Aplikasi Graf dalam Menentukan Jarak Tempuh Perjalanan Terpendek di Kota Bandung

Lukas Kurnia Jonathan - 13517006
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517006@std.stei.itb.ac.id

***Abstraksi*—Graf memiliki banyak sekali aplikasi kegunaan yang dapat dipakai untuk memecahkan berbagai persoalan, baik permasalahan kehidupan sehari-hari maupun hal yang lebih kompleks. Salah satu persoalan yang bisa diselesaikan menggunakan graf adalah mencari besar bobot minimum antara dua buah *node* yang bisa direpresentasikan sebagai jarak minimum 2 buah titik. Makalah ini membahas mengenai aplikasi graf dalam menentukan *shortest path* di kota Bandung, menggunakan salah satu algoritma pencarian jarak terpendek yaitu algoritma Dijsktra.**

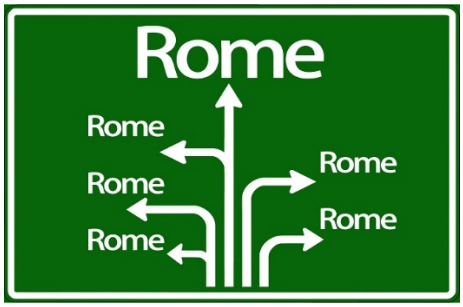
***Kata kunci*— Algoritma Dijkstra, Bandung, graf, jarak terpendek, *node*.**

I. PENDAHULUAN

Dewasa ini, dengan kemajuan teknologi dan ilmu pengetahuan yang terus menerus berkembang, manusia cenderung menginginkan suatu hal dengan mudah, murah, dan instan. Semua orang tentu akan senang apabila dapat melakukan suatu pekerjaan dengan efisien dan praktis. Untuk mencapai hal tersebut maka manusia akan men-*cut* opsi-opsi atau pilihan yang membuat sebuah pekerjaan menjadi tidak efisien. Sebaliknya, akan mengusahakan opsi atau pilihan yang meningkatkan dan memermudah suatu pekerjaan.

Perjalanan merupakan salah satu pekerjaan yang dilakukan seseorang untuk mencapai tujuan dari titik asal. Perjalanan dari satu tempat ke tempat lain terkadang tidak dibatasi hanya satu rute atau jalan, tetapi dapat ditempuh dengan melewati berbagai macam variasi rute lainnya. Sebagai contoh, di Bandung, perjalanan yang dilakukan dari tempat tinggal seseorang ke kampus ITB bisa melalui berbagai jalan alternatif, perjalanan dari tempat kos ke alun-alun Bandung dapat menggunakan berbagai alternatif jalan, bahkan tidak jarang jalan yang dilalui ketika berangkat dan pulang berbeda dikarenakan kondisi jalan yang satu arah ataupun dua arah di kota Bandung.

Seperti kata pepatah, *banyak jalan menuju Roma*, menunjukkan untuk mencapai tujuan atau destinasi akhir, banyak variasi jalan yang dapat ditempuh. Hal yang membedakan antara suatu jalan dengan jalan yang lain tentunya adalah jarak tempuh perjalanan yang diperoleh ketika melalui jalan tersebut.



Gambar 1. banyak jalan menuju roma (Sumber: https://www.kompasiana.com/arisperd/5894dc64959373f736948da6/jangan-khawatir-banyak-jalan-menuju-roma?page=all)

Melihat hal tersebut, penulis menggunakan salah satu cabang ilmu matematika, yaitu graf, untuk menentukan jarak tempuh perjalanan terpendek di Kota Bandung. Penelusuran jarak terpendek dilakukan dengan menerapkan salah satu algoritma pencarian *shortest path* yaitu Algoritma Dijkstra.

