APLIKASI ALGORITMA DJIKSTRA DALAM MENENTUKAN RUTE PERJALANAN TERCEPAT

Inka Anindya Riyadi / 13518038
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13518038@std.stei.itb.ac.id

***Abstract*—Persoalan yang sering timbul di masyarakat dengan penduduk yang banyak adalah kemacetan. Permasalahan ini bisa diselesaikan dengan salah satu metode yaitu dengan menggunakan graf. Peta dapat kita buat menjadi sebuah graf terhubung. Dengan menggunakan algoritma Djikstra, kita bisa menghitung rute tercepat dari suatu titik keberangkatan ke titik yang lainnya.**

***Keywords*—Macet, Graf, Algoritma Djikstra, Rute.**

I. PENDAHULUAN

Pada abad ke-21, banyak perkembangan zaman yang terlaksana, salah satunya teknologi. Perkembangan teknologi seiring dengan jalannya perkembangan industry yang diawali pada tahun 1750. Perkembangan industry hingga saat ini sudah memasuki revolusi industry 4.0. Teknologi adalah alat-alat yang diciptakan manusia untuk mempermudah kehidupan dengan ilmu yang dikembangkan secara terus menerus. Teknologi juga timbul karena adanya suatu keresahan pada manusia ketika melakukan sesuatu. Salah satu permasalahan yang ada pada masyarakat dalam melakukan kegiatan sehari-hari adalah efisiensi dalam bertransportasi. Kemacetan terjadi terutama pada kota-kota besar yang banyak penduduknya, yang merupakan pusat bisnis, sehingga banyak kendaraan yang berlalu Lalang dalam wilayah tersebut. Kemacetan tentunya menimbulkan efek negatif, salah satunya adalah menurunkan produktivitas pribadi yang juga dapat berpengaruh kepada produktivitas dalam lingkup yang lebih besar contohnya lingkup perusahaan, atau lingkup negara.

Gambar 1 : Kemacetan
Sumber:https://www.kompasiana.com/edwardiyonoekoprasojo/5d111106320f365f25427c63/problematika-kemacetan-jakarta

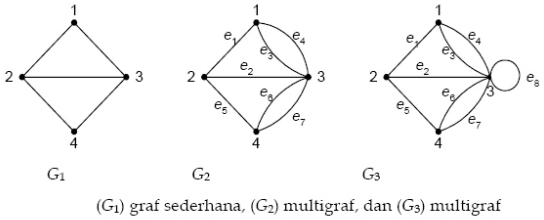
Salah satu solusi dalam permasalahan ini dapat diselesaikan dengan teknologi. Teknologi tersebut dapat berupa peta digital yang dapat menghitung estimasi waktu yang paling sedikit dari suatu perjalanan di suatu titik ke titik yang lainnya. Dengan adanya teknologi ini, diharapkan pengguna transportasi dapat menempuh suatu destinasi dengan waktu seefisien mungkin, menghindari kemacetan, ataupun hal-hal lain yang dapat membuat waktu tempuh menjadi lebih lama. Penyelesaian dari permasalahan ini dapat diselesaikan dengan menggunakan materi Graf dimana jalan dapat dibentuk menjadi graf dan berdasarkan teori-teori dan algoritma yang ada dalam graf, kita bisa menyelesaikan permasalahan ini.

II. LANDASAN TEORI

A. Graf

Graf adalah graf adalah himpunan dari objek-objek yang dinamakan titik, simpul, atau sudut dihubungkan oleh penghubung yang dinamakan garis atau sisi.

