Penerapan Algoritma Dijkstra dalam Penentuan Jalur Tercepat pada Google Maps

Nira Rizki Ramadhani 13516018
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13516018@std.stei.itb.ac.id

***Abstrak*—Dewasa ini, efisiensi adalah hal yang penting dalam berbagai aspek di kehidupan, salah satunya adalah dalam menentukan jalur tercepat saat ingin pergi dari suatu tempat ke tempat lain. Berbagai macam perkembangan teknologi sudah memungkinkan kita untuk mendapatkan efisiensi dalam hal tersebut, yakni dengan menggunakan Google Maps. Namun, dibutuhkan pula algoritma yang tepat untuk menghasilkan rute yang efisien. Makalah ini membahas penerapan algoritma Dijkstra dalam penentuan jalur tercepat pada Google Maps dengan merepresentasikan peta pada Google Maps sebagai sebuah graf berarah yang berbobot. Algoritma Dijkstra digunakan untuk menemukan jalur terpendek berdasarkan bobot terkecil dari satu titik ke titik lainnya.**

***Kata Kunci*—algoritma Dijkstra, graf, Google Maps, rute**

I. PENDAHULUAN

Dewasa ini, penggunaan aplikasi berbasis internet sudah menjadi "santapan" sehari-hari para pengguna gawai di dunia. Aplikasi berbasis internet kerap digunakan untuk memuat kegiatan sehari-hari, mulai dari tulisan, gambar, video hingga lokasi. Kini, hampir seluruh gawai memiliki fitur lokasi yang berguna untuk melacak letak gawai itu berada.

Salah satu kegunaan fitur tersebut yang paling berpengaruh adalah saat pengguna gawai menggunakan aplikasi sistem navigasi untuk mengetahui jalur menuju lokasi tertentu. Google Maps merupakan system navigasi yang sangat berguna untuk menjelajahi jalan di seluruh dunia agar sampai ke tujuan secepat mungkin. Salah satu cara menyelesaikannya adalah dengan memprediksi lalu lintas dan menghitung rute tercepat.

Selain itu, maraknya penggunaan transportasi *online* juga kerap membuat pengguna gawai terus mengaktifkan fitur lokasi pada gawainya. Aplikasi transportasi *online* memanfaatkan Google Maps untuk memberikan petunjuk lokasi pengguna dan pengemudi serta jalur yang dilalui untuk sampai ke lokasi tujuan. Setelah itu, aplikasi akan mencarikan pengemudi dengan lokasi terdekat dari pengguna yang memesan transportasi *online* tersebut. Pada dasarnya, aplikasi tersebut memiliki prinsip yang sama dengan Google Maps dalam menunjukkan jalur terdekat.

Namun, pernahkah kita terpikir bagaimana Google Maps dapat menunjukkan jalur terdekat dari lokasi awal menuju lokasi tujuan? Dalam kenyataannya, aplikasi Google Maps akan mengirimkan *Real-time Data* (RTD) ke Google kemudian informasi tersebut akan digunakan untuk mengkalkulasi banyak kendaraan pada suatu jalan dan seberapa cepat kendaraan tersebut melaju. Google pun akan menyimpan riwayat pola lalu lintas di jalan tertentu sehingga dapat memprediksi kondisi lalu lintas dalam waktu tertentu[4].

Dalam proses Google Maps memperoleh jalur terdekat menuju suatu tempat tujuan ini, algoritma Dijkstra dapat menjadi salah satu solusi yang dapat diterapkan dalam menentukan jalur terdekat dari suatu rute. Perlu diketahui bahwa rute pada peta merupakan salah satu contoh penerapan dari teori graf. Selain itu, perlu diketahui pula bahwa pengertian dari algoritma menurut KBBI adalah prosedur sistematis untuk memecahkan masalah matematis dalam langkah-langkah terbatas[5].

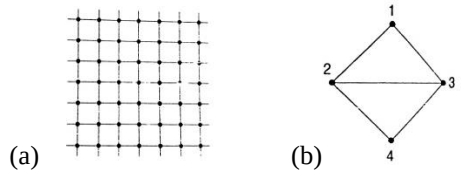
II. TEORI DASAR

A. Graf

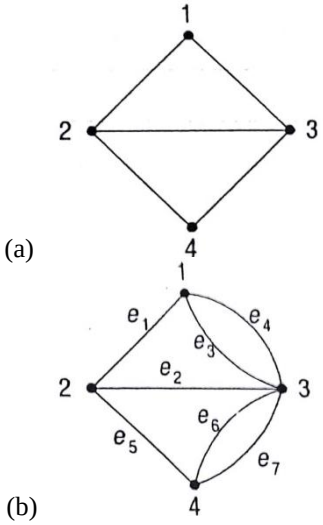
Teori Graf merupakan sebuah pokok bahasan yang sudah sangat tua namun memiliki banyak terapan hingga saat ini. Graf digunakan untuk merepresentasikan berbagai objek diskrit dan hubungan antara objek-objek tersebut. Representasi visual dari graf, yakni sebuah objek dinyatakan dengan noktah, titik, atau bulatan[1].