II. TEORI DASAR

A. Kota Bandung

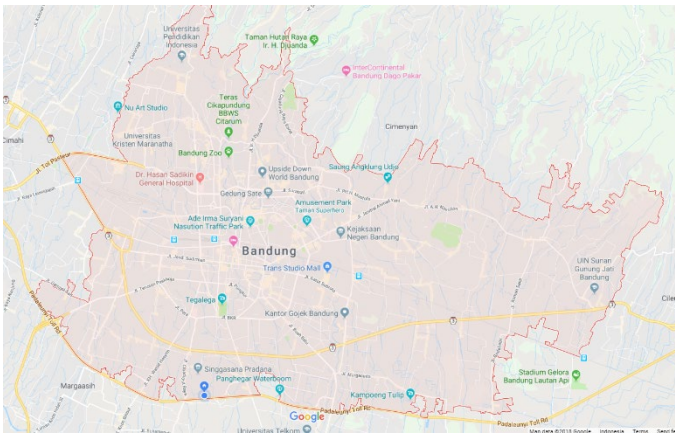
Indonesia adalah negara kepulauan yang besar, terdiri dari 34 provinsi, dimana setiap provinsi memiliki kota yang tersebar di berbagai daerah. Setiap kota memiliki luas yang berbeda dan penataan wilayah kota yang berbeda antara suatu kota dengan kota lainnya. Bandung merupakan salah satu kota di provinsi Jawa Barat dengan bentuk permukaan wilayah yang cukup unik. Selain berada di dataran tinggi, kota Bandung memiliki bentuk permukaan yang menyerupai cekungan dengan pusatnya berada di tengah kota.

Secara Geografis, Bandung terletak di bagian tengah Jawa Barat dengan ketinggian kurang lebih 768 m di atas permukaan laut. Kota Bandung memiliki luas wilayah sebesar 16.713 hektar, terbagi atas 30 kecamatan, 151 kelurahan, 1.561 RW, dan 9.691 RT. Kecamatan terluas di kota Bandung terletak di daerah selatan Bandung, yaitu kecamatan Gede Bage dengan luas 958 hektar. Sedangkan kecamatan terkecil juga masih terletak di bagian selatan Bandung, yaitu kecamatan Astana Anyar dengan luas 89 hektar[2].

Jumlah penduduk di kota Bandung yang tercatat pada tahun 2012 kurang lebih sebanyak 2.650.000 jiwa, terdiri dari sekitar 1.350.000 laki-laki dan 1.300.000 wanita[2].

Dari aspek pemerintahan, Bandung dipimpin oleh seorang

walikota dan seorang wakil walikota. Terdapat juga sekretaris daerah yang membawahi asisten sekretaris daerah, kepala dinas, kepala badan, kepala bagian, kepala kantor, perusahan daerah, inspektorat, dan kepala satuan polisi pamong praja[2].



Gambar 2. Peta kota Bandung (wilayah dengan warna merah)
(Sumber: maps.google.com)

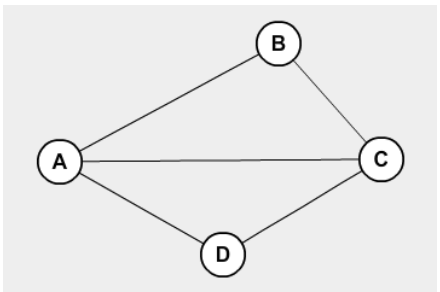
B. Graf

Graf adalah pokok bahasan yang sudah sangat tua, namun masih memiliki banyak sekali penerapan di masa kini. Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek tertentu[1].

1. Definisi Graf [1].

Secara matematis, graf dapat didefinisikan sebagai pasangan himpunan (V,E). Dimana V adalah himpunan tidak kosong dari simpul-simpul (*node*), dan E adalah himpunan dari sisi (*edge*) yang menghubungkan antara dua buah simpul. Notasi yang dipakai untuk menyatakan sebuah graf adalah G = (V,E).

Simpul atau *node* dalam graf dapat dinamai dengan huruf seperti a,b,c,d, .. ,z , dengan angka 1,2,3 , … ,9 , atau gabungan keduanya. Sedangkan sisi atau edge dituliskan sebagai pasangan antara kedua buah simpul. Misalkan terdapat sisi yang menghubungkan simpul antara a dan b, maka E = (a,b) .



Gambar 3. Contoh graf sederhana
(Sumber: buatan penulis)

Dari gambar 3, kita dapat menentukan himpunan simpul V dan sisi E yaitu

V = {A,B,C,D}

E = { (A,B), (B,C), (A,C), (A,D), (D,C) }

2. Jenis-Jenis Graf [1]

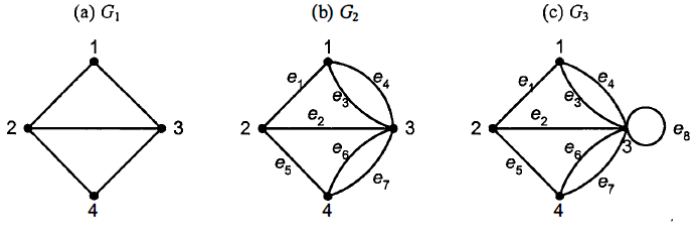
Berdasarkan ada tidaknya gelang atau sisi ganda pada sebuah graf, maka secara umum graf dapat dibagi menjadi dua jenis, yaitu graf sederhana dan graf tidak sederhana.