Secara matematis, definisi graf adalah sebagai berikut:
Graf G = (V, E), yang dalam hal ini:
V = himpunan tidak-kosong dari simpul-simpul (vertices) = { v1 , v2 , ... , vn }
E = himpunan sisi (edges) yang menghubungkan sepasang simpul = {e1 , e2 , ... , en }

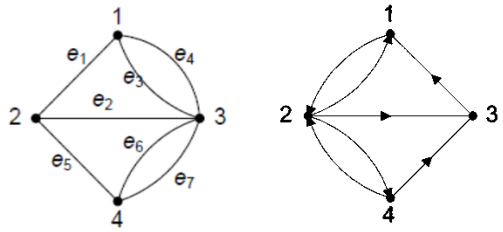


(G_1) graf sederhana, (G_2) multigraf, dan (G_3) multigraf

Gambar 2 : Contoh graf
Sumber:http://athayaniimtinan.blogspot.com/2017/12/pewarnaan-graf.html

Berdasarkan ada tidaknya gelang atau s

isi ganda pada suatu graf, maka graf digolongkan menjadi dua jenis: graf sederhana dan graf tak-sederhana. Berdasarkan orientasi arah pada sisi, maka secara umum graf dibedakan atas 2 jenis: graf berarah dan graf tidak berarah

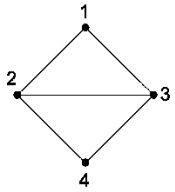


Gambar 3 : Jenis graf berdasarkan arah
Sumber:http://rabbitjeyek.blogspot.com/2011/12/teori-graf-4.html

Jenis	**Sisi**	**Sisi Ganda**	**Sisi Gelang**
Graf Sederhana	Tak-berarah	Tidak	Tidak
Graf Ganda	Tak-berarah	Ya	Tidak
Graf Semu	Tak-berarah	Ya	Ya
Graf Berarah	Berarah	Tidak	Ya
Graf-ganda Berarah	Berarah	Ya	Ya

B. Terminologi Graf

1. Ketetanggan (*Adjacent*)
   Dua buah simpul dikatakan bertetangga bila keduanya terhubung langsung.

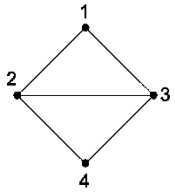


G_1

Gambar 4: Contoh graf
Sumber : Discrete Mathematics and Its Application

Pada graf G_1, simpul 1 bertetangga dengan simpul 2 dan 3, namun simpul 1 tidak bertetangga dengan simpul 4.

2. Bersisian (*Incidency)*
   Untuk sembarang sisi e = (vj, vk) dikatakan e bersisian dengan simpul vj , atau e bersisian dengan simpul vk



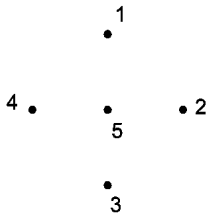
G_1

Gambar 5: Contoh graf
Sumber : Discrete Mathematics and Its Application

Pada graf G_1, sisi (2, 3) bersisian dengan simpul 2 dan simpul 3, sisi (2, 4) bersisian dengan simpul 2 dan simpul 4, tetapi sisi (1, 2) tidak bersisian dengan simpul 4

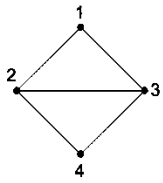
3. Simpul terpencil (*Isolated Vertex*)
   Simpul terpencil ialah simpul yang tidak mempunyai sisi yang bersisian dengannya.

4. Graf kosong (*Null Graph)*
   Graf yang himpunan sisinya merupakan himpunan kosong



Gambar 6: Contoh graf
Sumber : Discrete Mathematics and Its Application

5. Derajat (*Degree*)
   Derajat suatu simpul adalah jumlah sisi yang bersisian dengan simpul tersebut.
   Notasi: d(v)



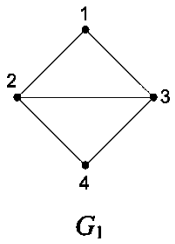
G_1

Gambar 7: Contoh graf
Sumber : Discrete Mathematics and Its Application

Pada graf G_1, jumlah derajat pada simpul 1 sama dengan simpul 4, yaitu d(1) = d(4) = 2 ,jumlah derajat pada simpul 2 sama dengan simpul 3 yaitu yaitu d(2) = d(3) = 3

6. Lintasan *(Path*)
   Lintasan yang panjangnya n dari simpul awal v0 ke simpul tujuan vn di dalam graf G ialah barisan

berselang-seling simpul-simpul dan sisi-sisi yang berbentuk v0, e1, v1, e2, v2,... , vn –1, en, vn sedemikian sehingga e1 = (v0, v1), e2 = (v1, v2), ... , en = (vn-1, vn) adalah sisi-sisi dari graf G.