Graf merupakan himpunan tidak kosong yang terdiri atas pasangan simpul-simpul (*vertices* atau *nodes*) dan sisi-sisi (*edges* atau *arcs*) yang dituliskan dengan notasi G = (V,E), yang dalam hal ini V himpunan simpul dan E merupakan himpunan sisi[1]. Graf memiliki struktur berupa beberapa simpul yang terhubung oleh beberapa sisi.

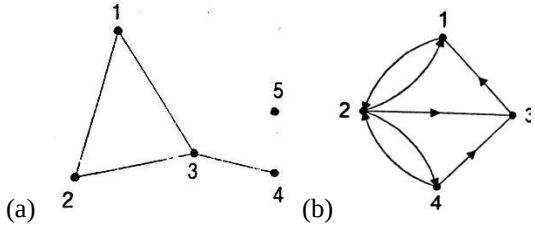
Graf dapat dikelompokkan berdasarkan beberapa jenis atau kategori bergantung pada sudut pandang pengelompokkannya. Berdasarkan ada tidaknya sisi ganda atau gelang, graf digolongkan menjadi dua jenis, yaitu graf sederhana (tidak ada gelang maupun sisi-ganda) dan graf tak-sederhana (mengandung sisi ganda atau gelang). Selanjutnya, berdasarkan orientasi arah pada sisi, graf dibedakan atas dua jenis, yaitu graf tak-berarah (sisinya tidak mempunyai orientasi arah) dan graf berarah (setiap sisinya diberikan orientasi arah). Selain itu, berdasarkan jumlah simpul pada suatu graf, terdapat dua jenis graf, yaitu graf berhingga (memiliki jumlah simpul n, berhingga) dan graf tak-berhingga (memiliki jumlah simpul n, tak berhingga banyaknya)[1].



Gambar 1. (a) Graf Tak Berhingga (b) Graf Berhingga
(Sumber: [1])



Gambar 2. (a) Graf Sederhana (b) Graf Tak-Sederhana
(Sumber: [1])



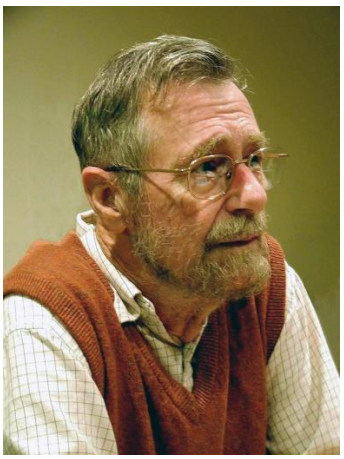
Gambar 3. (a) Graf Tak-Berarah (b) Graf Berarah
(Sumber: [1])

Graf memiliki beberapa terminologi (istilah), di antaranya:

1. Bertetangga (*Adjacent*)
   Simpul *u* dan *v* pada graf tak-berarah *G* dikatakan bertetangga apabila keduanya terhubung langsung dengan sebuah sisi, misalkan *(u,v)*, dengan *(u,v)* merupakan sisi pada graf *G*[1].
2. Bersisian (*Incident*)
   Untuk sembarang sisi *e = (u, v)*, sisi *e* dikatakan bersisian dengan simpul *u* dan simpul *v*[1].
3. Simpul Terpencil (*Isolated Vertex*)
   Apabila sebuah simpul tidak memiliki sisi yang bersisian dengannya atau tidak satupun bertetangga dengan simpul-simpul lainnya, maka simpul tersebut disebut simpul terpencil[1].
4. Graf Kosong (*Empty Graph* atau *Null Graph*)
   Apabila sebuah graf memiliki himpunan sisi berupa himpunan kosong, maka graf tersebut disebut graf kosong atau dapat ditulis sebagai N_n, dengan *n* merupakan jumlah simpul[1].
5. Derajat (*Degree*)
   Jumlah sisi yang bersisian dengan simpul pada graf tak-berarah disebut derajat suatu simpul[1].
6. Lintasan (*Path*)
   Barisan berselang-seling simpul-simpul dan sisi-sisi yang berbentuk v_o, e_1, v_1, e_2, v_2,... , v_{n-1}, e_n, v_n sedemikian sehingga $e_1 = (v_o, v_1)$, $e_2 = (v_1, v_2)$,... , $e_n = (v_{n-1}, v_n)$ adalah sisi-sisi dari graf *G* disebut lintasan yang panjangnya n dari simpul awal v_o ke simpul tujuan v_n[1].
7. Siklus atau Sirkuit
   Lintasan yang berawal dan berakhir pada simpul yang sama disebut siklus atau sirkuit[1].
8. Terhubung (*Connected*)
   Jika setiap pasang simpul v_i dan v_j di dalam himpunan V pada suatu graf tak-berarah terdapat sebuah lintasan dari v_i ke v_j serta lintasan dari v_j ke v_i disebut graf terhubung, jika tidak memenuhi syarat tersebut maka disebut graf tak-terhubung (*disconnected*)[1].
9. Upagraf dan Upagraf Merentang
   Upagraf *G* merupakan sebuah graf yang tiap simpul dan sisinya bagian dari graf *G*. Upagraf Merentang *G* merupakan upagraf yang memiliki semua simpul graf *G*[1].
10. *Cut-Set*
    Cut-Set dari graf terhubung *G* adalah himpunan sisi yang apabila dihapuskan dari *G* akan menyebabkan *G* tidak terhubung[1].
11. Graf Berbobot
    Apabila sebuah graf memiliki sisi-sisi yang mempunyai harga atau bobot, maka graf tersebut disebut graf berbobot[1].

