Pengaplikasian Graf Dalam Menentukan Lokasi Huni Strategis

Rafi Raihansyah Munandar (13519154)[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13519154@std.stei.itb.ac.id

***Abstraksi*—Graf merupakan salah satu cabang ilmu matematika yang dapat digunakan untuk menyelesaikan berbagai persoalan baik dalam kehidupan sehari-hari maupun persoalan kompleks yang lebih dalam. Makalah ini membahas aplikasi graf dalam menentukan lokasi tempat huni yang strategis berdasarkan jarak dari lokasi huni menuju tempat-tempat yang sering dikunjungi atau dibutuhkan. Dengan mengakumulasikan jarak terpendek menuju masing-masing tempat, maka dapat ditentukan lokasi paling strategis berdasarkan total jarak yang dibutuhkan.**

***Kata Kunci*—Matematika, graf, aplikasi, strategis, tempat, jarak terpendek.**

I. PENDAHULUAN

Tempat tinggal atau tempat huni merupakan salah satu kebutuhan manusia. Berbagai bentuk tempat tinggal sudah banyak tersedia, seperti rumah, apartemen, kost, dan masih banyak lagi. Kegunaan tempat tinggal bagi manusia sangatlah penting, contohnya adalah untuk beristirahat dan berlindung.

Tidak semua kebutuhan dapat dipenuhi dalam tempat tinggal. Banyak hal yang tidak dapat diperbaharui dalam tempat tinggal sendiri, seperti bahan makanan, kebutuhan perabot, tempat kerja, dan masih banyak lagi. Tentu, manusia bepergian ke luar tempat tinggal untuk memenuhi kebutuhan tersebut, seperti pergi bekerja ke kantor, belajar ke kampus, atau bahkan berlibur ke mall. Untuk itu, lokasi huni yang strategis cukup penting dalam menambahkan efisiensi waktu bepergian ke tempat-tempat tersebut. Selain menghemat waktu, biaya transportasi maupun bensin kendaraan pribadi juga semakin hemat.

Salah satu permasalahan klasik dalam matematika, khususnya graf, adalah menentukan jarak terdekat dari dua buah titik yang saling terhubung dengan titik lainnya. Hal ini disebut juga dengan *shortest path problem*. Berbagai implementasi masalah ini dilakukan untuk banyak sektor, seperti optimalisasi jaringan, desain sirkuit, dan pemetaan.

Berdasarkan hal tersebut, penulis menerapkan ilmu matematika yaitu graf untuk menghitung jarak terpendek dari lokasi tempat tinggal menuju tempat-tempat yang akan sering dikunjungi penghuni. Algoritma Dijkstra akan digunakan untuk menentukan jarak terpendek dari dua buah node.

II. TEORI DASAR

A. Graf

1. Definisi Graf[2]

Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut, sehingga secara sederhana graf didefinisikan sebagai kumpulan titik yang dihubungkan oleh garis-garis/sisi. Definisi matematis untuk graf adalah, pasangan terurut himpunan (V,E), dimana V merupakan himpunan beranggotakan titik-titik (*vertex*) dan E merupakan himpunan beranggotakan sisi-sisi (*edges*).

2. Jenis Graf

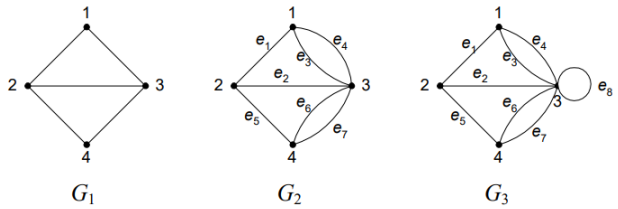
Graf dapat digolongkan berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, sehingga graf dapat dibagi jenisnya menjadi dua, yaitu:

2.1 *Graf Sederhana* (Simple Graph)

Graf sederhana memiliki ciri berupa tidak mengandung gelang maupun sisi ganda.

