Penerapan Algoritma Dijkstra pada aplikasi Waze

Willy and 13512070[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13512070@stei.itb.ac.id

***Abstrak*—Waze merupakan suatu aplikasi pada telepon genggam yang berguna mencari jalan terpendek. Dari sebuah posisi pada peta menuju tempat yang ingin dicapai dengan mempertimbangkan faktor kemacetan, kecelakaan, dan sebagainya akan dihasilkan olehnya dengan beberapa solusi jalur. Waze mendapatkan faktor-faktor tersebut dari pengguna yang menggunakan aplikasinya. Pengguna apliksi dapat ikut berpartisipasi memberikan informasi kepada Waze. Peta pada Waze juga didapatkan dari peta Google Maps, sehingga Waze tidak memerlukan satelit untuk memerperbaharui peta dunia.**

I. LATAR BELAKANG

Semakin maju era teknologi dunia, telepon genggam semakin murah terdapat di pasar. Telepon genggam sangat berpengaruh di kehidupan keseharian kita. Kita sering melihatnya dimana dan terdapat pada siapa pun. Dengan telepon genggam yang fungsi awalnya hanya untuk memberikan fungsi telepon agar komunikasi menjadi lebih mudah, telah berubah. Telepon genggam sekarang bisa diisikan dengan aplikasi yang canggih.

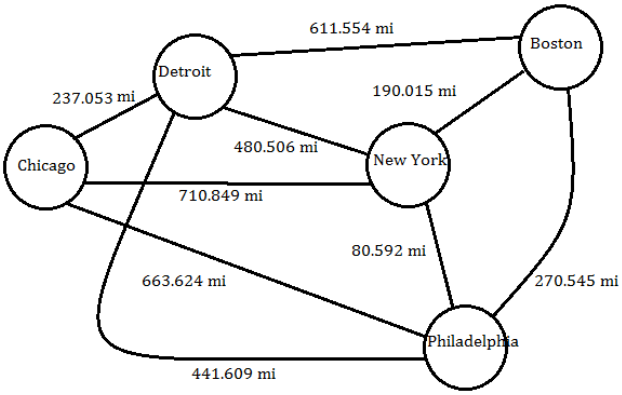
Internet adalah salah satu entitas yang berperan besar di era perkembangan teknologi informasi ini. Internet digunakan oleh banyak entitas untuk memajukan entitas tersebut sendiri. Telepon genggam adalah salah satu entitas yang menggunakannya. Pengguna sekarang dapat membuka halaman web dimana pun dengan telepon genggam masa kini. Pada awalnya hanya dapat membuka halaman web, namun makin berkembangnya telepon genggam, telepon semakin terintegrasi dengan jaringan internet sehingga dapat berhubungan dengan email pengguna.

Aplikasi pada telepon genggam juga semakin terintegrasi dan berhubung dengan jaringan internet. Dewasa ini semakin jarang terdapat aplikasi yang *independent* tanpa bantuan internet. Hal ini terjadi karena aplikasi yang diperlukan pengguna atau aplikasi yang diperbarui memperlukan data yang terbaru dan sangat besar, sehingga tidak mungkin telepon genggam pengguna dapat menyimpan data aplikasi yang semakin lama semakin besar yang tempat penyimpanan datanya tidak begitu besar bila dibandingkan dengan data penyimpanan komputer.

Fungsi peta adalah untuk mengetahui arah dan jalur yang tepat untuk mencapai suatu tempat tertentu. Di dalam peta tersebut terdapat berbagai macam informasi mengenai jalur yang ingin kita lewati. Informasi-informasi yang diberikan oleh peta adalah seperti dataran tinggi, dataran rendah, sungai, gunung dan lain lain. Telepon genggam sekarang dapat mendapatkan lokasi pengguna pada peta dengan aplikasi GPS yang sudah terdapat pada telepong pengguna. Peta digital itu menjadi salah satu kebutuhan yang kita perlukan ketika kita sedang ingin berpergian ke suatu tempat.