B. Algoritma Dijkstra

Nama Algoritma Dijkstra diambil dari nama penemunya, yaitu Edsger Wybe Dijkstra. Ia adalah seorang ahli komputer asal Belanda yang lahir di Rotterdam pada tanggal 11 Mei 1930 dan meninggal dunia pada tanggal 6 Agustus 2002[8].



Gambar 4. Edsger Wybe Dijkstra
(Sumber: [8])

Algoritma ini bertujuan untuk menemukan jalur terpendek berdasarkan bobot terkecil dari satu titik ke titik lainnya. Misalkan sebuah titik menggambarkan gedung dan garis menggambarkan jalan, maka algoritma Dijkstra melakukan perhitungan terhadap semua kemungkinan bobot terkecil dari setiap titik[3].

Pertama-tama, tentukan titik yang akan menjadi simpul awal kemudian berikan bobot atau jarak dari simpul awal menuju simpul lain terdekat satu per satu, algoritma Dijkstra akan melakukan pengembangan pecarian dari satu titik ke titik lain dan ke titik selanjutnya secara bertahap[3].

Cara kerja algoritma Dijkstra untuk mencari lintasan terpendek dari simpul a ke simpul z dalam sebuah graf berbobot adalah sebagai berikut.

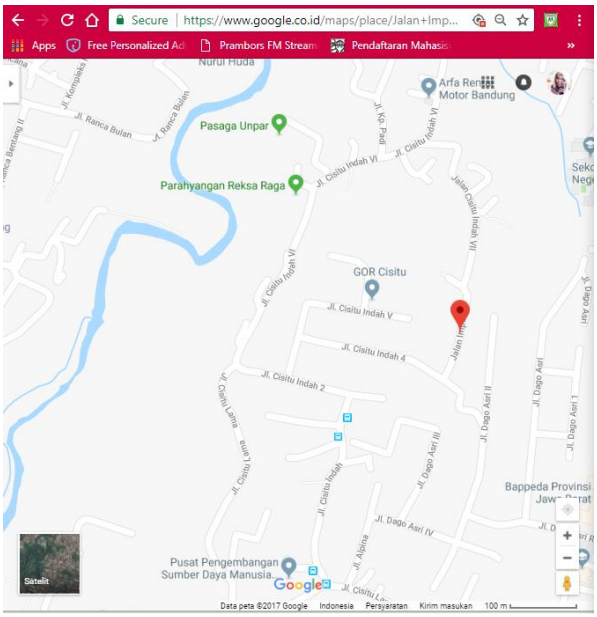
1. Beri nilai bobot atau jarak untuk setiap simpul ke simpul lainnya, lalu inisialisasikan nilai 0 pada simpul awal dan nilai tak-hingga terhadap simpul lain (belum terisi).
2. Inisialisasi semua simpul yang belum pernah dijamah dan inisialisasi simpul awal sebagai "Simpul Keberangkatan".
3. Dari simpul keberangkatan, pertimbangkan simpul tetangga yang belum terjamah dan hitung jaraknya dari simpul keberangkatan. Sebagai contoh, apabila simpul keberangkatan A ke B memiliki bobot jarak 10 dan dari B ke C memiliki bobot jarak 5, maka jarak ke C melewati B menjadi 10 + 5 = 15. Jika jarak ini lebih kecil daripada jarak sebelumnya (yang telah terekam), maka tuliskan ulang data jarak dengan jarak yang baru, yaitu dengan jarak yang lebih kecil.
4. Setelah selesai mempertimbangkan setiap jarak terhadap simpul tetangga, tandai simpul yang sudah terjamah sebagai "Simpul Terjamah". Simpul terjamah tidak akan pernah ditinjau kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.
5. Inisialisasikan "Simpul Belum Terjamah" dengan jarak terkecil (dari simpul keberangkatan) sebagai "Simpul Keberangkatan" selanjutnya dan dilanjutkan kembali ke butir 3[3].

Berikut adaah *pseudocode* dari algoritma Dijkstra[3].

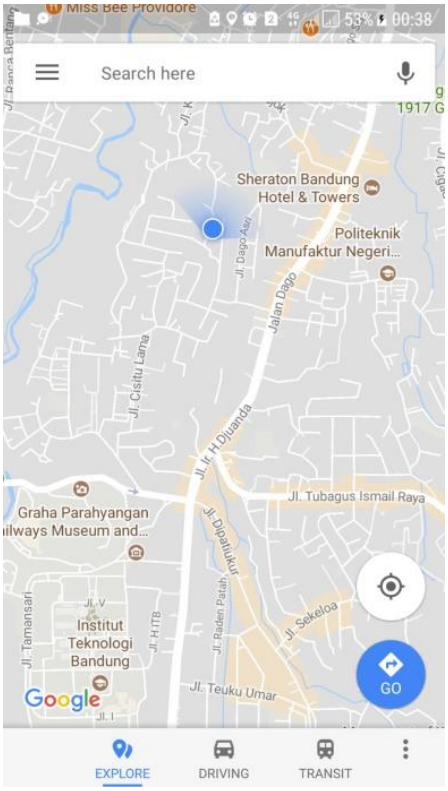
```
Procedure Dijkstra (w,a,z,L)

  L(a) := 0
  for semua simpul x ≠ a do
  L(x) := ∞
  T := himpunan semua simpul
  //T merupakan himpunan simpul yang
panjang terpendeknya dari a belum ditemukan

  while z ϵ T do
  begin
      pilih v ϵ T dengan minimum L(v)
      T := T-{v}
      for setiap x ϵ T di samping v do
      L(x) := min{L(x), L(v)+w(v,x)}
  end
end Dijkstra
```