G_1

Gambar 8: Contoh graf
Sumber : Discrete Mathematics and Its Application

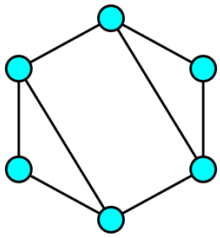
Pada graf G_1, lintasan 1, 2, 4, 3 adalah lintasan dengan barisan sisi (1,2), (2,4), (4,3).

Panjang lintasan adalah jumlah sisi dalam lintasan tersebut.
Lintasan 1, 2, 4, 3 pada G1 memiliki panjang 3.

7. Sirkuit (*Circuit*)
   Lintasan yang berawal dan berakhir pada simpul yang sama disebut sirkuit atau siklus. Panjang sirkuit adalah jumlah sisi dalam sirkuit tersebut

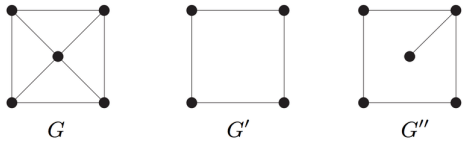
8. Terhubung (*Connected*)
   Dua buah simpul v1 dan simpul v2 disebut terhubung jika terdapat lintasan dari v1 ke v2. G disebut graf terhubung (connected graph) jika untuk setiap pasang simpul vi dan vj dalam himpunan V terdapat lintasan dari vi ke vj. Jika tidak, maka G disebut graf tak-terhubung (disconnected graph).

   Graf berarah G dikatakan terhubung jika graf tidak berarahnya terhubung (graf tidak berarah dari G diperoleh dengan menghilangkan arahnya). Dua simpul, u dan v, pada graf berarah G disebut terhubung kuat (strongly connected) jika terdapat lintasan berarah dari u ke v dan juga lintasan berarah dari v ke u. Jika u dan v tidak terhubung kuat tetapi terhubung pada graf tidak berarahnya, maka u dan v dikatakan terhubung lemah (weakly coonected). Graf berarah G disebut graf terhubung kuat (strongly connected graph) apabila untuk setiap pasang simpul sembarang u dan v di G, terhubung kuat. Kalau tidak, G disebut graf terhubung lemah.



Gambar 9 : Contoh graf terhubung
Sumber: https://www.wikiwand.com/en/K-edge-connected_graph

9. Upagraf (*Subgraph*)
   Misalkan G = (V, E) adalah sebuah graf. G1 = (V1, E1) adalah upagraf (subgraph) dari G jika V1 □ V dan E1 □ E. Komplemen dari upagraf G1 terhadap graf G adalah graf G2 = (V2, E2) sedemikian sehingga E2 = E - E1 dan V2 adalah himpunan simpul yang anggota-anggota E2 bersisian dengannya. Komponen graf (connected component) adalah jumlah maksimum upagraf terhubung dalam graf G. Pada graf berarah, komponen terhubung kuat (strongly connected component) adalah jumlah maksimum upagraf yang terhubung kuat.



Gambar 10 : Graf dan Upagraf
Sumber:https://www.researchgate.net/figure/A-graph-G-with-subgraphs-G-and-G-G-is-an-induces-subgraph-of-G-but-G-is-not-On-the_fig3_275027315

10. Upagraf rentang (*Spanning Subgraph)*
    Upagraf G1 = (V1, E1) dari G = (V, E) dikatakan upagraf rentang jika V1 =V (yaitu G1 mengandung semua simpul dari G).