Peta digital yang sangat terkenal adalah Google Map. Siapapun telah memiliki kemewahan mencari petunjuk tentang cara untuk mendapatkan suatu tempat di Google Maps telah cenderung mengabaikan atau mengabaikan seluk-beluk bagaimana jalan semacam itu dihitung. Seluruh premis dari Google Maps menggunakan grafik raksasa besar dengan node dan tepi untuk mencari tahu cara tercepat atau terpendek untuk bepergian . Itu saja Google Maps adalah - grafik yang sangat besar dengan banyak simpul dan sisi. Namun, Data yang begitu besar akan diperlukan untuk menganalisis graf besar dan melacak semua simpul dan sisi, itu mengesankan bahwa perhitungan tersebut dapat dilakukan oleh Google dalam waktu singkat .

Bagian ini dari kertas oleh EW Dijkstra (http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf) yang ditulis pada tahun 1959 menguraikan metodologi dasar yang digunakan Google untuk membuat perhitungan ini. Grafik tersebut akan terlihat seperti berikut :



http://blogs.cornell.edu/

(Catatan : grafik ini disederhanakan dan tidak untuk skala)

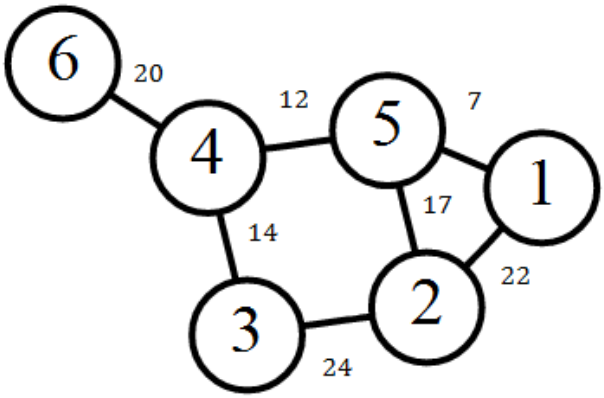
Grafik di atas menunjukkan lima kota besar di utara ,

bagian timur laut Amerika Serikat. Setiap simpul adalah sebuah kota dan setiap sisi dalam grafik merupakan jalur penerbangan jarak lurus yang , katakanlah , seekor burung gagak akan memakan waktu pergi dari satu node ke yang lain.

Kenyataannya, ini bukan jalan yang ideal karena kebanyakan orang yang menggunakan Google Maps bukanlah gagak dan mereka tidak dapat mengambil lurus penerbangan-jalan . Hal ini terutama terjadi karena jalur teoritis ini akan melewati air atau jalur yang tidak memiliki jalan. Karena grafik ini menunjukkan sisi hubungan antara setiap node, sangat jelas apa yang "jarak terpendek" antara dua node (karena jarak terpendek antara dua titik adalah garis lurus). Namun, tidak semua kota-kota besar secara langsung dihubungkan dalam pengertian ini . Ada banyak jalan yang diperlukan untuk diambil untuk benar-benar melintasi dari satu simpul ke simpul lain . Dalam kasus Google Maps , jalan ini akan diwakili oleh sisi pada grafik . Grafik yang digunakan oleh Google Maps akan mencakup lebih banyak node mulai dari kota-kota besar dan kota-kota yang lebih kecil untuk desa atau persimpangan bahkan jalan . Karena Google Maps dapat mengarahkan seseorang dari satu alamat ke alamat lain (untuk sebagian besar), maka bisa dibayangkan berapa banyak simpul ada dalam grafik lengkap. Ini akan memakan waktu lama untuk menghitung jalur terpendek dengan tangan - dan jumlah non - layak waktu melalui komputer . Namun, tindakan ini dilakukan cukup cepat oleh Google Maps berkali-kali dalam waktu singkat .