III. PENGGUNAAN APLIKASI GOOGLE MAPS UNTUK PENENTUAN JALUR TECEPAT

Google maps adalah sebuah perangkat lunak dalam internet yang berisi peta atas sebuah wilayah atau lokasi. Google maps merupakan perluasan manfaat dari situs google. Sebelumnya, situs ini hanya dikenal sebagai mesin pencari atau *search engine* di dunia internet saja. namun seiring perkembangan teknologi, google membuat terobosan baru yang berbeda dengan mesin pencari lainnya.



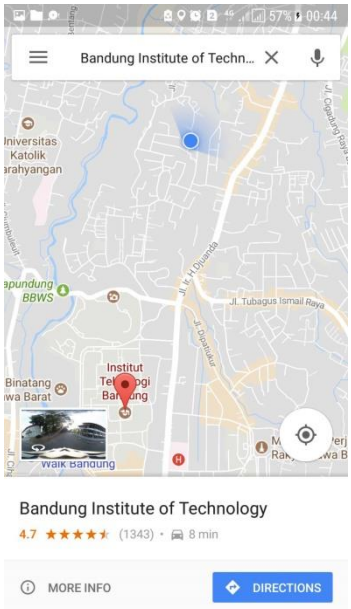
Gambar 1. Tampilan Google Maps Melalui Situs Google Maps



Gambar 2. Tampilan Google Maps Melalui Aplikasi Google Maps pada Gawai

Cara memulai atau menghentikan navigasi pada Google Maps adalah sebagai berikut[6].

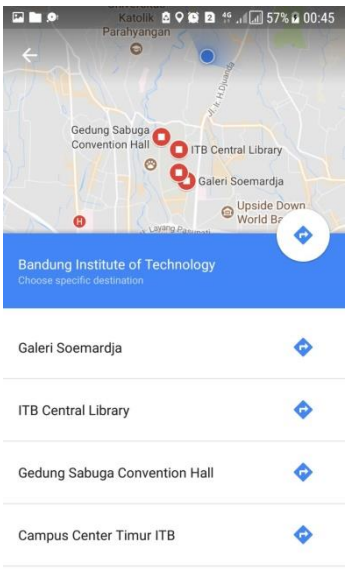
1. Buka aplikasi Google Maps atau buka tautan https://maps.google.com/.
2. Telusuri tempat atau klik tempat tersebut pada peta. Jika pengguna mengetikan sebuah tempat, maka tempat tersebut akan diberikan tanda berwarna merah, sedangkan tanda berwarna biru merupakan lokasi pengguna saat ini.



Gambar 3. Tampilan Google Maps Saat Menunjukkan Lokasi Tujuan

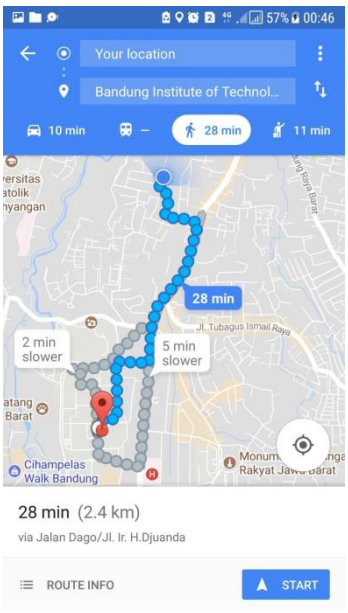
3. Di kanan bawah, klik "Petunjuk Arah" ("*Directions*"). Jika tombol tersebut diklik selama beberapa saat, maka navigasi akan dimulai dan langkah 4-6 dapat dilewati.
4. Opsional: Untuk menambahkan tujuan lainnya, klik tombol "Lainnya ⋮ › Tambahkan perhentian", di kanan atas. Opsi tersebut memungkinkan pengguna untuk menambahkan hingga sembilan perhentian. Jika sudah selesai, klik tombol "Selesai".
5. Pilih salah satu opsi berikut.
   - Mengemudi:
   - Transportasi umum:
   - Jalan kaki:
   - Transportasi online:
   - Bersepeda:

   Selain itu, pengguna pun dapat memilih pintu atau jalur masuk mana yang ingin dilewati.



