiap sisi dan mencari sisi yang mempunyai bobot paling kecil untuk mencapai tempat yang kita tuju. Selama melakukan perhitungan tersebut, *map-based service* akan mengikuti sebuah algoritma tertentu. Setiap *map-based service* akan mempunyai algoritma – algoritma yang berbeda. Akan tetapi algoritma yang umum digunakan dalam *map-based service* adalah algoritma Dijkstra.



Gambar 8 Contoh penggunaan aplikasi *Map-based Services*

d. Algoritma Dijkstra

Hingga saat ini, sudah terdapat banyak algoritma untuk mencari jalur terpendek. Namun algoritma yang paling terkenal adalah algoritma yang ditemukan oleh matematikawan Belanda bernama Edsger Wybe Dijkstra. Algoritma ini kemudian dinamakan sesuai dengan penemunya, yaitu algorimta Dijkstra. Algoritma asli Dijkstra tidak menggunakan *minimal priority queue* sehingga kompleksitasnya adalah $O(|V|^2)$, dengan V adalah jumlah simpul. Namun setelah dimodifikasi dengan implementasi *minimal priority queue* dan *Fibonacci heap*, kompleksitas algoritma Dijkstra meningkat menjadi $O(|E| + |V| \log |V|)$, dengan E adalah jumlah sisi. Algoritma Dijkstra yang baru ini kemudian menjadi algoritma tercepat untuk mencari jalur

terpendek graf berarah yang berbobot positif. Algoritma Dijkstra juga menjadi dasar pengembangan dari berbagai algoritma pencarian rute terpendek yang lain.

Pada awalnya algoritma Djikstra hanya diterapkan untuk mencari lintasan terpendek pada sebuah graf berarah. Meski demikian, algoritma Dijkstra juga dapat dipakai pada sebuah graf tak-berarah

Algoritma Dijkstra sendiri mencari lintasan terpendek dengan menggunakan prinsip *greedy algorithm*. *Greedy algorithm* adalah sebuah algoritma yang pada setiap langkahnya membuat pilihan optimal secara lokal dengan harapan akan menemukan solusi global yang optimal. Secara umum, *greedy algorithm* sendiri mungkin tidak menghasilkan solusi global yang optimal, namun akan menghasilkan solusi optimal lokal yang nilainya bisa mendekati solusi optimal global.

Greedy algorithm pada algoritma Dijsktra menyatakan bahwa pada setiap langkah, kita memilih sebuah sisi yang berbobot minimum dan memasukkannya ke dalam suatu himpunan solusi.

Berikut ini penulis akan menjelaskan langkah – langkah yang digunakan dalam algoritma Djikstra. Algoritma Djikstra mempunyai tiga parameter, yaitu sebuah graf G (G = {V, E}, l bobot sisi graf $\{l_e : e \in E\}$, dan simpul s (s € V}. Algoritma Dijkstra akan membentuk sebuah pohon merentang minimum yang mengandung rute terpendek dari simpul awal ke seluruh simpul lainnya.

Berikut adalah langkah – langkah dari algoritma Dijkstra :

a. Untuk setiap simpul yang dihubungkan dengan sisi, berikan nilai bobot awal. Jika ada simpul yang tidak dihubungkan dengan minimal sebuah sisi, beri nilai tak hingga (akan dianggap sebagai nilai terkecil). Beri nilai 0 untuk sisi dari suatu sisi ke sisi itu sendiri.
b. Tetapkan simpul awal sebagai simpul "aktif". Buat sebuah himpunan untuk seluruh simpul yang belum dikunjungi.
c. Untuk simpul "aktif", hitung seluruh jarak simpul yang bertetangga dengannya. Misal, jika simpul aktif A mempunyai nilai jarak 2 dan simpul A dihubungkan dengan simpul B melalui sebuah sisi berbobot 4, maka jarak A – B adalah 2 + 4 = 6. Jika jarak ini bernilai lebih kecil dari jarak sebelumnya maka pilih jarak tersebut.
d. Hilangkan simpul – simpul yang sudah dihitung jaraknya tersebut dari himpunan simpul yang belum dikunjungi.
e. Jika simpul tujuan ternyata termasuk dalam simpul yang sudah dikunjungi atau jika jarak terkecil di antara himpunan simpul yang belum dikunjungi adalah tak hingga maka algoritma sudah selesai.
f. Jika kondisi (e) tidak terpenuhi, pilih simpul dengan bobot terkecil yang belum dikunjungi dari himpunan yang tersisa. Tetapkan simpul tersebut sebagai simpul "aktif" dan ulangi dari langkah (c) hingga pada akhirnya kondisi di (e) terpenuhi atau seluruh simpul di himpunan simpul yang belum dikunjungi sudah habis.

