BAB II

TINJAUAN PUSTAKA

2.1. Teori Graf

2.1.1. Definisi Graf

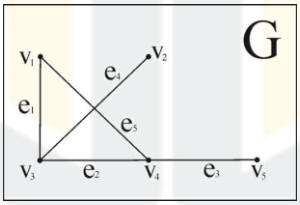
Graf digunakan untuk mempresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Kumpulan titik yang dihubungkan satu sama lain melalui rusuk disebut graf. Graf G adalah pasangan himpunan (V, E) dengan V adalah himpunan tidak kosong dari titik-titik G dan E adalah himpunan (munkin kosong) sisi dari G yang menghubungkan dua buah titik.

Contoh 2.1. Berikut ini adalah graf G yang berisi himpunan sisi E dan himpunan titik V.

$$\mathrm{V} = \{v_1, v_2, v_3, v_4, v_5\}$$

$$\mathrm{E} = \{e_1, e_2, e_3, e_4, e_5\}$$

Graf G tersebut dapat digambar sebagai berikut:



Gambar 2.1 Graf G

2.1.2. Jenis - Jenis Graf

Secara umum, graf dapat digolongkan berdasarkan ada atau tidaknya gelang atau sisi ganda pada suatu graf. Jenis graf tersebut adalah graf sederhana dan graf tidak sederhana.

a. Graf sederhana (*simple graph)*

   Graf sederhana adalah graf yang tidak mengandung *loop* maupun sisi ganda. Sebuah sisi yang berawal dan berakhir pada simpul yang sama disebut *loop*.

b. Graf tak-sederhana (*unsimple graph*)

Graf tak-sederhana yaitu graf yang mengandung *loop* atau sisi ganda. Ada dua macam graf tak-sederhana, yaitu graf ganda dan graf semu.

1. Graf ganda

   Graf ganda adalah graf yang mengandung sisi ganda. Sepasang simpul dapat dihubungkan oleh lebih dari dua sisi ganda.

2. Graf semu

   Graf semu adalah graf yang mengandung *loop*. Graf semu lebih umum daripada graf ganda, karena graf semu dapat terhubung kedirinya sendiri.

Contoh 2.2 Sebagai ilustrasi dari Graf tersebut, diberikan $V = \{v_1, v_2, v_3, v_4\}$ dan E_1, E_2, E_3 adalah himpunan sisi sebagai berikut:

a) Graf sederhana G_1 memuat himpunan titik V dan himpunan rusuk E_1 sebagai berikut ini:

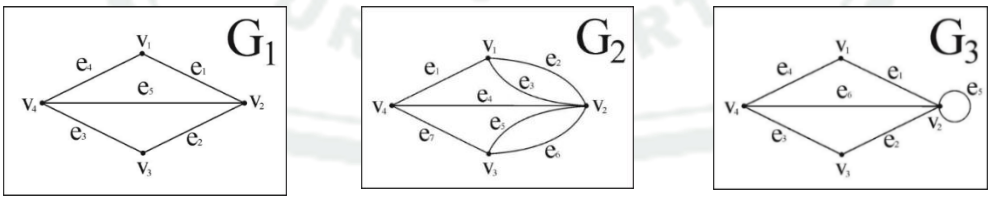
$$\mathrm{V} = \{ v_1,\ v_2,\ v_3,\ v_4\}$$
$$\mathrm{E} = \{e_1, e_2, e_3, e_4, e_5\}$$

b) Graf ganda G_2 memuat himpunan titik V dan himpunan sisi E_2 sebagai berikut :

$$V = \{v_1,\ v_2,\ v_3,\ v_4\}$$
$$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$$

c) Graf semu G_3 memuat himpunan simpul V dan himpunan sisi E_3 sebagai berikut:

$$V = \{v_1,\ v_2,\ v_3,\ v_4\}$$
$$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$$



a) Graf Sederhana b) Graf Ganda c) Graf Semu

Gambar 2.2 Graf Sederhana, Graf Ganda dan Graf Semu

Sedangkan, berdasarkan orientasi arah pada sisi, maka secara umum graf dibedakan atas dua jenis:

a. Graf tak berarah (*undirect grap*h)

   Graf tak berarah adalah graf yang rusuknya tidak mempunyai orientasi arah. Urutan pasangan titik yang terhubung oleh rusuk tidak diperhatikan. Pada graf tak berarah $(v_1, v_2) = (v_2, v_1)$ adalah rusuk yang sama.

b. Graf berarah (*direct graph* atau *digraph*)

   Graf berarah adalah graf yang setiap rusuknya diberikan orientasi arah. Pada graf berarah (v_1, v_2) dan (v_2, v_1) menyatakan dua buah rusuk yang berbeda, dalam arti kata bahwa $(v_1, v_2) \neq (v_2, v_1)$. Jadi untuk rusuk (v_1, v_2) titik titik v_1 disebut titik asal dan titik v_2 disebut titik terminal atau titik tujuan.