Rahasianya cukup sederhana, yaitu Algoritma Dijkstra yang dituangkan dalam makalah ini adalah salah satu dari banyak cara untuk menemukan jalan semacam itu , tetapi cara lain juga berlaku seperti diuraikan pada halaman wikipedia untuk penemuan jalan terpendek, http://en.wikipedia.org/wiki/Shortest_path_problem

Google dapat memfasilitasi masalah ini dengan meliputi simpul dalam simpul lain . Jika seseorang ingin menemukan jarak terpendek antara Philadelpha dan Boston, itu tidak akan layak atau bijaksana untuk memasukkan node seperti Seattle dalam grafik selama perhitungan. Google Maps akan mengambil sub-bagian dari seluruh grafik dan bekerja hanya pada bagian itu . Bagaimana jika kita memiliki simpul raksasa yang disebut "Timur Laut Amerika Serikat" yang meliputi grafik untuk semua kota di wilayah itu? Mungkin dalam simpul tersebut, kita dapat memiliki node lain seperti yang disebut "New York" yang berfokus hanya pada grafik dengan node yang masuk dalam kategori tersebut . Semakin dekat kota-kota yang satu sama lain (karena secara visual dapat dilihat melalui "zoom" fitur) maka semakin sedikit pekerjaan yang harus dilakukan . Mari kita mempertimbangkan grafik berikut ini :



(Catatan : ini juga tidak tertarik pada skala).

Grafik ini berasal dari halaman wikipedia diposting sebelumnya dengan sedikit modifikasi. Kita dapat mengambil setiap node menjadi simpul di Google Maps (sehingga bisa menjadi kota , persimpangan , atau rumah seseorang) dan keunggulan untuk menjadi jarak antara dua node (bisa jalan atau trotoar). Sebuah sisi bisa mewakili jalan, trotoar, atau sesuatu yang serupa. Dalam hal ini, menggunakan metodologi seperti algoritma Dijkstra yang dituangkan dalam paruh kedua makalahnya akan memungkinkan program seperti Google Maps untuk menghitung jarak terpendek . Dalam kasus grafik ini , kita sebagai orang-orang dapat melihat bahwa jarak terpendek dari node 3 dan 1 akan 3 - > 4 - > 5 - > 1 dan tidak 3 - > 2 - > 1 karena bagaimana nilai-nilai dari sisi dijumlahkan (14+12+7 < 24+22). Secara sistematis, program komputer yang menjalankan algoritma Dijkstra juga akan sampai pada kesimpulan yang sama dalam hal jarak terpendek . Hanya mengikuti langkah demi langkah petunjuk yang tercantum di koran bisa memandu Anda melalui bagaimana komputer akan melakukan masalah ini .

Meskipun masalah ini pada grafik yang lebih besar akan lebih sulit , Google memiliki beberapa server kinerja tinggi dan database yang semuanya beroperasi pada kecepatan ekstrim untuk melakukan perhitungan seperti ini hampir tidak ada waktu . Hal yang besar tentang hal itu adalah bahwa makna atau nilai dari sisi bisa berubah (bukan mewakili jarak , itu akan mewakili waktu perjalanan dengan mobil) belum algoritma itu sendiri tidak akan berubah. Atribut pada simpul atau sisi mewakili adalah sewenang-wenang dan hanya sebuah parameter, ditetapkan untuk kenyamanan pengguna.

Aplikasi ini grafik di dunia nyata adalah contoh kecil namun satu super penting yang umum bagi kehidupan masyarakat. Banyak orang meremehkan kekuatan grafik.

Waze merupakan salah satu aplikasi yang yang menggunakan semua entitas yang dibahas di atas. Aplikasi ini menggunakan GPS yang terdapat pada telepon genggam pengguna untuk menentukan lokasi pegguna sekarang. Setelah dia mengetahui posisinya yang sekarang, dia mengambil informasi mengenai peta sekitarnya menggunakan Google Map, misalnya di jalan martadinata macet. Informasi seperti kemacetan, terdapat polisi, terdapat perbaikan jalan adalah informasi yang diberikan oleh pengguna lainnya. Informasi tersebut

digunakan aplikasinya untuk mencari jalan terpendek yang ingin dituju oleh pengguna. Jalan terpendek dalam kasus ini adalah jalan yang memerlukan waktu paling sedikit untuk menuju tujuan dengan melihat faktor kemacetan.