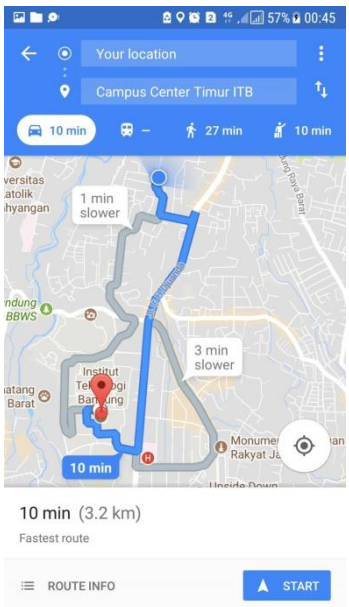
Gambar 4. Tampilan Google Maps Saat Memberikan Opsi Tujuan Lebih Spesifik

6. Jika rute lain tersedia, rute tersebut akan ditampilkan dengan warna abu-abu pada peta. Untuk mengikuti rute alternatif, klik pada garis abu-abu tersebut.

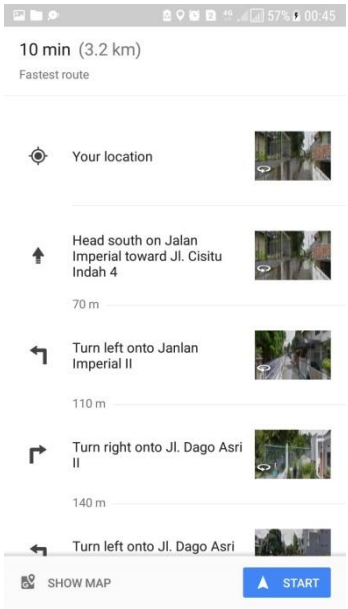


Gambar 5. Tampilan Google Maps Sebelum Navigasi Dimulai Jika Ingin Pergi dengan Berjalan Kaki

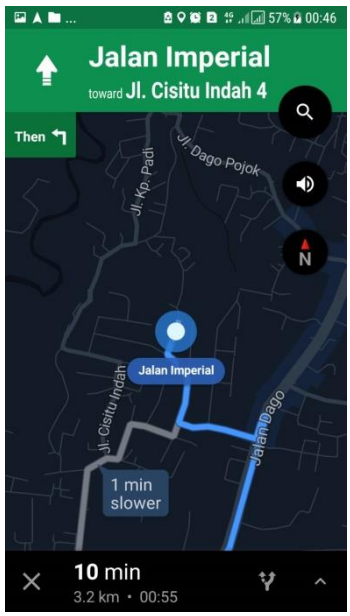
7. Untuk memulai navigasi, klik "Mulai". Jika pengguna melihat tulisan "Mencari GPS" artinya ponsel pengguna tersebut sedang mencoba mendapatkan sinyal GPS.



Gambar 6. Tampilan Google Maps dari Jalan Imperial Menuju ITB (CC Timur) Sebelum Navigasi Dimulai Jika Ingin Pergi dengan Mengemudi



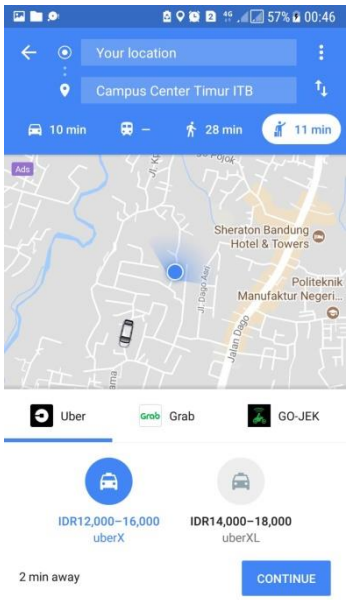
Gambar 7. Tampilan Google Maps untuk Rute Tercepat



Gambar 8. Tampilan Google Maps Setelah Navigasi Dimulai

8. Untuk menghentikan atau membatalkan navigasi, klik tombol “Keluar” di kanan bawah.

Sebagai informasi tambahan, aplikasi Google Maps juga memungkinkan pengguna untuk menggunakan transportasi *online* yang tersedia serta akan memberikan informasi tentang estimasi harga yang harus dibayarkan.



Gambar 9. Tampilan Google Maps untuk Menggunakan Transportasi *Online*

IV. PENERAPAN ALGORITMA DIJKSTRA DALAM MENENTUKAN JALUR TERCEPAT PADA GOOGLE MAPS

Seperti yang telah dibahas pada Bab II, algoritma Dijkstra merupakan suatu algoritma rakus (*greedy*) yang digunakan