2.1 Graf Sederhana (*simple graph*)

Graf yang tidak memiliki sisi ganda disebut graf sederhana. Gambar 4(a) adalah contoh graf sederhana. Pada graf sederhana tidak terdapat sisi dengan pasangan terurut. Sehingga sisi (1,2) akan sama dengan (2,1).

2.2 Graf Tidak Sederhana (*unsimple graph*)

Graf yang mengandung sisi ganda atau gelang dinamakan graf tidak sederhana. Terdapat dua macam graf tidak sederhana, yaitu graf ganda dan graf semu. Graf ganda memiliki sisi ganda, sedangkan graf semu memiliki sisi gelang (*loop*).



Gambar 4 (a) Graf Sederhana (b) Graf Ganda (c) Graf Semu
(Sumber : [1])

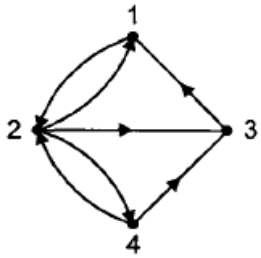
Selain berdasarkan ada tidaknya gelang atau sisi ganda, graf juga dapat dibagi sesuai orientasi arahnya. Berdasarkan orientasi arahnya graf dibedakan menjadi graf tidak berarah dan graf berarah.

2.3 Graf Tidak Berarah

Graf tidak berarah adalah graf yang ujung sisinya tidak mempunyai orientasi arah. Hal ini mengakibatkan sisi (a,b) dan (b,a) menjadi sisi yang sama. Ketiga buah graf pada Gambar 4 adalah graf tidak berarah.

2.4 Graf Berarah

Graf berarah adalah graf yang memiliki orientasi arah pada ujung sisinya. Dalam graf berarah, sisi (a,b) dan (b,a) adalah dua buah sisi yang berbeda.



Gambar 5. Graf Berarah
(Sumber : [1])

3. Terminologi Graf[1].

3.1 Bertetangga (*Adjacent*)

Bertetangga adalah kondisi dimana kedua buah simpul terhubung pada sebuah sisi yang sama. Pada gambar 4(a) simpul 1 bertetangga dengan 2 dan 3, demikian juga dengan simpul 2

bertetangga dengan simpul 1, 3 dan 4.

3.2 Bersisian (*Incident*)

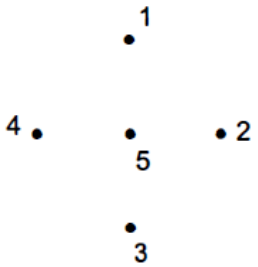
Sebuah sisi e(u,v) dikatakan bersisian dengan u dan bersisian dengan v. Sebagai contoh pada gambar 4(a) sisi (1,3) bersisian dengan simpul 1 dan simpul 3.

3.3 Simpul Terpencil (*Isolated vertex*)

Simpul terpencil adalah simpul yang tidak mempunyai simpul lain yang bertetangga dengan simpul tersebut. Pada gambar 6, semua simpul adalah simpul terpencil.

3.4 Graf Kosong (*null graph*)

Graf yang memiliki himpunan sisi kosong (tidak memiliki sisi sama sekali).



Gambar 6. Graf kosong dengan banyaknya simpul 5 buah
(Sumber : [1])

3.5 Derajat (*degree*)

Derajat adalah banyaknya sisi yang terhubung dengan sebuah simpul atau banyaknya sisi yang bersisian (*incident*) dengan simpul tersebut. Notasi : *d(v)* menyatakan derajat simpul v. Sebagai contoh, pada gambar 4(c)

d(1) = d(4) = d(2) = 3
d(3) = 7

d(3) memiliki derajat sebanyak 7 dikarenakan terdapat sebuah sisi, dua buah sisi ganda, dan sebuah sisi kalang atau *loop*. Derajat simpul yang memiliki sebuah *loop* bernilai 2.