2.2 *Graf Tidak Sederhana* (Unsimple-graph)

Graf yang mengandung gelang atau sisi ganda disebut juga dengan graf tidak sederhana. Graf tidak sederhana ini dibagi lagi menjadi dua, yaitu graf ganda *(multi-graph)* dan graf semu *(pseudo-graph)*



Gambar 1: (a) G1 graf sederhana, (b) G2 graf ganda, (c) G3 graf semu (Sumber: [1])

Selain berdasarkan gelang atau sisi ganda suatu graf, graf juga dapat digolongkan berdasarkan orientasi arah pada sisi graf:

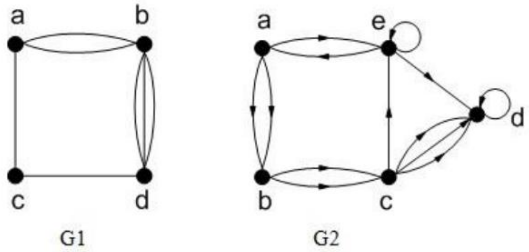
2.3 *Graf Tak-berarah* (Undirected Graph)

Graf yang sisinya tidak mempunyai orientasi arah disebut dengan graf tak-berarah. Graf ini hanya menghubungkan

sekumpulan node tetapi tidak ada arah yang menunjuk pada sisi-sisinya.

2.4 Graf Berarah (Directed Graph *atau* Digraph)

Graf yang setiap sisinya memiliki orientasi arah disebut dengan graf berarah. Masing-masing node yang terhubung dengan node lainnya memiliki sisi yang menunjukkan suatu arah.



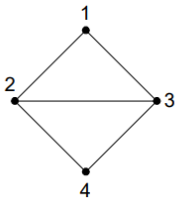
Gambar 2: G1 graf tak-berarah, G2 graf berarah
(Sumber: [1])

3. Terminologi Graf

Graf memiliki beberapa terminologi, antara lain:

3.1 Ketetanggaan (Adjacent)

Dua buah simpul *(node)* dikatakan bertetangga *(adjacent)* apabila keduanya terhubung langsung oleh satu sisi.



Gambar 3: Graf sederhana
(Sumber: [1])

Jika ditinjau dari graf pada gambar 3, maka dapat dilihat bahwa simpul 1 dikatakan bertetangga dengan simpul 2 dan 3, namun simpul 1 tidak bertetangga dengan simpul 4.

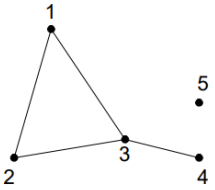
3.2 Bersisian (Incidency)

Untuk sembarang sisi *e = (vj, vk)* dikatan *e* bersisian dengan simpul *vj*, atau *e* bersisian dengan simpul *vk*.

Tinjau graf pada gambar 3, maka dapat dilihat bahwa sisi (2, 3) bersisian dengan simpul 2 dan simpul 3, tetapi sisi (1, 2) tidak bersisian dengan simpul 4.

3.3 Simpul Terpencil (Isolated Vortex)

Simpul yang tidak mempunyai sisi yang bersisian dengannya disebut dengan simpul terpencil.

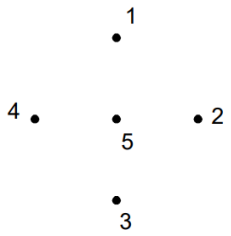


Gambar 4: Graf yang memiliki simpul terpencil
(Sumber: [1])

Tinjau graf pada gambar 4, maka dapat dilihat bahwa simpul 5 merupakan simpul terpencil karena tidak bersisian dengan simpul manapun lainnya.

3.4 Graf Kosong (null graph *atau* empty graph)

Graf yang himpunan sisinya merupakan himpunan kosong disebut dengan graf kosong.



Gambar 5: Graf N_5 (graf kosong)
(Sumber: [1])

3.5 Derajat (Degree)

Derajat dari suatu simpul adalah jumlah sisi yang bersisian dengan simpul tersebut.