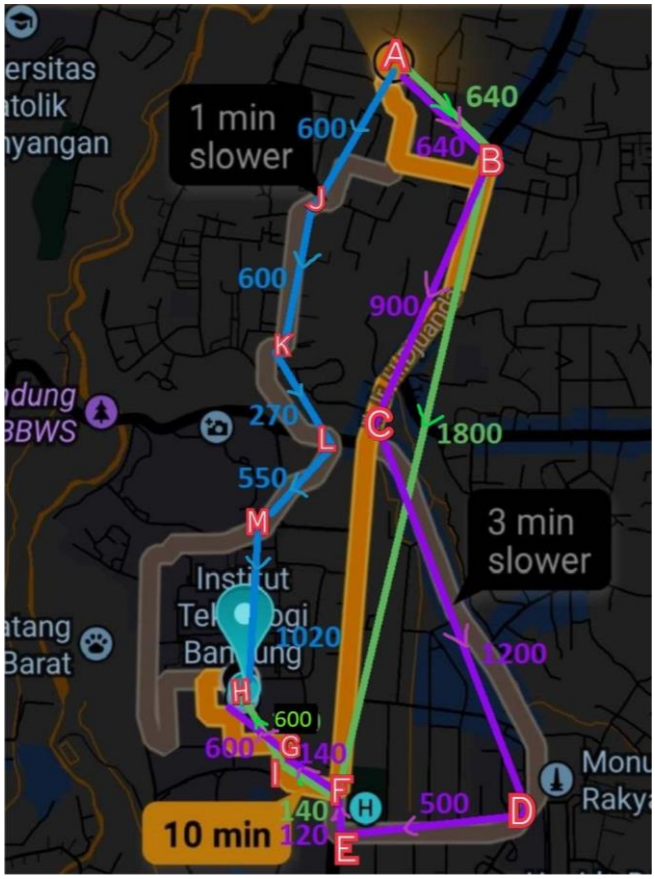
untuk mencari rute permasalahan terpendek antara simpul sumber dan simpul tujuan untuk suatu graf berarah berdasarkan bobot atau jarak pada sisi yang bernilai tak-negatif[9].

Pada algoritma Dijkstra, total biaya untuk mencapai suatu simpul adalah

$$f(n_i) = g(n) + c(n, n_i)$$ [9]

Pada Gambar 1, terdapat sebuah rute dari Jalan Imperial, Dago Asri, Bandung menuju Institut Teknologi Bandung. Pada rute tersebut terdapat tiga jalur alternatif, yaitu jalur berwarna biru, ungu, dan hijau. Menurut aplikasi Google Maps yang penulis miliki, jalur tercepat menuju Institut Teknologi Bandung adalah jalur berwarna hijau, yakni 10 menit. Sedangkan, jalur berwarna biru menghabiskan waktu selama 11 menit, dan jalur berwarna ungu menghabiskan waktu selama 13 menit. Informasi bobot (jarak) pada setiap sisi yang dituliskan pada Gambar 1 diperoleh dari aplikasi Google Maps apabila menekan tombol "Informasi Rute" ("*Route Info*") pada bagian bawah di samping tombol "Mulai" ("*Start*").

Untuk mempemudah perhitungan, setiap angka pada peta yang merupakan jarak dalam meter diganti menjadi jarak dalam hektometer ($1\ hm = 10^2\ m$) agar seluruh angka terdiri atas maksimal dua digit dan penulisan satuan tidak dicantumkan dalam tabel. Selain itu, untuk memperjelas informasi pada Gambar 1, panah pada simpul A hanya mengarah ke simpul-simpul lain, dan panah pada simpul H hanya mengarah ke dirinya sendiri.



Gambar 1. Peta Rute Jalan Imperial Menuju ITB pada Google Maps dengan Warna yang Dibalikkan (*Invert Color*)

Selanjutnya, penulis membuat tabel iterasi untuk mengerjakan algoritma Dijkstra sesuai dengan langkah-langkah yang telah disebutkan pada Bab II dengan sedikit perbedaan cara yang akan dijelaskan setelah ini. Cara yang penulis gunakan ini diperoleh dari sumber yang tertera pada Referensi butir [2] dan [7].

A merupakan Simpul Keberangkatan dan H merupakan Simpul Tujuan. Simpul terjamah diberi warna kuning dan simpul yang akan dijamah selanjutnya diberi warna yang sama serta diberi tanda bintang (*) untuk memudahkan penulis dalam mengerjakan. Simpul-simpul yang belum terjamah akan dijamah setelah Simpul Tujuan tercapai, yakni simpul H. Sel tabel berisi jarak yang ditempuh hingga menuju simpul yang tertera pada kolom sel yang bersangkutan melalui simpul yang tertera setelah penulisan jarak. Misalkan pada sel tabel (A, B) tertera "6,4 A", maka maksudnya adalah jarak yang ditempuh hingga simpul B (karena berada pada kolom B) adalah 6,4 hm melalui simpul A. Informasi isi tabel untuk simpul yang sudah dijamah dan belum dijamah tetap dituliskan kembali pada baris berikutnya hingga ada simpul lain yang memiliki jarak (bobot) lebih sedikit atau lebih baik lalu akan menggantikannya. Misalkan "∞" pada sel (J, F) digantikan dengan "24,4 B" pada sel (B, F) karena 24,4 lebih baik dibandingkan tak-hingga.