Segala pengguna yang sering menggunakan aplikasi ini untuk mencari jalur yang ingin dilewatinya kebanyakan tidak mempedulikan cara aplikasinya untuk menentukan jalur terpendek untuk menuju tujuan pengguna. Padahal peta adalah graf yang memiliki simpul dan sisi yang sangat besar. Namun dengan diberikan dengan data yang cukup besar yang diperlukan untuk dianalisis dan disimpan, aplikasi ini sangatlah mengagumkan bisa menkalkulasi dalam waktu yang singkat.

II. LANDASAN TEORI

Graf adalah struktur disrit yang mengandung simpul dan sisi yang menghubungkan antar simpul. Graf terdapat berbagai jenis, tergantung terhadap sisinya apakah memiliki arah, terhadap beberapa sisi sekaligus apakah dapat menghubungkan simpul yang sama juga, dan perputaran di suatu simpul diberikan. Kebanyakan segala masalah dapat diselesaikan setelah model masalah tersebut dilihat dari perspektif graf. Contohnya, graf dapat menjelaskan kompetisi antar spesies dalam ekologi nische, bagaimana graf direpresentasi untuk menjelaskan siapa yang mempengaruhi siapa dalam struktur organisasi, and bagaimana graf digunakan untuk memperhitungkan hasil dari pertandingan round-robin. Graf digunakan juga untuk menjelaskan hubungan antar 2 orang, kolaborasi antar peniliti, telepon dari beberapa nomor telepon, dan hubungan antar halaman website.

Menggunakan model graf, kita dapat menentukan apakah dimungkinan untuk berjalan menyusuri semua jalan-jalan di kota tanpa turun jalan dua kali, dan kita dapat menemukan jumlah warna yang diperlukan untuk mewarnai wilayah peta. Graf dapat digunakan untuk menentukan apakah sirkuit dapat diimplementasikan pada sirkuit planar papan. Dapat dibedakan antara dua senyawa kimia dengan rumus molekul yang sama tetapi struktur yang berbeda dengan menggunakan graf . Kita bisa menentukan apakah dua komputer yang terhubung dengan link komunikasi menggunakan model graph jaringan komputer .

Graf dengan bobot ditugaskan untuk ujung-ujungnya dapat digunakan untuk memecahkan masalah seperti menemukan jalur terpendek antara dua kota di jaringan.

Definisi formal dari graf adalah

Sebuah graf G adalah (V, E) mengandung
V, dimana adalah himpunan simpul yang tidak kosong
E, dimana adalah himpunan sisi.
Setiap sisi terdapat 1 atau 2 simpul yang terhubung. Simpul − simpul tersebut disebut dengan endpoints.
Sebuah sisi selalu terhubungan dengan endpoints nya.

Himpunan simpul, V, dari sebuah graf mungkin tak terhingga banyaknya. Himpunan simpul dari graf yang tak terhingga atau himpunan sisi dari graf yang tak terhingga biasanya disebut graf tak terbatas. Banyak simpul dan sisi yang terbatas pada sebuah graf biasanya disebut dengan graf terbatas.

Sebuah graf dimana setiap sisi behubung dengan 2 simpul yang berbeda dan tidak ada 2 sisi yang yang terhubung dengan simpul-simpul yang sama disebut dengan graf yang mudah. Di dalam graf yang mudah, setiap sisi yang terasosiasi dengan simpul-simpul dan tidak ada sisi yang hubungan asosiasi dengan simpul tersebut sama. Secara konsekuen, jika ada sebuah sisi pada sebuah graf simple yang terhubung dengan {u, v}, maka kita juga dapat menyebutkan bahwa {u, v} adalah sisi dari graf tersebut.