Tinjau graf pada gambar 3, maka dapat dilihat bahwa

derajat pada simpul 1 ($d(1)$) = $d(4)$ = 2
$d(2)$ = $d(3)$ = 3

Tinjau graf pada gambar 4, maka dapat dilihat bahwa

$d(3)$ = 3
$d(4)$ = 1
$d(5)$ = 0

3.6 Lintasan (Path)

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

Pada gambar 3, lintasan dari 1 ke 4 dapat tuliskan seperti 1, 2, 4 atau 1, 2, 3, 4 atau 1, 3, 4.

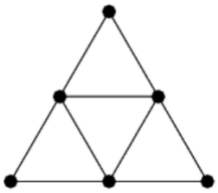
3.7 Siklus (Cycle) *atau Sirkuit* (Circuit)

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

Pada gambar 4, lintasan 4, 3, 1, 2, 3, 4 disebut sebagai sirkuit atau siklus.

3.8 Keterhubungan (Connected)

Dua buah simpul v1 dan simpul v2 disebut terhubung jika terdapat lintasan dari v1 ke v2. Graf 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.

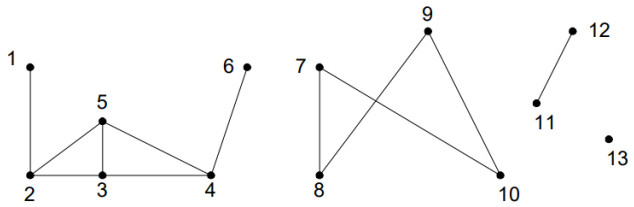


Gambar 6: Contoh graf terhubung
(Sumber: [1])

3.9 Upagraf (Subgraf) dan Komplemen Upagraf

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.

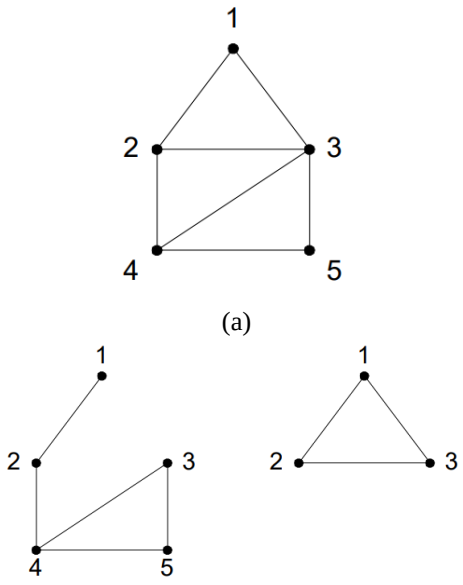
Komponen graf (connected component) adalah jumlah maksimum upagraf terhubung dalam graf G.



Gambar 7: Graf dengan 4 komponen
(Sumber: [1])

3.10 Upagraf Merentang (Spanning Subgraph)

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



Gambar 8: (a) Graf G (b) upagraf merentang dari G (c) bukan upagraf merentang dari G. (Sumber: [1])

3.11 Cut-set

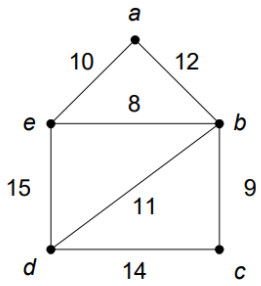
Cut-set dari graf terhubung G adalah himpunan sisi yang bila dibuang dari G menyebabkan G tidak terhubung.

Gambar 9: Graf yang dipecah menjadi 2 graf tak-terhubung
(Sumber: [1])

Graf pada gambar 9 memilici *cut-set* {(1,2), (1,5), (3,5), (3,4)}. Jika sisi-sisi tersebut dihilangkan, maka akan terbentuk 2 buah graf yang terpisah atau tak-terhubung.