Contoh 2.3 Sebagai ilustrasi dari Graf tersebut, diberikan $V = \{v_1, v_2, v_3\}$ dan $E = \{e_1, e_2, e_3\}$ adalah himpunan sisi sebagai berikut:

a) Graf tak berarah menunjukan titik yang terhubung sebagai berikut :

$$V = (v_1, v_2) = (v_2, v_1), (v_2, v_3) = (v_3, v_2), (v_3, v_4) = (v_4, v_3), (v_4, v_1) = (v_1, v_4)$$

b) Graf berarah menunjukan titik yang terhubung sebagai berikut :

$$V = (v_1, v_3), (v_3, v_2), (v_2, v_1)$$

a) Graf Tak Berarah b) Graf Berarah

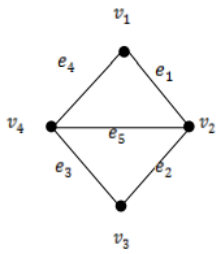
Gambar 2.3 Graf Tak Berarah dan Graf Berarah

2.1.3. Terminologi Dasar Graf

a. Bertetangga (*Adacent*)

   Dua buah simpul dikatakan bertetangga bila keduanya terhubung langsung dengan sebuah sisi. Simpul v_j bertetangga dengan v_k jika (v_j, v_k) adalah sebuah sisi pada graf G (Munir, 2005).

Bertetangga ditunjukan pada Gambar 2.4



Gambar 2.4 Contoh simpul bertetangga

Pada Gambar 2.4 simpul v_1 bertetangga dengan simpul v_2 dan v_4, simpul v_3 tidak bertetangga dengan simpul v_1.

b. Bersisian *(Incodency)*

Untuk sembarang sisi $e = (v_j, v_k)$, sisi e dikatakan bersisian dengan titik v_j dan titik v_k (Munir, 2005). Pada Gambar 2.4, sisi e_1 bersisian dengan titik v_1 dan titik v_2, tetapi sisi e_1 tidak bersisian dengan titik v_4.

c. Graf Kosong (*Null Graph*)

Graf yang himpunan sisinya merupakan himpunan kosong disebut Graf Kosong (*Null Graph*) dan ditulis sebagai N_n, n adalah jumlah titik (Munir, 2005). Graf kosong adalah graf yang tidak memiliki sisi. Graf kosong ditunjukan pada Gambar 2.5

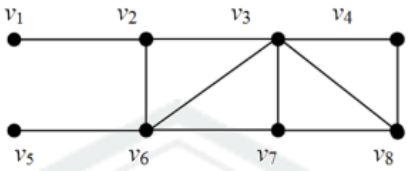
Gambar 2.5 Graf Kosong

d. Derajat (*Degree)*

Banyaknya sisi yang mengelilingi sebuah simpul pada sebuah graf tak berarah disebut derajat (Munir, 2005). Pada Gambar 2.4, graf G_1: $d(v_1) = d(v_3) = 2, d(v_2) = d(v_4) = 3$.

e. Jalan (*Walk*)

Susunan titik-titik dalam graf yang dimulai dari titik v_1 dan berakhir di titik v_n disebut jalan (*walk)*. Panjang dari sebuah jalan (*walk)* ditentukan oleh berapa banyak sisi yang dimilikinya.



Gambar 2.6 *Walk*

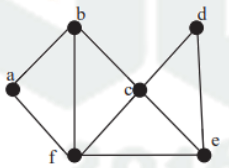
Gambar 2.6 memperlihatkan $W = v_1, v_2, v_3, v_6, v_2, v_3, v_8$ dengan *walk* 6.

f. Lintasan (*Path*)

Lintasan adalah barisan titik – titik dalam graf yang semua titiknya berlainan. Contoh lintasan dapat dilihat pada gambar 2.6 $P = v_1, v_2, v_6, v_3, v_8$ dengan Panjang lintasan 4.

g. Sirkuit (*Circuit)*

Sirkuit adalah lintasan yang berawal dan berakhir pada simpul yang sama. Sebuah sirkuit dikatakan sirkuit sederhana jika setiap rusuk yang dilalui berbeda. Contoh sirkuit dari graf pada Gambar 2.7, adalah a-b-c-d-e-f-a.



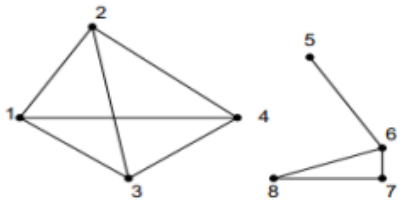
Gambar 2.7 Contoh Sirkuit

h. Terhubung (*Connected*)

Ada dua cara untuk mendefinisikan sebuah graf terhubung, yaitu untuk graf tak berarah dan untuk graf berarah.