Berdasarkan orientasi arah pada sisi, maka secara umum graf dibedakan menjadi 2 jenis :

1. Graf tak - berarah (undirected graph)

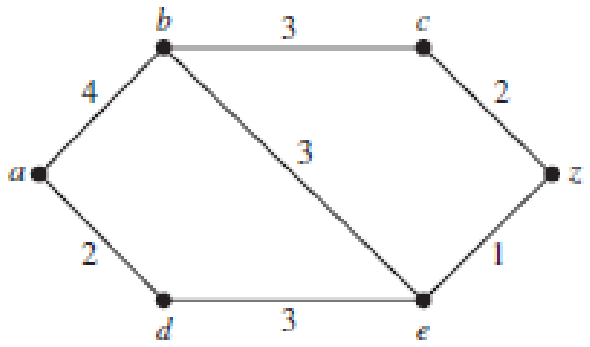
Graf yang sisinya tidak mempunyai orientasi arah disebut graf tak - berarah. Pada graf tak - berarah, urutan pasangan simpul yang dihubungkan oleh sisi tidak diperhatikan. Jadi, (vj, vk) = (vk, vj) adalah sisi yang sama.

2. Graf berarah (directed graph atau di graf)

Graf yang setiap sisinya diberikan orientasi arah disebut sebagai graf berarah. Kita lebih suka menyebut sisi berarah dengan sebutan busur (arc). Pada graf berarah, (vj, vk) _ (vk, vj). Untuk busur(vj, vk), simpul vj dinamakan simpul asal (initial vertex) dan simpul vk dinamakan simpul terminal(terminal vertex).

Masalah jalan terpendek

Banyak masalah dapat dimodelkan menggunakan grafik dengan bobot ditugaskan untuk pinggirannya. Sebagai ilustrasi, mempertimbangkan bagaimana sistem airline dapat dimodelkan. Terdapat contoh model grafik dasar dengan mewakili kota dengan simpul dan penerbangan pada bagian tepinya. Masalah yang melibatkan jarak dapat dimodelkan dengan menetapkan jarak antara kota-kota ke pinggiran. Masalah yang melibatkan waktu penerbangan dapat dimodelkan dengan menetapkan jadwal penerbangan ke sisi. Masalah yang melibatkan harga dapat dimodelkan oleh Algoritma Djikstra. Algoritma Greedy yang ditemukan oleh matematikawan belan Edsger Dijkstra pada tahun 1959. Algoritma ini hanya bisa digunakan untuk graf yang tidak siklis dan tidak terdapat sisi yang berbobot negatif.



Contoh graf

Contoh tersebut menggambarkan prinsip-prinsip umum yang digunakan dalam algoritma Dijkstra. Perhatikan bahwa terpendek path dari a ke z bisa ditemukan oleh pendekatan kekerasan dengan memeriksa panjang setiap path dari a sampai z. Namun, pendekatan kekerasan ini tidak praktis bagi manusia dan bahkan untuk komputer untuk graf dengan sisi yang berjumlah besar. Kita sekarang akan mempertimbangkan masalah umum menemukan panjang jalur terpendek antara dan z dalam graf terhubung berbobot sederhana diarahkan . Dijkstra Algoritma hasil oleh menemukan panjang jalur terpendek dari a ke simpul pertama , panjang jalur terpendek dari a ke simpul kedua , dan seterusnya, sampai panjang jalur terpendek dari a ke z ditemukan. sebagai sisi manfaat, algoritma ini mudah diperluas untuk menemukan panjang lintasan terpendek dari a ke semua simpul lain dari grafik, dan bukan hanya untuk z . Algoritma ini bergantung pada serangkaian iterasi . Satu set dibedakan simpul dibangun dengan menambahkan satu titik pada setiap iterasi . Sebuah prosedur pelabelan dilakukan pada setiap iterasi . di prosedur pelabelan ini, w simpul diberi label dengan panjang jalur terpendek dari a ke w yang hanya berisi simpul sudah di set dibedakan . Vertex ditambahkan ke set dibedakan adalah salah satu dengan label minimal antara mereka simpul belum di set.