11. *Cut-set*
    Cut-set dari graf terhubung G adalah himpunan sisi yang bila dibuang dari G menyebabkan G tidak terhubung. Jadi, cut-set selalu menghasilkan dua buah komponen.

12. Graf berbobot (*Weighted Graph)*
    Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga (bobot).

C. Algoritma Djikstra
   Algoritma Djikstra dapat digunakan untuk mencari lintasan dengan bobot minimum dari suatu simpul ke simpul yang lainnya. Algoritma ini bekerja dengan membuat jalur ke satu simpul optimal pada setiap langkah. Jika pada langkah ke n, setidaknya ada n simpul yang sudah kita tahu jalur terpendek.

   Langkah-langkah algoritma Dijkstra dapat dilakukan dengan langkah-langkah berikut:

   1. Tentukan titik mana yang akan menjadi node awal, lalu beri bobot jarak pada node pertama ke node terdekat satu per satu, Dijkstra akan melakukan pengembangan pencarian dari satu

titik ke titik lain dan ke titik selanjutnya tahap demi tahap.

2. 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.
3. Set semua node yang belum dilalui dan set node awal sebagai "Node keberangkatan"
4. Dari node keberangkatan, pertimbangkan node tetangga yang belum dilalui dan hitung jaraknya dari titik keberangkatan. Jika jarak ini lebih kecil dari jarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru
5. Saat kita selesai mempertimbangkan setiap jarak terhadap node tetangga, tandai node yang telah dilalui sebagai "Node dilewati". Node yang dilewati tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.
6. Set "Node belum dilewati" dengan jarak terkecil (dari node keberangkatan) sebagai "Node Keberangkatan" selanjutnya dan ulangi langkah 5.

Iteration	Unvisited (Q)	Visited (S)	Current	Node : Min = (dist[node], prev[node])iteration						
				V1	V2	V3	V4	V5	V6	V7
	Initialization (V1,V2,V3,V4,V5,V6,V7)	(-)		(0, -)0	(∞, -)0	(∞, -)0	(∞, -)0	(∞, -)0	(∞, -)0	(∞, -)0
1	(V2,V3,V4,V5,V6,V7)	(V1)	V1		(5,V1)1	(7,V1)1	(12,V1)1	(∞,V1)1	(∞,V1)1	(∞,V1)1
2	(V3,V4,V5,V6,V7)	(V1,V2)	V2			(6,V2)2	(12,V1)1	(11,V2)2	(∞,V2)2	(∞,V2)2
3	(V4,V5,V6,V7)	(V1,V2, V3)	V3				(7,V3)3	(11,V3)3	(16,V3)3	(∞,V3)3
4	(V5,V6,V7)	(V1,V2, V3,V4)	V4					(11,V3)3	(16,V3)3	(∞,V3)3
5	(V6,V7)	(V1,V2, V3,V4,V5)	V5						(13,V5)5	(18,V5)5
6	(V7)	(V1,V2, V3,V4,V5,V6)	V6							(16,V6)6

Gambar 11 : Contoh Penyelesaian Algoritma Djikstra Menggunakan Tabel

Sumber:https://mti.binus.ac.id/2017/11/28/algoritma-dijkstra/

IV. APLIKASI ALGORITMA DJIKSTRA DALAM MENENTUKAN RUTE PERJALANAN TERCEPAT

Permasalahan yang sering ditemukan dalam kehidupan sehari-hari yaitu kemacetan yang menyebabkan termakannya waktu dijalan yang dapat menurunkan produktivitas. Solusi dari permasalahan ini adalah dengan dibuatnya suatu peta digital dan dengan peta tersebut dapat menentukan rute perjalanan mana yang tercepat ke suatu tujuan.

Peta ini bisa kita gambar dalam bentuk graf terhubung, dimana setiap adanya persimpangan diperlakukan seperti simpul pada graf. Rute perjalanan merupakan suatu lintasan graf, namun bukan merupakan sirkuit. Apabila rute merupakan suatu sirkuit, maka estimasi waktu perjalanan bernilai 0.