Berikut penulis tampilkan algoritma Dijkstra dalam sebuah bahasa pemrograman :

Figure 4.8 Dijkstra's shortest-path algorithm.

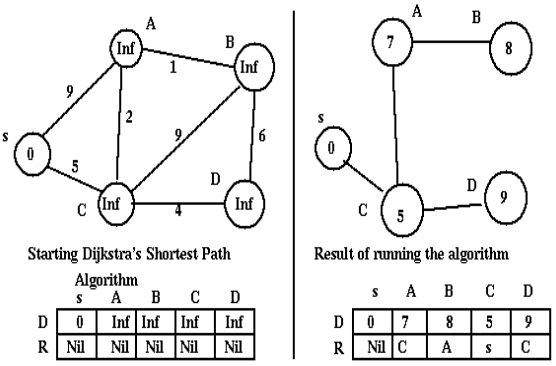
```
procedure dijkstra(G, l, s)
Input:  Graph G = (V, E), directed or undirected;
        positive edge lengths {l_e : e ∈ E}; vertex s ∈ V
Output: For all vertices u reachable from s, dist(u) is set
        to the distance from s to u.

for all u ∈ V:
   dist(u) = ∞
   prev(u) = nil
dist(s) = 0

H = makequeue(V)  (using dist-values as keys)
while H is not empty:
   u = deletemin(H)
   for all edges (u, v) ∈ E:
      if dist(v) > dist(u) + l(u, v):
         dist(v) = dist(u) + l(u, v)
         prev(v) = u
         decreasekey(H, v)
```

Gambar 9 Algoritma Dijkstra

Dan berikut penulis tampilkan contoh penggunaan dari algoritma Dijkstra untuk lebih memudahkan pemahaman :



Algorithm

	s	A	B	C	D
D	0	Inf	Inf	Inf	Inf
R	Nil	Nil	Nil	Nil	Nil

	s	A	B	C	D
D	0	7	8	5	9
R	Nil	C	A	s	C

Gambar 10 Hasil Penerapan Algoritma Dijkstra

Dari Gambar 9 di atas dapat dilihat hasil penerapan dari algoritma Dijkstra, dengan simpul s sebagai simpul awal.

a. Pada mulanya, kita menganggap bahwa jarak dari s ke simpul yang lain adalah tak hingga.
b. Bandingkan nilai sisi yang bertetangga dengan s (s-A bernilai 9 dan s-C bernilai 5). Dipilih s-C dengan nilai 5.
c. Sekarang gunakan C sebagai simpul aktif, bandingkan nilai sisi yang bertetangga dengan C, dengan tidak menggunakan lagi

simpul yang sudah dikunjungi (simpul s). Dari Gambar 9 terlihat bahwa C-A bernilai 2 dan C-B bernilai 9 dan C-D bernilai 4. Dipiliah C-A dengan nilai 2. Dengan ini di dapat pula jarak s-A yang terkecil yaitu jarak s-C ditambah jarak C-A = 5 + 2 = 7.

d. Sekarang gunakan A sebagai simpul aktif. Simpul yang bertetangga dengan A dan belum dikunjungi adalah simpul B, dengan nilai A-B = 1. Sehingga didapat pula jarak s-B yang terkecil yaitu jarak s-C ditambah jarak C-A ditambah jarak A-B = 5 + 2 + 1 = 8.

e. Sekarang gunakan B sebagai simpul aktif. Simpul yang bertetangga dengan B dan belum dikunjungi adalah simpul D, dengan nilai B-D = 6. Akan tetapi, karena harus ada jarak terdekat untuk s-D, kita juga perlu melihat dari simpul D. Yang bertetangga dengan simpul D selain simpul B adalah simpul C, dengan nilai D-C = 4. Karena nilai D-C lebih kecil daripada nilai D-B, maka kita memilih D-C. Sehingga didapat jarak s-D yang terkecil yaitu jarak s-C ditambah jarak C-D = 5 + 4 = 9.

f. Karena semua simpul sudah pernah dikunjungi, maka algoritma Dijkstra sudah selesai dan dihasilkan jarak terpendek dari simpul awal s ke semua simpul yang lain.