Algoritma Dijkstra.

Ini dimulai dengan pelabelan dengan 0 dan simpul lain dengan ∞ . Kami menggunakan L_0 notasi (a) = 0 dan L_0 (v) = ∞ untuk label ini sebelum setiap iterasi telah terjadi (subskrip 0 singkatan "0" iterasi). Label-label ini panjang jalur terpendek dari a ke simpul , di mana jalan hanya berisi simpul a. (Karena tidak ada jalan dari a ke simpul berbeda dari ada , ∞ adalah panjang jalur terpendek antara dan simpul ini.)

Dijkstra Algoritma hasil dengan membentuk satu set dibedakan dari simpul . Biarkan S_k menunjukkan ini ditetapkan setelah iterasi k prosedur pelabelan . Kita mulai dengan $S_0 = \emptyset$. Set S_k terbentuk dari S_{k-1} dengan menambahkan simpul u tidak di S_{k-1} dengan label terkecil.

Setelah u ditambahkan ke S_k, kami memperbarui label semua simpul tidak S_k, sehingga $L_k(v)$, yang label v titik pada tahap ke-k , adalah panjang jalur terpendek dari a ke v yang berisi simpul hanya di k (yaitu, simpul yang sudah di set dibedakan bersama-sama dengan u).

Perhatikan bahwa cara kita memilih vertex u untuk menambah S_k pada setiap langkah adalah pilihan yang optimal pada setiap langkah , membuat ini sebuah algoritma yang tamak. Biarkan v menjadi simpul yang tidak ada di S_k. Untuk memperbarui label v , perhatikan bahwa L_k (v) adalah panjang dari terpendekpath dari a ke v yang hanya terdapat simpul di S_k . Update dapat dilakukan secara efisien ketika observasi ini digunakan : Sebuah jalur terpendek dari a ke v hanya berisi elemen S_k adalah baik jalur terpendek dari a ke v yang hanya berisi unsur S_{k-1} (yaitu, simpul dibedakan tidak termasuk u) , atau merupakan jalur terpendek dari a ke u pada (k-1) st panggung dengan tepi {u, v} menambahkan. Dengan kata lain,

$L_k(a, v) = \min\{L_{k-1}(a, v) , L_{k-1}(a, u)+w(u, v)\}$, dimana w (u , v) adalah panjang dari tepi dengan u dan v sebagai titik akhir. Prosedur ini diiterasi berturut-turut menambahkan simpul ke set dibedakan sampai z ditambahkan . Ketika z ditambahkan ke set dibedakan , label adalah panjang jalur terpendek dari a sampai z.

Pembuktian Algoritma Dijkstra

Asumsikan bahwa hipotesis induktif berlaku untuk iterasi ke-k . Biarkan v menjadi simpul ditambahkan ke S pada (k + 1) st iterasi , sehingga v adalah vertex tidak di S pada akhir iterasi k dengan label terkecil (dalam kasus dasi, setiap simpul dengan label terkecil mungkin digunakan) .

```
procedure Dijkstra(G: weighted connected simple graph, with
      all weights positive)
{G has vertices a = v0, v1, ..., vn = z and lengths 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
      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}
return L(z) {L(z) = length of a shortest path from a to z}
```

Psuedocode Algoritma Dijkstra

Dari hipotesis induktif kita melihat bahwa simpul di S sebelum (k+1) iterasi diberi label dengan panjang jalur terpendek dari. Juga , v harus diberi label dengan panjang jalur terpendek untuk itu dari. Jika ini tidak terjadi, pada akhir iterasi k akan ada jalan panjang kurang dari $L_k(v)$ mengandung simpul tidak di S [karena $L_k(v)$ adalah panjang jalur terpendek dari a ke v hanya berisi simpul di S setelah iterasi k]. Misalkan u titik pertama tidak di S di jalan semacam itu. Ada jalan dengan panjang kurang dari $L_k(v)$ dari ke u hanya berisi simpul dari S. Hal ini bertentangan dengan pilihan v Oleh karena itu, (i) memegang pada akhir (k+1) iterasi .