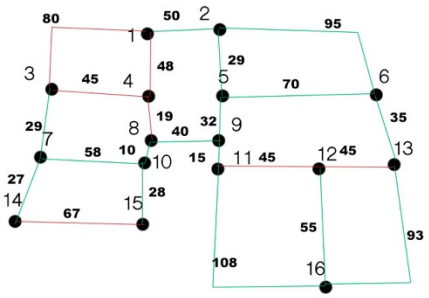
Untuk mencari rute tercepat, maka kita harus mencari lintasan dengan bobot seminimum mungkin, dimana bobot itu dalam satuan waktu. Bobot tiap simpul ini merupakan hasil perkalian dari jarak dan tingkat kemacetan diantara 2 simpul, sehingga satuan dari bobot merupakan waktu.

Untuk menyelesaikan permasalahan ini, kita dapat menggunakan algoritma Djikstra untuk mencari rute tercepat. Sebuah rute perjalanan merupakan suatu lintasan dalam graf. Dalam kasus ini, asumsi rute adalah 2 arah sehingga graf merupakan bentuk tidak berarah.

Berikut merupakan peta dari salah satu wilayah di Bandung.

Gambar 12 : Salah satu peta wilayah di Kota Bandung
Sumber: https://www.google.com/maps/@-6.8899551,107.6252504,18z

Dengan peta tersebut, kita dapat membuat graf tidak berarahnya, yaitu seperti berikut:

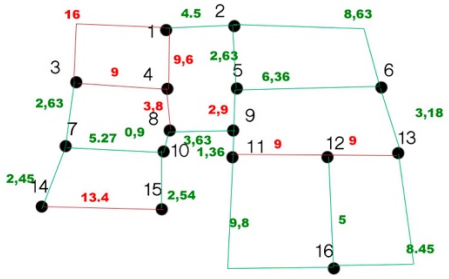


Gambar 13 : Graf dengan bobot jarak

Dalam graf ini, bobot merupakan jarak dari suatu simpul ke simpul yang lainnya dengan satuan meter (m). Setiap sisi mempunyai warna, baik hijau atau merah. Ini merupakan simulasi kemacetan dalam daerah tersebut disuatu waktu. Jalan lancar apabila sisi mempunyai graf berwarna hijau dan jalan macet apabila sisi mempunyai graf berwarna merah. Bobot antara simpul 1 dan 3 adalah 80 m. Bobot antara simpul 1 dan 2 adalah 50 m. Bobot antara simpul 2 dan simpul 6 adalah 95 m. Bobot antara simpul 2 dan simpul 5 adalah 29 m. Bobot antara simpul 5 dan 6 adalah 70 m. Bobot antara simpul 5 dan 9 adalah 32 m. Bobot antara simpul 9 dan 11 adalah 15 m. Bobot antara simpul 11 dan 12 adalah 45 m. Bobot antara simpul 11 dan 16 adalah 108 m. Bobot antara simpul 12 dan 16 adalah 45 m. Bobot antara simpul 16 dan 13 adalah 93 m. Bobot antara simpul 12 dan 13 adalah 45 m. Bobot antara simpul 13 dan 6 adalah 35 m. Bobot antara simpul 9 dan 8 adalah 40 m. Bobot antara simpul 8 dan simpul 4 adalah 19 m. Bobot antara simpul 4 dan 1 adalah 48 m. Bobot antara simpul 4 dan 3 adalah 45 m. Bobo tantara simpul 8 dan 10 adalah 10 m. Bobot antara simpul 10 dan 15 adalah 28 m. Bobot antara simpul 15 dan 14 adalah 67 m. Bobot antara simpul 14 dan 7 adalah 27 m. Bobot antara simpul 7 dan 10 adalah 58 m. Bobot antara simpul 3 dan 7 adalah 29 m.