Berikut ini adalah algoritma Dijkstra dalam bentuk yang lain.

ALGORITHM 1 Dijkstra's Algorithm.

```
procedure Dijkstra(G: weighted connected simple graph, with
    all weights positive)
{G has vertices a = v0, v1, ..., vn = z and weights w(vi, vj)
    where w(vi, vj) = ∞ if {vi, vj} is not an edge in G}
for i := 1 to n
    L(vi) := ∞
L(a) := 0
S := ∅
{the labels are now initialized so that the label of a is 0 and all
    other labels are ∞, and S is the empty set}
while z ∉ S
begin
    u := a vertex not in S with L(u) minimal
    S := S ∪ {u}
    for all vertices v not in S
        if L(u) + w(u, v) < L(v) then L(v) := L(u) + w(u, v)
    {this adds a vertex to S with minimal label and updates the
        labels of vertices not in S}
end {L(z) = length of a shortest path from a to z}
```

Gambar 11 Algoritma Dijkstra (2)

Dengan program pada Gambar 10 sebagai acuan, kita dapat membuktikan bahwa setelah algoritma selesai dijalankan, dihasilkan jarak terpendek dari sebuah simpul awal ke simpul manapun. Untuk membuktikannya, kita dapat menggunakan indusi matematik sebagai berikut:

Pada pengulangan / iterasi ke-k :

i. bobot dari setiap simpul di dalam himpunan S adalah rute terpendek dari simpul awal a ke simpul tersebut

ii. bobot dari setiap simpul yang tidak di dalam himpunan S adalah rute terpendek dari simpul awal a ke simpul tersebut yang mengandung (selain simpul itu sendiri) simpul di dalam S

Ketika k = 0, S = ø, sehingga bobot dari rute terpendek dari simpul awal a ke simpul yang lain adalah ∞. Terbukti bahwa basis benar.

Misalkan v adalah simpul yang ditambahkan ke S pada iterasi ke-(k+1), v adalah simpul yang tidak ada di S pada iterasi ke k dengan bobot terkecil.

Sebelum iterasi ke-(k+1) dijalankan, simpul-simpul di S bernilai dengan rute terpendek dari simpul awal a (v juga termasuk). Jika hal ini tidak terjadi, pada akhir iterasi ke-k akan rute dengan panjang kurang dari $L_k(v)$ mengandung simpul yang tidak berada di S. Misalkan u adalah simpul pertama yang bukan anggota S. Maka akan ada rute dengan panjang kurang dari $L_k(v)$ dari a ke u yang dibentuk oleh anggota – anggota S. Hal ini saling bertentangan dengan pilihan v sebelumnya. Sehingga (i) benar untuk iterasi ke-(k+1)

Misalkan u adalah sebuah simpul yang tidak ada di dalam S setelah iterasi ke-(k+1). Rute terpendek dari a ke u yang mengandung anggota S bisa mengandung v atau tidak. Jika tidak mengandung v, maka panjangnya adalah $L_k(u)$. Jika mengandung v, maka itu adalah rute dari a ke v yang terpendek yang dibentuk oleh anggota-anggota di S selain v, diikuti oleh sisi dari v ke u. Sehingga panjangnya akan menjadi $L_k(v) + w(v, u)$. In imenunjukkan bahwa (ii) benar karena $L_{k+1}(u) = \min\{L_k(u), L_k(v) + w(v, u)\}$.

e. Kelemahan *Map-based Services*

Meskipun tampaknya *map-based services* sangat populer dan sangat praktis untuk digunakan di zaman modern ini, *map-based services* juga mempunyai beberapa kelemahan, beberapa di antaranya adalah sebagai berikut :

a. Ketergantungan alat navigasi pengguna pada sinyal

b. Kebanyakan *map-based services* hanya mengambil data berupa bobot jarak untuk mencari rute terpendek. Sehingga pengambilan rute terpendek belum tentu menghasilkan waktu tercepat untuk mencapai tempat tujuan. Misalnya saja, faktor kemacetan dan keadaan jalan tidak diperhitungkan.

c. Data lokasi tidak tepat atau tidak terdeteksi. Beberapa *server* meng-*update* data mereka beberapa bulan bahkan beberapa tahun sekali. Sehingga bisa saja jika suatu jalan tidak dipakai lagi atau jika ada suatu pembangunan jalan baru, data tersebut belum ada di *server*.

IV. KESIMPULAN

Konsep dan teori graf dapat diimplementasikan sebagai solusi dari masalah sehari-hari seperti pencarian rute terpendek.