3.12 Graf Berbobot (Weighted Graph)

Graf yang setiap sisinya diberi harga/nilai disebut juga dengan graf berbobot.



Gambar 10: Graf berbobot
(Sumber: [1])

B. Algoritma Dijkstra[3]

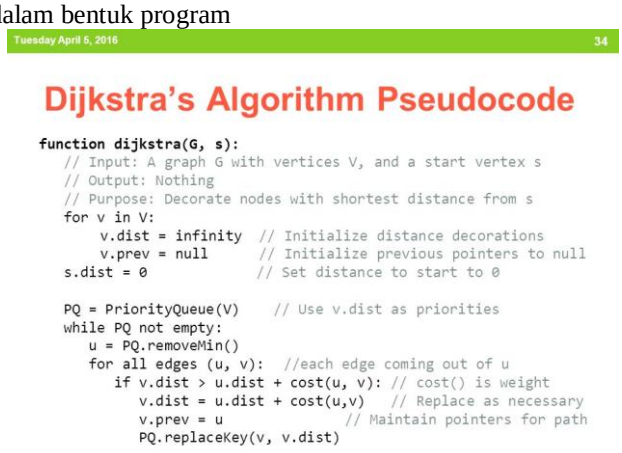
Algoritma Dijkstra adalah sebuah algoritma yang dipakai dalam memecahkan permasalahan jarak terpendek untuk sebuah graf berarah. Algoritma ini ditemukan oleh Edsger Dijkstra, seorang ilmuwan komputer yang berasal dari Belanda. Algoritma ini dipublikasikan pada tahun 1959 dalam jurnal *Numerische Mathematik* yang berjudul "*A Note on Two Problems in Connection with Graphs*".

Algoritma Dijkstra memiliki beberapa langkah, antara lain

1. Tentukan titik atau simpul yang akan menjadi simpul awal, lalu tentukan bobot jarak pada simpul pertama ke simpul tertangganya satu per satu.
2. Beri nilai bobot (jarak) untuk setiap titik ke titik lainnya, lalu set nilai 0 pada simpul awal dan nilai tak hingga terhadap node lain (belum terisi).
3. Tentukan semua simpul yang belum dilalui dan tentukan simpul awal sebagai "Simpul keberangkatan"
4. Dari simpul keberangkatan, tinjau simpul 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 telah selesai meninjau setiap jarak terhadap simpul tetangga, tandai simpul yang telah dilalui sebagai "Sudah dikunjungi". Simpul yang dilewati tidak akan pernah dicek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.

6. Tentukan “Simpul belum dilewati” dengan jarak terkecil (dari node keberangkatan) sebagai “Simpul Keberangkatan” selanjutnya dan ulangi langkah 5.

Berikut contoh *pseudocode* algoritma dijkstra untuk memperjelas gambaran algoritma dijkstra jika dijalankan dalam bentuk program

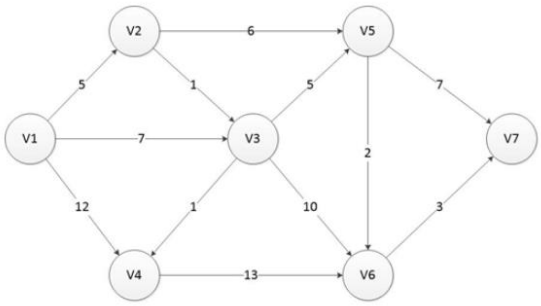


Gambar 11: *pseudocode* algoritma dijkstra
(Sumber: [4])

III. PEMBAHASAN DAN ANALISIS

A. Algoritma Dijkstra

Algoritma Dijkstra adalah salah satu algoritma yang digunakan untuk menentukan jarak terpendek dari dua buah node dalam suatu graf. Terdapat kasus seperti berikut



Gambar 12: Contoh permasalahan
Sumber: [3]

Misalkan permasalahan pada Graf 11 akan diselesaikan dengan menggunakan algoritma Dijkstra. Akan dicari rute terpendek dari V1 menuju V7. Maka, tabel yang akan terbentuk berupa