Berdasarkan hasil Survei Primer Bandung Road Safety Annual Report 2017, kecepatan rata-rata lalu lintas di Kota Bandung 40 km/jam. Sedangkan, untuk kecepatan rata-rata kendaraan pada salah satu wilayah yang ramai di ruas Cibiru, Soekarno Hatta, Ibrahim Adjie, Gatot Subroto, dan Asia Afrika sebesar 20 km/jam. Berdasarkan data ini, saya mengklasifikasi apabila jalan tersebut tidak ada kemacetan maka kecepatannya saya ubah menjadi m/s yaitu 11.11 m/s dan saya bulatkan menjadi 11 m/s. Apabila terjadi kemacetan saya ubah menjadi m/s yaitu 5.5 dan saya bulatkan menjadi 5 m/s.

Berdasarkan data tersebut, dapat dihitung bobot baru dengan satuan waktu yang merupakan hasil bagi dari jarak/kecepatannya. Berikut merupakan hasil waktu yang diperoleh masing-masing jalannya.



Gambar 14 : Graf dengan bobot waktu

9,6 detik. Waktu yang ditempuh dari simpul 4 ke simpul 3 adalah 9 detik. Waktu yang ditempuh dari simpul 8 ke simpul 10 adalah 0,9 detik. Waktu yang ditempuh dari simpul 10 ke simpul 15 adalah 2,54 detik. Waktu yang ditempuh dari simpul 15 ke simpul 14 adalah 13,4 detik. Waktu yang ditempuh dari simpul 14 ke simpul 7 adalah 2,45 detik. Waktu yang ditempuh dari simpul 7 ke simpul 10 adalah 5,27 detik. Waktu yang ditempuh dari simpul 7 ke simpul 3 adalah 2,63 detik.

Setelah adanya perhitungan waktu antar masing-masing simpul, kita dapat menghitung lintasan suatu tempat menggunakan algoritma Djikstra. Dengan menggunakan algoritma ini, kita harus menentukan suatu simpul sebagai titik keberangkatan. Dalam konteks ini saya memilih simpul 11 sebagai titik keberangkatan. Misalkan, Budi ingin menuju suatu tempat kafe yang berada di Kawasan yang berada di peta tersebut, katakanlah kafe berada dalam persimpangan jalan. Namun, karena budi ingin belajar di kafe dengan efektif, ia tidak ingin membuang-buang waktu di jalan sehingga ia butuh bantuan lewat jalur manakah ia bisa mencapai kafe dengan durasi tercepat. Permasalahan ini bisa kita selesaikan menggunakan algoritma Djikstra.

Berikut merupakan hasil dari algoritma Djikstra dalam bentuk tabel. Masing-masing simpul mempunyai 1 kolom yang terdiri dari Time dan Node. Time merupakan bobot grf

ALGORITMA DJIKSTRA

No	1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16	
	Time	Node	Time	Node	Time	Node	Time	Node	Time	Node	Time	Node	Time	Node	Time	Node	Time	Node	Time	Node	Time	Node	Time	Node	Time	Node	Time	Node	Time	Node	Time	Node
11	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	1,36	11	∞	∞	0,0	11	9,0	11	∞	∞	∞	∞	∞	∞	9,8	11
9	∞	∞	∞	∞	∞	∞	∞	∞	2,9	9	∞	∞	∞	∞	3,63	9	1,4	11	∞	∞			9,0	11	∞	∞	∞	∞	∞	∞	9,8	11
5	∞	∞	2,63	5	∞	∞	∞	∞	2,9	9	6,36	5	∞	∞	3,63	9			∞	∞			9,0	11	∞	∞	∞	∞	∞	∞	9,8	11
2	4,5	2	2,63	5	∞	∞	∞	∞			6,36	5	∞	∞	3,63	9			∞	∞			9,0	11	∞	∞	∞	∞	∞	∞	9,8	11
8	4,5	2			∞	∞	3,8	8			6,36	5	∞	∞	3,63	9			0,9	8			9,0	11	∞	∞	∞	∞	∞	∞	9,8	11
10	4,5	2			∞	∞	3,8	8			6,4	5	5,27	10					0,9	8			9,0	11	∞	∞	∞	∞	2,54	10	9,8	11
15	4,5	2			∞	∞	3,8	8			6,4	5	5,3	10									9,0	11	∞	∞	13,4	15	2,54	10	9,8	11
4	4,5	2			9,0	4	3,8	8			6,4	5	5,3	10									9,0	11	∞	∞	13,4	15			9,8	11
1	4,5	2			9,0	4					6,4	5	5,3	10									9,0	11	∞	∞	13,4	15			9,8	11
7					2,63	7					6,4	5	5,3	10									9,0	11	∞	∞	2,45	7			9,8	11
14					2,63	7					6,4	5											9,0	11	∞	∞	2,45	7			9,8	11
3					2,63	7					6,4	5											9,0	11	∞	∞					9,8	11
6											6,4	5											9,0	11	3,18	6					9,8	11
13																							9,0	11	3,18	6					8,45	13
16																							9,0	11							8,45	13
12																							9,0	11								