Mulai ↓

A →	B	C	D	E	F	G	H	I	J	K	L	M
A	6,4 A	∞	∞	∞	∞	∞	∞	∞	*6 A	∞	∞	∞
J	*6,4 A	∞	∞	∞	∞	∞	∞	∞	6 A	12 J	∞	∞
B	6,4 A	15,4 B	∞	∞	24,4 B	∞	∞	∞	6 A	*12 J	∞	∞
K	6,4 A	15,4 B	∞	∞	24,4 B	∞	∞	∞	6 A	12 J	*14,7 K	∞
L	6,4 A	*15,4 B	∞	∞	24,4 B	∞	∞	∞	6 A	12 J	14,7 K	20,2 L
C	6,4 A	15,4 B	27,4 C	∞	24,4 B	∞	∞	∞	6 A	12 J	14,7 K	*20,2 L
M	6,4 A	15,4 B	27,4 C	∞	*24,4 B	∞	30,4 M	∞	6 A	12 J	14,7 K	20,2 L
F	6,4 A	15,4 B	27,4 C	∞	24,4 B	35 F	30,4 M	*25,8 F	6 A	12 J	14,7 K	20,2 L
I	6,4 A	15,4 B	*27,4 C	∞	24,4 B	35 F	Min(30,4 M;31,8 I) = 30,4 M	25,8 F	6 A	12 J	14,7 K	20,2 L
D	6,4 A	15,4 B	27,4 C	32,4 D	24,4 B	35 F	*30,4 M	*25,8 F	6 A	12 J	14,7 K	20,2 L
H	6,4 A	15,4 B	27,4 C	*32,4 D	24,4 B	35 F	30,4 M	25,8 F	6 A	12 J	14,7 K	20,2 L
E	6,4 A	15,4 B	27,4 C	32,4 D	min(24,4 B;33,6 E) = 24,4 B	*35 F	30,4 M	25,8 F	6 A	12 J	14,7 K	20,2 L
G	6,4 A	15,4 B	27,4 C	32,4 D	24,4 B	35 F	min(30,4 M; 41 G) = 30,4 M	25,8 F	6 A	12 J	14,7 K	20,2 L

Gambar 2. Tabel Iterasi

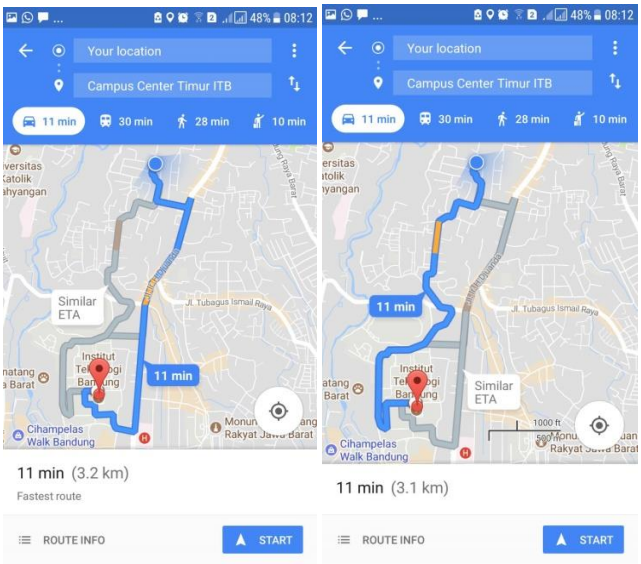
Pada awalnya, simpul keberangkatan A akan menuju simpul J karena memiliki jarak terdekat (6 < 6,4). Selanjutnya, dari simpul J akan menuju simpul K (jika menyesuaikan dengan jalur pada peta). Namun, simpul A menuju B memiliki jarak lebih sedikit (pada tabel dituliskan 6,4 A di sel (J,B)), sehingga simpul yang akan dijamah selanjutnya adalah simpul B. Dari simpul B, diperoleh jalur menuju simpul C dan F, namun sudah terisi juga kolom A, J, dan K pada pengisian sebelumnya. Karena A dan J sudah dijamah sebelumnya (sudah berwarna kuning), maka pilihan simpul yang akan dijamah selanjutnya hanya C, F atau K, kemudian dipilih simpul K karena jarak yang dihasilkan paling sedikit (12 < 15,4 dan 12 < 24,4).

Setelah itu, isi tabel (B, K) diberi tanda bintang ("*") dan diwarnai kuning yang artinya K akan dituliskan dan ditinjau pada baris setelah B. Selanjutnya, tinjau simpul K dan akan dilanjutkan ke simpul L. Dari simpul L akan dilanjutkan ke simpul C, M, F, I, D, dan H dengan cara yang sama.

Untuk melihat jalur mana yang memiliki jarak terdekat sampai tujuan, kita lihat hasil yang diperoleh di sel tabel (H, H). Pada bagian tersebut tertera "30,4 M" yang artinya jarak dari simpul A menuju simpul H adalah sebesar 30,4 hm melalui simpul M. Kemudian, kita tinjau kembali asal mula M. Agar dapat mencapai simpul H dari simpul A melalui simpul M, rute yang dilalui adalah: A-J-K-L-M-H. Oleh karena itu, rute A-J-K-L-M-H atau lintasan biru pada peta merupakan jalur terdekat menurut algoritma Dijkstra.

Untuk simpul-simpul yang belum terjamah, yaitu E dan G akan dijamah pada baris setelah H. Setelah dijamah, diperoleh jarak sebesar 41 hm pada kolom H melalui simpul G (tertulis "41 G"), atau melalui rute A-B-C-D-E-F-G-H. Namun, karena jarak tersebut tidak lebih sedikit daripada jarak sebelumnya, yakni 30,4 hm melalui simpul M, maka hasil terkecil yang merupakan jarak terdekat pada kolom H tetap 30,4 hm melalui M.