Sudah dikunjungi	Posisi sekarang	V2	V3	V4	V5	V6	V7
-	V1	5	7	12	∞	∞	∞
V1	V2	5	6	12	11	∞	∞
V1, V2	V3	5	6	7	11	16	∞
V1, V2, V3	V4	5	6	7	11	16	∞
V1, V2, V3, V4	V5	5	6	7	11	13	18
V1, V2, V3, V4, V5	V6	5	6	7	11	13	16

Tabel 2.1: Tabel hasil algoritma dijkstra
Sumber: penulis

Langkah pertama adalah menentukan simpul yang akan menjadi lokasi awal, pada kasus ini alokasi awalnya terdapat pada simpul V1. Kemudian, ditentukan nilai bobot dari V1 menuju tetangganya, yaitu menuju V2 sebesar 5, V3 sebesar 7, dan V4 sebesar 12. Simpan nilai pada kolom tabel yang bersesuaian dengan tetangga yang dituju. Isi nilai simpul yang belum dikunjungi dengan symbol tak-hingga. V1 kemudian dimasukkan dalam kolom “sudah dikunjungi”.

Tentukan nilai simpul yang memiliki bobot paling kecil dan belum dikunjungi, pada kasus ini adalah V2. Maka, posisi sekarang berada di V2. Tinjau tetangga dari V2, pada kasus ini adalah V3 dan V5. Nilai menuju simpul tetangga dari V2 bernilai sebesar bobot V2 + bobot menuju simpul dari V2. Dapat dilihat bahwa untuk menuju V3, maka nilai barunya adalah 5 + 1 = 6. Karena nilai barunya lebih kecil daripada nilai lama menuju V3 (6 < 7), maka perbarui nilai pada kolom V3 dengan nilai yang lebih kecil. Hal yang sama dilakukan untuk menuju V5, yaitu 5 + 6 = 11. Karena 11 < ∞, maka ubah nilai pada kolom V5. Simpan V2 dalam kolom “sudah dikunjungi”.

Simpul berikutnya yang belum pernah dikunjungi dan memiliki bobot paling kecil adalah V3. Tinjau tetangga dari simpul V3, yaitu V4, V5, dan V6. V3 menuju V4 memiliki bobot 7, dan karena 7 < 12 (bobot V4 sebelumnya) maka perbarui nilai kolom V4 tersebut. Hal yang sama dilakukan untuk V5 dan V6.

Simpul yang dikunjungi selanjutnya adalah V4. Simpul V4 hanya memiliki satu tetangga yaitu V6. Bobot untuk menuju V6 dari V4 adalah 7 + 13 = 20. Dapat dilihat bahwa nilai bobot yang baru ini lebih besar dibandingkan nilai bobot V6 sebelumnya. Oleh karena itu, nilai bobot V6 tetap dan tidak berubah.

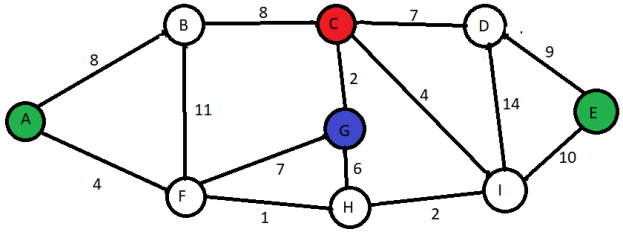
V5 bertetangga hanya dengan V6 dan V7. Bobot menuju V6 dari V5 adalah 11 + 2 = 13. Karena nilai baru ini lebih kecil dibandingkan dengan bobot nilai V6 sebelumnya, maka perbarui nilai pada kolom V6. Begitupun juga dengan V7, karena 11 + 7 < ∞ maka nilai pada kolom V7 diisi dengan nilai barunya.