Tabel 1 : Algoritma Djikstra

Waktu yang ditempuh dari simpul 1 ke simpul 3 adalah 16 detik. Waktu yang ditempuh dari simpul 1 ke simpul 2 adalah 4,5 detik. Waktu yang ditempuh dari simpul 2 ke simpul 6 adalah 8,62 detik. Waktu yang ditempuh dari simpul 2 ke simpul 5 adalah 2,63 detik. Waktu yang ditempuh dari simpul 5 ke simpul 6 adalah 6,36 detik. Waktu yang ditempuh dari simpul 5 ke simpul 9 adalah 2,9 detik. Waktu yang ditempuh dari simpul 9 ke simpul 11 adalah 1,36 detik. Waktu yang ditempuh dari simpul 11 ke simpul 12 adalah 9 detik. Waktu yang ditempuh dari simpul 11 ke simpul 16 adalah 9,8 detik. Waktu yang ditempuh dari simpul 12 ke simpul 16 adalah 5 detik. Waktu yang ditempuh dari simpul 12 ke simpul ke 13 adalah 9 detik. Waktu yang ditempuh dari simpul 16 ke simpul 13 adalah 8,45 detik. Waktu yang ditempuh dari simpul 13 ke simpul 6 adalah 3,18 detik. Waktu yang ditempuh dari simpul 9 ke simpul 8 adalah 3,63 detik. Waktu yang ditempuh dari simpul 8 ke simpul 4 adalah 3,8 detik. Waktu yang ditempuh dari simpul 4 ke simpul 1 adalah

dari suatu simpul ke simpul yang lainnya. Node menyimpan asal simpul ketika menghitung waktu dari satu simpul ke simpul yang lainnya. Algoritma dimulai dari simpul ke 11 karena kita telah menentukan simpul 11 sebagai titik mula keberangkatan. Dari simpul tersebut, kita mencari sisi yang bersisian dengan simpul 11 dan memasukkan waktu ke dalam tabel sesuai dengan kolomnya kemudian memasukkan juga asal simpul ke dalam kolom Node yaitu 11. Bagi sisi yang tidak bersisian dengan simpul 11, maka akan dimasukkan nilai tak hingga. Setelah memasukkan nilai-nilai dengan sisi yang bersisian dengan simpul 11, selanjutnya mengecek dari waktu yang sudah dimasukkan, mana waktu yang tercepat pada suatu simpul selain sisi 11. Simpul tersebut adalah simpul 9 yang mempunyai waktu tempuh 1,4 detik.