Pada Gambar 1, kita dapat melihat bahwa Google Maps menyarankan pengguna untuk melalui jalur berwarna hijau (yang aslinya merupakan jalur oranye), yakni rute A-B-F-I-H dibandingkan jalur berwarna biru (yang secara perhitungan algoritma Dijkstra memiliki jarak terdekat). Perbedaan waktu yang tercatat adalah "*1 min slower*" alias satu menit lebih lambat dan perbedaan jarak yang ditempuh adalah 31,8 hm - 30,4 hm = 1,4 hm atau setara dengan 140 meter. Hal tersebut dapat terjadi karena banyak kemungkinan, salah satunya adalah karena jalur berwarna hijau memiliki lebih sedikit belokan atau gangguan di jalan sehingga memungkinkan perjalanan akan lebih lancar. Google Maps dapat mendeteksi kecepatan dari suatu kendaraan pada lokasi tertentu dengan melacak lokasi gawai yang dibawa oleh sang pengemudi (seperti telah dijelaskan pada Bab I).



Gambar 3. Peta Rute Jalan Imperial Menuju ITB pada Google Maps di Pagi Hari

Sistem navigasi yang penulis gunakan untuk contoh di atas diakses pada malam hari sekitar pukul 01.00. Setelah pagi hari, penulis kembali mengecek Google Maps untuk melakukan navigasi dari lokasi awal menuju lokasi tujuan yang sama, hasil yang diperoleh dapat dilihat pada Gambar 3.

Pada Gambar 3, kita dapat melihat bahwa Google Maps hanya menyarankan dua jalur tercepat sesuai dengan hasil kalkulasi menggunakan algoritma Dijkstra (kanan) dan alternatif tercepat menurut Google Maps saat percobaan sebelumnya (kiri). Kini, terdapat tulisan "*Similar ETA*" pada kedua jalur tersebut, dengan ETA merupakan singkatan dari *Estimated Time of Arrival* atau dapat diartikan sebagai estimasi waktu kedatangan yang serupa. Artinya, hasil kalkulasi menggunakan algoritma Dijkstra dapat diterapkan untuk menentukan jalur tercepat pada Google Maps karena memperoleh hasil yang sesuai.

V. KESIMPULAN

Google Maps memiliki banyak algoritma dan berbagai pertimbangan lain dalam menentukan jalur tercepat dari suatu lokasi awal menuju lokasi tujuan. Algoritma Dijkstra dapat menjadi salah satu alternatif untuk menentukan jalur tercepat dari berbagai alternatif jalur yang tersedia karena algoritma Dijkstra dapat menunjukkan jalur dengan jarak terdekat dan berpotensi memiliki waktu tercepat untuk sampai ke tujuan.

VI. PENUTUP

Pertama-tama, penulis mengucapkan puji syukur kepada Allah SWT. yang telah memberikan kesempatan serta kelancaran untuk menyelesaikan makalah ini. Penulis juga mengucapkan banyak terima kasih kepada orangtua penulis karena telah memberikan bantuan moral, material, doa serta telah menjadi salah satu sumber motivasi penulis. Selain itu, penulis juga mengucapkan banyak terima kasih kepada Bapak Judhi Santoso, selaku dosen dari Mata Kuliah Matematika Diskrit Kelas 03, serta kepada Bapak Rinaldi Munir dan Ibu Harili yang juga merupakan dosen dari Mata Kuliah Matematika Diskrit dan telah bersama-sama mengampu mahasiswa/i agar dapat memahami mata kuliah ini. Tak lupa, penulis pun mengucapkan banyak terima kasih kepada seluruh sumber informasi penulis dalam pembuatan makalah ini yang telah dicantumkan pada Referensi.

REFERENSI

[1] Rinaldi Munir, Matematika Diskrit, edisi ketiga. Bandung: Penerbit Informatika Bandung, 2010.

[2] Graphs: Dijkstra's Algorithm. Diakses dari https://www.youtube.com/watch?v=8Ls1RqHCOPw pada 1 Desember 2017 pukul 10.00.

[3] Metode Pencarian Jalur Terpendek (Dijkstra Algorithm) – Sutikno. Diakses dari: http://sutikno.blog.undip.ac.id/files/2011/10/dijkstra-pencarian-jalur-terpendek.pdf pada 1 Desember 2017 pukul 11.00.

[4] How Google Maps Knows When There's Traffic. Diakses dari: https://www.youtube.com/watch?v=nLasxr0GsoQ pada 1 Desember 2017 pukul 11.00.

[5] KBBI Daring: Pencarian. Diakses dari: https://kbbi.kemdikbud.go.id pada 1 Desember 2017 pukul 13.00.
[6] Menggunakan Navigasi di Aplikasi Google Maps. Diakses dari: https://support.google.com/maps/answer/3273406?co=GENIE.Platform%3DAndroid&hl=id pada 1 Desember 2017 pukul 16.00.
[7] Implementasi Algoritma Dijkstra. Diakses dari: https://www.youtube.com/watch?v=3YUMgvv8YUs&t=25s pada 1 Desember 2017 pukul 16.00.
[8] DIJKSTRA'S ALGORITHM. Diakses dari: http://math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Melissa.pdf pada 1 Desember 2017 pukul 16.00.
[9] Library Binus Page 39 of 40. Diakses dari: http://library.binus.ac.id/eColls/eThesisdoc/Bab2HTML/2009200226IFBab2/page39.html pada 2 Desember 2017 pukul 12.45.

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, 3 Desember 2017

Nira Rizki Ramadhani 13516018