Simpul terakhir yang belum dikunjungi adalah simpul V6. Maka, akan ditinjau simpul V6 dan tetangganya, yaitu V7. Bobot nilai menuju V7 dari V6 adalah 13 + 3 = 16. Dapat dilihat bahwa 16 < 18 (bobot nilai V7 sebelumnya). Oleh karena itu, bobot nilai V7 diperbarui.

Didapatkan hasil akhir dari tabel, yaitu untuk mencapai V7 dari V1 hanya diperlukan bobot minimum sebesar 16. Rute dengan bobot 16 adalah V1-V2-V3-V5-V6-V7. Maka, dapat disimpulkan bahwa rute V1-V2-V3-V5-V6-V7 adalah rute terpendek dari V1 menuju V7.

B. Menentukan Lokasi Huni Strategis

Pada bagian ini, penulis akan menjelaskan cara menentukan lokasi huni strategis yang direpresentasikan oleh suatu graf. Suatu daerah dapat direpresentasikan dalam sebuah graf, pada persoalan ini contohnya



Gambar 13: Graf persoalan lokasi huni
Sumber: penulis

Keterangan:
Simpul merah (C) : Tempat berbelanja (supermarket)
Simpul biru (G) : Tempat rekreasi (mall)
Simbul hijau (A & E) : Lokasi huni (rumah, apartemen)
Simpul putih : Persimpangan jalan

Pada gambar 13, dapat dilihat ada suatu graf yang merepresentasikan yang merepresentasikan suatu daerah pada kota. Simpul berwarna hijau merupakan representasi lokasi huni (komplek perumahan, apartemen, kos, dan sebagainya). Simpul berwarna merah merepresentasikan tempat berbelanja kebutuhan sehari-hari seperti supermarket, sedangkan simbul berwarna biru merepresentasikan tempat rekreasi dan bersenang-senang seperti mall. Sisi pada setiap simpul direpresentasikan sebagai jalur dua arah.

Permasalahan ini dapat diselesaikan dengan meninjau jarak terpendek dari masing-masing lokasi huni ke simpul merah dan biru. Dengan menjumlahkan bobot akhir keduanya, maka dapat dibandingkan untuk mendapatkan lokasi huni paling strategis, yaitu dengan bobot nilai akhir yang lebih kecil.

1. Lokasi Huni A

Dengan menggunakan algoritma dijkstra, didapatkan tabel sebagai berikut.

Sudah dikunjungi	Posisi sekarang	B	C	D	E	F	G	H	I
-	A	8	∞	∞	∞	4	∞	∞	∞
A	F	8	∞	∞	∞	4	11	5	∞
A, F	H	8	∞	∞	∞	4	11	5	7
A, F, H	I	8	11	∞	∞	4	11	5	7
A, F, H, I	B	8	11	∞	17	4	11	5	7
A, F, H, I, B	C	8	11	18	17	4	11	5	7
A, F, H, I, B, C	G	8	11	18	17	4	11	5	7
A, F, H, I, B, C, G	D	8	11	18	17	4	11	5	7

Tabel 3.1: Algoritma Dijkstra pada lokasi huni A
Sumber: penulis
Keterangan warna kuning: sedang dikunjungi/kolom telah dikunjungi

Berdasarkan hasil tabel 1, maka dapat ditentukan bahwa jarak menuju tempat berbelanja (simpul merah) atau simpul C memiliki bobot minimum 11 dengan rute A-F-H-I. Untuk menuju ke tempat rekreasi (simpul biru) atau simpul G, rute dengan bobot minimum tersebut adalah 11 dengan rute A-F-G. Lokasi huni A memiliki nilai strategis berupa jumlah kedua rute menuju dua lokasi tersebut, yaitu 22.