Berdasarkan algoritma ini, kita bisa menentukan jalan tercepat untuk menuju suatu kafe yang ada di persimpangan

jalan. Contohnya, Budi ingin ke kafe yang ada di persimpangan jalan pada simpul 13. Berdasarkan algoritma Djikstra, bila Budi ingin ke kafe tersebut maka melewati simpul 11→ 9 → 5 → 6 → 13. Melalui lintasan tersebut, Budi dapat menuju kafe dalam waktu 12,8 detik dengan total jarak 142 m. Sedangkan ada rute lain yang lebih dekat yaitu 11 → 12 → 13 dengan total jarak 90 m dan total waktu 18 detik. Rute kedua lebih lama karena adanya kemacetan pada simpul 11 → 12 dan 12 → 13. Maka, menurut data tersebut Budi dapat memilih rute yang lebih cepat untuk menuju ke kafe 13 meskpun menempuh jarak yang lebih jauh.

V. Conclusion

Permasalahan yang sering ada di masyarakat adalah padatnya suatu wilayah terutama di kota-kota besar yang menimbulkan kemacetan. Permasalahan tersebut dapat diselesaikan dengan salah satu metode matematika, yaitu Graf. Peta wilayah bisa kita analogikan sebagai Graf dengan persimpangan sebagai simpulnya. Dengan adanya aplikasi graf dalam menentukan menentukan rute perjalanan tercepat, bisa mencari rute tercepat ke suatu lokasi dari suatu titik keberangkatan. Perhitungan bobot graf bisa dalan satuan waktu, sehingga kita bisa menempuh suatu tujuan dalam waktu seminimum mungkin, meskipun jarak yang ditempuh lebih jauh. Dalam makalah ini, masih jauh dari kata sempurna, salah satunya belum menangani kasus apabila letak dari suatu tujuan tidak berada pada simpul.

Penerapan dari algoritma Djikstra ini bisa diperdalam dalam penerapan sehari-hari yang lainnya. Penggunaan algoritma ini sangat berguna bagi masyarakat yang tinggal di kota-kota besar yang ingin mengefisienkan waktu perjalanan ke suatu tempat.

VII. Acknowledgment

Pertama-tama, penulis mengucapkan terimakasih kepada puji syukur kepada Tuhan Yang Maha Esa atas berkat dan rahmatnya, penulis bisa menyelesaikan tugas makalah ini. Penulis juga mengucapkan terimakasih kepada kedua orangtua penulis serta kakak yang selalu mendoakan akan kesuksesan penulis dan turut mendukung penulis. Penulis juga menguncapkan terimakasih kepada Ibu Harlili selaku dosen mata kuliah Matematika Diskrit, yang selama ini memberikan ilmu Matematika Diskrit yang sangat membantu pengerjaan makalah ini dan Bapak Rinaldi Munir, yang selama satu semester ini selalu membantu dengan adanya *website* yang berisi materi-materi kuliah, latihan-latihan soal untuk kuis dan ujian, dan semua dokumen pembelajaran, soal dan lainnya yang sangat berguna dalam proses pembelajaran Matematika Diskrit.

References

[1] Rosen, K. H. Discrete Mathematics and Its Application. New York: McGraw-Hill, 2012, edisi ketujuh

[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf. Diakses pada 4 Desember 2019.

[3] https://www.kompasiana.com/edwardiyonoekoprasojo/5d111106320f365f25427c63/problematika. Diakses pada 4 Desember 2019.

[4] http://athayaniimtinan.blogspot.com/2017/12/pewarnaan-graf.html. Diakses pada 4 Desember 2019.

[5] http://rabbitjeyek.blogspot.com/2011/12/teori-graf-4.html. Diakses pada 4 Desember 2019.

[6] https://www.researchgate.net/figure/A-graph-G-with-subgraphs-G-and-G-G-is-an-induces-subgraph-of-G-but-G-is-not-On-the_fig3_275027315. Diakses pada 4 Desember 2019.

[7] https://www.wikiwand.com/en/K-edge-connected_graph. Diakses pada 4 Desember 2019.

[8] https://mti.binus.ac.id/2017/11/28/algoritma-dijkstra/. Diakses pada 4 Desember 2019.

[9] https://www.google.com/maps/@-6.8899551,107252504,18z. Diakses pada 4 Desember 2019.

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, 5 Desember 2019

Inka Anindya Riyadi / 13518038