Map-based service adalah sebuah layanan mengenai peta, navigasi, rute, dan lokasi. *Map-based service* sering digunakan untuk mencari rute terpendek antara dua tempat dalaam kehidupan sehari – hari.

Dalam *map-based service* tempat tujuan dapat dianggap sebagai sebuah simpul, sedangkan jalan –jalan yang menghubungkan tempat – tempat tersebut dapat dianggap dengan sebuah sisi.

Algoritma Dijkstra dapat digunakan dalam program *map-based service* sebagai algoritma yang mangkus dalam mencari rute terpendek antara dua tempat.

Meskipun *map-based service* sangat praktis dan mudah untuk digunakan, namun *map-based service* memiliki beberapa kelemahan.

V. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa, karena atas rahmat-Nya penulis dapat menyelesaikan makalah ini sebagaimana mestinya. Penulis juga mengucapkan terima kasih kepada Bapak Rinaldi Munir, dan Bu Harlili sebagai dosen pengajar mata kuliah IF2120 Matematika Diskrit yang telah memberi banyak pelajaran. Tidak lupa penulis juga mengucapkan terima kasih kepada semua pihak yang telah memberikan saran dan koreksi, yang tidak dapat disebutkan satu – persatu.

VI. DAFTAR REFERENSI

[1] Munir, Rinaldi. 2003. Matematika Diskrit Edisi Kedua. Bandung : Penerbit Informatika

[2] Rosen, Kenneth H. 2007. Discrete Mathematics and It's Applications, 6th Edition. New York : McGraw-Hill.

[3] http://www.maa.org/sites/default/files/images/cms_upload/Konigsberg_colour37936.jpg diakses tanggal 14 Desember 2013 pukul 10:00

[4] http://physics.weber.edu/carroll/honors_images/BarbasiBridges.jpg diakses tanggal 14 Desember 2013 pukul 10:11

[5] http://graphml.graphdrawing.org/primer/simple.png diakses tanggal 14 Desember 2013 pukul 10:30

[6] http://images.devshed.com/af/stories/an_insight/2.jpg diakses tanggal 14 Desember 2013 pukul 10:44

[7] http://web.cecs.pdx.edu/~sheard/course/Cs163/Graphics/graph6.png diakses tanggal 14 Desember 2013 pukul 10:55

[8] http://link.highedweb.org/files/map-pins.jpg diakses tanggal 14 Desember 2013 pukul 11:24

[9] http://www.emeraldinsight.com/content_images/fig/1270830302014.png diakses tanggal 14 Desember 2013 pukul 15:45

[10] http://www.cis.temple.edu/~wangp/3223-DA/Lecture/Figure-7-Dijkstra.JPG diakses tanggal 14 Desember 2013 pukul 15:47

[11] http://www.cis.temple.edu/~ingargio/cis307/readings/dijkst.gif diakses tanggal 14 Desember 2013 pukul 16:36

[12] http://www.theatlantic.com/technology/archive/2012/09/how-google-builds-its-maps-and-what-it-means-for-the-future-of-everything/261913/ diakses tanggal 14 Desember 2013 pukul 11:36

[13] http://googleblog.blogspot.com/2005/02/mapping-your-way.html diakses tanggal 14 Desember 2013 pukul 11:39

[14] http://www.opengeospatial.org/standards/wms diakses tanggal 14 Desember 2013 pukul 11:53

[15] http://www.loc.gov/rr/scitech/mysteries/global.html diakses tanggal 14 Desember 2013 pukul 11:57

[16] http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html diakses tanggal 14 Desember 2013 pukul 12:11

[17] http://www.ask.com/question/how-often-is-google-earth-updated diakses tanggal 14 Desember 2013 pukul 16:58

[18] http://www.theconnectivist.com/2013/07/how-google-tracks-traffic/ diakses tanggal 13 Desember 2013 pukul 21:13

[19] http://socialmediatoday.com/sites/socialmediatoday.com/files/waze.png diakses tanggal 14 Desember 2013 pukul 20:49

[20] http://oivietnam.com/wp-content/uploads/2013/10/google-maps-logo.jpg diakses tanggal 14 Desember 2013 pukul 20:55

[21] http://howtotechguru.com/shortest-driving-route-on-google-maps/726 diakses tanggal 14 Desember 2013 pukul 21:00

VII. PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 14 Desember 2013

William Stefan Hartono
13512098