2. Lokasi Huni B

Untuk menentukan jarak terpendek dari B menuju simpul merah dan biru, digunakan algoritma dijkstra sehingga menghasilkan tabel berikut

Sudah dikunjungi	Posisi sekarang	A	B	C	D	F	G	H	I
-	E	∞	∞	∞	9	∞	∞	∞	10
E	D	∞	∞	16	9	∞	∞	∞	10
E, D	I	∞	∞	14	9	∞	∞	12	10
E, D, I	H			14	9	13	18	12	10
E, D, I, H	F	17	24	14	9	13	18	12	10
E, D, I, H, F	C	17	22	14	9	13	16	12	10
E, D, I, H, F, C	G	17	22	14	9	13	16	12	10
E, D, I, H, F, C, G	A	17	22	14	9	13	16	12	10

Tabel 3.2: Algoritma Dijkstra pada lokasi huni B
Sumber: penulis
Keterangan warna kuning: sedang dikunjungi/kolom telah dikunjungi

Berdasarkan hasil pada tabel satu, bobot minimum dari simpul E menuju simpul C atau tempat berbelanja adalah 14 dengan rute E-I-C. Untuk menuju simpul G atau tempat rekreasi, bobot minimum dari simpul E adalah 16 dengan rute E-I-C-G. Oleh karena itu, lokasi huni E memiliki nilai strategis sejumlah 30.

IV. KESIMPULAN

Graf dapat diaplikasikan untuk merepresentasikan berbagai masalah, contohnya adalah untuk menggambarkan lokasi pada suatu daerah yang terhubung oleh jalan. Penggunaan graf ini dapat ditujukan untuk menyelesaikan masalah seperti jarak terdekat, cara melewati semua jalan minimal satu kali, atau cara melewati semua lokasi minimal sekali dengan jarak minimum.

Salah satu aplikasi graf adalah untuk menentukan jarak terdekat. Ada berbagai algoritma yang dapat digunakan untuk menentukan jarak terdekat antara dua buah node, salah satunya adalah Algoritma Dijkstra. Algoritma ini mencari rute jalan dengan bobot nilai terkecil, kemudian menelusuri jalan menuju tetangganya. Kemudian, ditentukan lagi dari simpul yang belum pernah ditelusuri lalu ditentukan lagi bobot jalan menuju tetangganya, dan ulangi hingga semua simpul telah terlewati.

Pada permasalahan yang penulis analisis, algoritma ini digunakan untuk menentukan lokasi strategis jika terdapat dua lokasi huni berbeda pada suatu daerah. Suatu lokasi huni disebut strategis apabila ia memiliki bobot nilai paling minimum menuju tempat-tempat yang akan sering dikunjungi, seperti tempat berbelanja kebutuhan sehari-hari serta tempat rekreasi untuk bersenang-senang.

Dapat disimpulkan bahwa untuk permasalahan pada Bab 3

(gambar 13), terdapat dua buah lokasi huni yaitu simpul A dan E serta lokasi berbelanja yaitu simpul C dan lokasi rekreasi yaitu simpul G. Berdasarkan hasil analisis dan perhitungan algoritma yang telah dilakukan, dapat disimpulkan bahwa lokasi huni pada simpul A lebih strategis dibandingkan dengan lokasi huni pada simpul E karena bobot nilai total pada simpul A lebih sedikit dibandingkan dengan simpul E.

REFERENCES

[1] IF2120 Matematika Diskrit – Semester I Tahun 2020/2021 [Internet] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/matdis20-21.htm diakses 9 Desember 2020 pukul 14.17.

[2] Pengertian Graf [Internet] https://lmsspada.kemdikbud.go.id/mod/resource/view.php?id=47638 diakses 9 Desember 2020 pukul 14.16.

[3] Algoritma Dijkstra [Internet] https://mti.binus.ac.id/2017/11/28/algoritma-dijkstra/ diakses 11 Desember 2020 pukul 19.21

[4] SHORTEST PATH IN GRAPHS [Internet] https://slideplayer.com/slide/10658514/ diakses 11 Desember 2020 pukul 22.52.

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, 11 Desember 2020

Rafi Raihansyah Munandar (13519154)