1) Graf tak berarah G dikatakan terhubung (*connected graph*) jika ada lintasan di dari u ke v untuk setiap pasang simpul u dan v di dalam himpunan V. hal tersebut menyiratkan bahwa harus ada lintasan dari v ke u. Sebaliknya, maka G disebut graf tak terhubung (*disconnected graph*).

Gambar 2.8 adalah contoh dari graf tak berarah yang terhubung.



Gambar 2.8 Graf Tak Berarah yang Terhubung

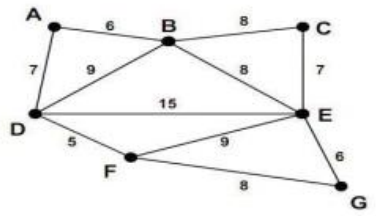
2) Graf berarah G dikatakan terhubung jika graf tak berarahnya terhubung (graf tak berarah dari G diperoleh dengan menghilangkan arahya) (Munir, 2005). Pada graf berarah, keterhubungan dua buah simpul dibedakan menjadi dua, yaitu terhubung kuat dan terhubung lemah.

Gambar 2.9 Graf Berarah Terhubung

Graf pada Gambar 2.9 (a) merupakan graf terhubung kuat, karena untuk sembarang sepasang simpul di dalam graf tersebut terdapat lintasan. Sedangkan graf pada Gambar 2.9 (b) merupakan graf terhubung lemah, karena tidak semua pasangan simpul mempunyai lintasan arah.

i. Graf Berbobot (*Weighted Graph*)

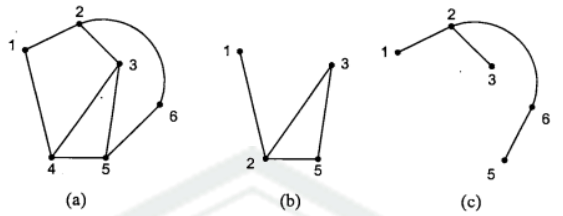
Graf dengan sebuah harga (bobot) yang diberikan pada setiap rusuknya disebut graf berbobot (Munir, 2005). Tergantung dari permasalahan yang ingin diilistrasikan oleh graf tersebut, bobot pada setiap rusuknya dapat berbeda-beda. Jarak antara dua buah kota, biaya perjalanan antar dua buah kota, ongkos produksi, dan factor lain dapat diwakili oleh bobot pada graf berbobot.



Gambar 2.10 Graf Berbobot

j. Graf Bagian (*Subgraf)* dan Komplemen Graf Bagian

Misalkan $G = (V, E)$ adalah sebuah graf. $G_1 = (V_1, E_1)$ adalah graf bagian dari G. Komplemen dari graf bagian dari G adalah $G_2 = (V_2, E_2)$



Gambar 2.11 (a) Graf G, (b) Graf G_1, (c) Graf G_2

Komplemen dari G_1 terhadap graf G adalah graf $G_2 = (V_2, E_2)$ sedemikian sehingga $E_2 = E - E_1$ dan V_2 adalah himpunan simpul yang anggota–anggota E_2 bersisian dengannya.

k. *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.

Gambar 2.12 Graf *Cut – Set*

Pada Gambar 2.12, {(1,2), (1,5), (3,5), (3,4)} adalah *cut–set*. Himpunan {(1,2), (1,5)} dan himpunan {(2,6)} juga merupakan *cut–set*, tetapi {(1,2), (2,5), (4,5)} bukan *cut–set* sebab himpunan bagian {(1,2), (2,5)} adalah *cut–set*.

2.2. Algoritma Dijkstra

Algoritma yang popular untuk menentukan rute terpendek adalah Algoritma Dijkstra. Algoritma ini sederhana dalam penggunaanya dengan menggunakan simpul-simpul sederhana pada jaringan jalan yang tidak rumit.. Edsger W. Dijkstra merupakan pencipta algoritma ini dimana hal tersebut sumber dari nama algoritma ini. Algoritma Dijkstra bekerja dengan cara mencari jalur antara dua simpul dalam

sebuah graf berbobot dengan mencari jarak terpendek antara simpul awal dan simpul lain sehingga terbentuk simpul awal hingga simpul tujuan dengan bobot terkecil (Ismantohadi & Iryanto, 2018). Untuk mendapatkan solusi, Algoritma Dijkstra menggunakan prinsip mencari jawaban terbaik di setiap tahap untuk mendapatkan solusi optimum pada langkah selanjutnya yang akan mengarah pada solusi terbaik. Hal ini membuat kompleksitas waktu Algoritma Dijkstra menjadi cukup besar.