Misalkan u vertex tidak di S setelah k + 1 iterations. Jalur terpendek dari a ke u hanya berisi elemen dari S baik mengandung v atau tidak. Jika tidak mengandung v, maka dengan hipotesis induktif panjangnya adalah $L_k(u)$. Jika tidak mengandung v, maka harus terdiri dari jalan dari a ke v panjang sesingkat mungkin mengandung unsur S selain v, diikuti oleh tepi dari v ke u. Dalam hal ini , panjangnya akan $L_k(v)+w(v, u)$. Hal ini menunjukkan bahwa (ii) benar, karena $L_{k+1}(u) = \min\{L_k(u), L_k(v)+w(v, u)\}$.

III. Analisis dan Pembahasan

Pada awalnya peta yang terdapat pada aplikasi waze berasal dari Google Map. Dari peta tersebut, aplikasi waze mengkonversi setiap jalan menjadi sisi pada graf dan setiap cabang pada jalan menjadi sebuah simpul. Setiap jalan besar dan kecil terdapat asumsi kecepatan rata-rata untuk sebuah kendaraan sehingga dimungkinkan untuk mendapat waktu estimasi sebuah kendaraan mencapai tujuannya. Informasi yang diberikan oleh pengguna dapat mempengaruhi kecepatan rata-rata yang diasumsikan. Informasi kemacetan sendiri memiliki tingkat kemacetan, yang mengartikan bahwa tingkat kemacetan yang semakin tinggi mempengaruhi kecepatan rata-rata sebuah kendaraan pada jalan tersebut. Bobot sisi yang merupakan konversi dari jalan bukanlah jarak dari sebuah simpul ke simpul lainnya melainkan waktu yang diperlukan dari sebuah simpul menuju simpul lainnya dengan kecepatan rata-rata yang diberikan di sisi tersebut.

Tampilan Aplikasi Waze

Cara kerja algoritma Dijkstra pada Waze adalah :

1. Berasumsi posisi sekarang berada di simpul.
2. Memberikan nilai 0 pada simpul awal dan sisanya adalah tak terhingga karena kita tidak mengetahui waktu yang diperlukan untuk menuju simpul lainnya.
3. Untuk simpul posisi sekarang, carilah semua waktu tentatif yang diperlukan menuju simpul tetangganya. Bila waktu yang didapatkan sebelumnya lebih kecil dari hasil yang didapat sekarang maka tetap simpan hasil yang sebelumnya dan sebaliknya.
4. Setelah kita mencari tentatif waktu yang diperlukan ke semua simpul tetangga dari simpul sekarang, maka simpul sekarang dihapus dari himpunan simpul yang belum pernah dikunjungi. Simpul yang sudah pernah dikunjungi tidak akan pernah digunakan lagi.
5. Bila simpul tujuan telah dikunjungi atau waktu tentatif yang paling kecil pada simpul yang terdapat di himpunan simpul yang belum dikunjungi adalah tak terhingga, maka algoritma ini berhenti.
6. Pilih simpul yang memiliki waktu tentatif yang paling kecil, dan simpul tersebut menjadi simpul sekarang. Ulangi langkah 3.

V. Kesimpulan

Algoritma dijkstra digunakan untuk mencari jalur tercepat pada aplikasi Waze. Waze mengubah faktor-

faktor tersebut menjadi memperlambat kecepatan mobil atau memperpanjang jarak dari simpul ke simpul lainnya.

REFERENCES

[1] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2013-2014/Makalah2013/MakalahIF2211-2013-087.pdf diakses pada hari Minggu, 18 Mei 2014

[2] Kenneth H. Rosen, Discrete Mathematics and Its Application, Mc Graw Hill, 1999.

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, 19 Mei 2014

Willy dan 13512070