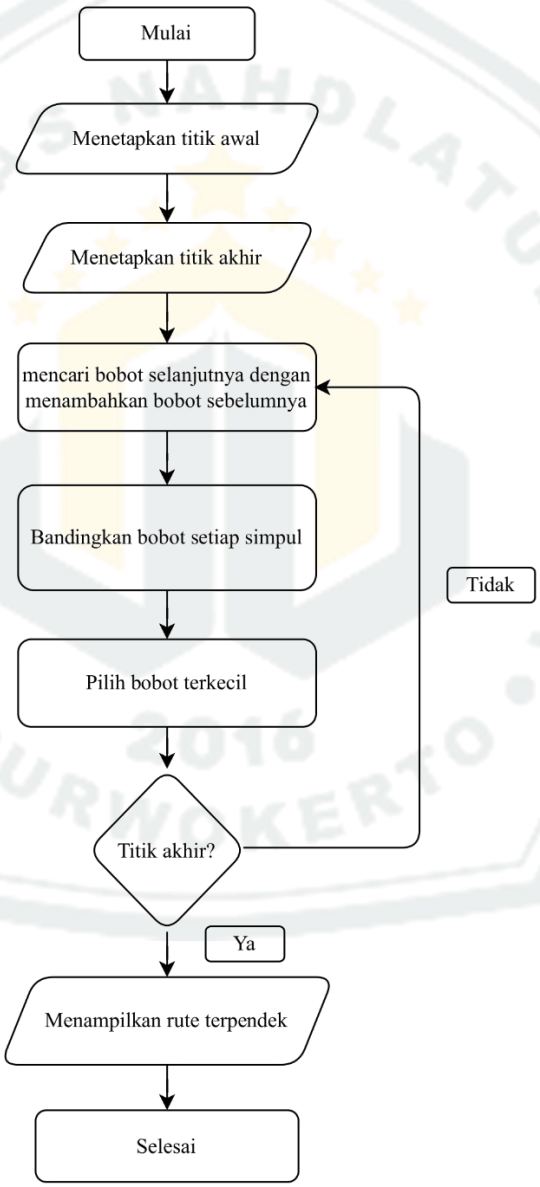
Cara kerja Algoritma Dijkstra memanfaatkan struktur data berprioritas untuk memilih simpul dengan bobot paling kecil (Putro et al.,2019). Secara umum, Tahapan Algoritma Dijkstra ini dapat dilakukan dengan tahapan berikut:

1. Tentukan titik awal dimana titik tersebut akan menjadi *node* pertama atau *node* awal.
2. Tentukan bobot (jarak) dari setiap *node* ke *node* lainnya.
3. Tandai semua *node* yang belum dilalui dan tandai *node* awal sebagai "*node* keberangkatan".
4. Dari *node* keberangkatan, pertimpangkan *node* yang belum dilalui dan perhitungkan jarak dari *node* awal keberangkatan.
5. Setelah kita selesai mengitung dan mempertimbangkan setiap jarak pada *node* lainnya yang berdekatan, tandai *node* yang sudah dilalui sebagai "*node* dilewati". *Node* yang sudah dilewati tidak akan pernah dicek kembali, jarak yang akan diambil adalah jarak dengan bobot paling kecil bobotnya,
6. Perhitungkan *node* belum dilewati, dengan bobot jarak paling kecil dari *node* keberangkatan sebagai *node* keberangkatan untuk keberangkatan selanjutnya dan ulangi langkah.

Dengan demikian hanya simpul yang paling penting yang akan ditelusuri. Pendekatan ini membandingkan nilai (bobot) setiap simpul pada satu tingkat untuk menentukan simpul mana yang diprioritaskan. Setelah ini nilai (bobot) dari setiap simpul tersebut disimpan untuk dibandingkan dengan nilai yang akan ditemukan dari rute yang baru ditemukan kemudian, begitu seterusnya sampai ditemukan simpul yang dicari. Algoritma Dijkstra digunakan untuk mengidentifikasi jalur terpendek dari sebuah graf, algoritma ini menemukan jalur yang optimal karena

ketika memilih jalur dari simpul – simpul yang belum terpilih dan simpul dengan bobot terendah yang akan dipilih. Oleh karena itu, algoritma ini sering digunakan dalam berbagai bidang yang memerlukan perhitungan jalur terpendek seperti perencanaan tranportasi. Algoritma Dijkstra mencari jarak terpendek dari *node* asal ke *node* terdekatnya, kemudian ke *node* kedua, dan seterusnya. Berikut ini adalah contoh dari graf yang akan diselesaikan dengan Algoritma Dijkstra.

Berikut *flowchart* untuk menggambarkan langkah langkah Algoritma Dijkstra:



Gambar 2.13 *Flowchart* Algoritma Dijkstra