3.6 Lintasan *(path)*

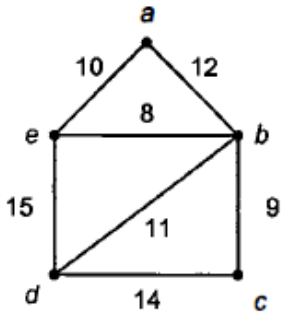
Lintasan adalah barisan berselang-seling simpul-simpul dan sisi-sisi yang berbentuk $v_0, e_1, v_1, e_2, v_2, \ldots v_{n-1}, e_n, v_n$ sedemikian sehingga $e_1 = (v_0,v_1)$, $e_2 = (v_1,v_2)$, …. , $e_n = (v_{n-1},v_n)$ adalah sisi-sisi dari sebuah graf. Pada gambar 4(a) lintasan dari simpul 1 ke simpul 4 dapat dituliskan sebagai 1,2,4 atau 1,2,3,4 atau 1,2,4.

3.7 Siklus (*cycle*) atau sirkuit (*circuit*)

Siklus adalah sebuah lintasan yang berawal dan juga berakhir di sebuah simpul yang sama. Sebagai contoh, pada gambar 4(a), 1,2,4,3,1 adalah sebuah sirkuit.

3.8 Graf Berbobot (*Weighted Graph*)

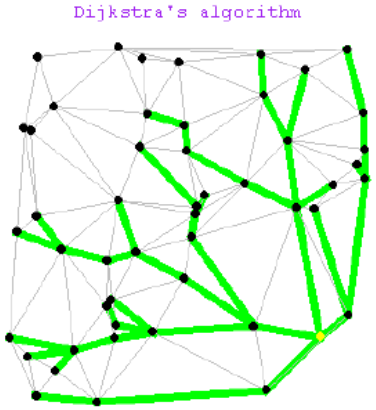
Graf berbobot adalah graf yang setiap sisinya memiliki sebuah nilai atau bobot tertentu. Bobot pada graf dapat digunakan untuk menunjukkan waktu, jarak, ongkos produksi, dan sebagainya tegrantung persoalan yang ingin direpresentasikan dalam sebuah graf.



Gambar 7. Graf Berbobot
(Sumber : [1])

C. Algoritma Dijkstra [3]

Salah satu algoritma pencarian rute atau lintasan terpendek dari sebuah graf adalah algoritma Dijkstra.



Gambar 8. Dijkstra Algorithm
(Sumber: [3])

1. Sejarah singkat Algoritma Dijkstra

Algoritma Dijkstra adalah sebuah algoritma yang digunakan untuk menyelesaikan masalah *shortest path*. Algoritma ini ditemukan oleh Edsger Wybe Dijkstra (1930-2002). Ia adalah seorang *computer scientist* yang berasal dari negeri Belanda. Pada tahun 1972 beliau memeroleh penghargaan A.M. Turing Award, sebuah penghargaan yang bergengsi di bidang *computer science*. Selain itu, banyak *essay* yang beliau terbitkan mengenai programming.

2. Penyelesaian persoalan dengan Algoritma Dijkstra

Salah satu persoalan yang sering terjadi adalah menemukan lintasan terpendek dari sebuah simpul V dengan simpul-simpul lainnya pada sebuah graf. Akan tetapi, tidak perlu khawatir karena hal tersebut dapat diselesaikan menggunakan algoritma Dijkstra. Pada naskah asli dari Edsger, algoritma ini diterapkan untuk mencari jarak terpendek pada sebuah graf berarah, namun juga benar untuk sebuah graf tidak berarah.

Beberapa syarat yang harus terpenuhi untuk dapat menentukan lintasan terpendek pada suatu graf dengan menggunakan algoritma dijkstra adalah [3]

- Graf berarah, namun tetap benar untuk graf yang tidak berarah.
- Graf harus terhubung.

- Seluruh sisi harus memiliki bobot yang tidak negatif. Jika bobot negatif maka akan menghasilkan solusi yang tidak berhingga banyaknya.

Secara umum, berikut adalah langkah-langkah kerja yang dilakukan menggunakan algoritma Dijkstra: [4]

a. Set node awal dengan besar 0 dan node lain dengan besar tak hingga
b. Set node yang belum pernah dilalui dan node awal di set sebagai node keberangkatan.
c. Dari node keberangkatan, pertimbangkan node tetangga yang belum pernah dilalui dan hitung jaraknya dari titik keberangkatan. Apabila jaraknya lebih kecil dari jarak sebelumnya, hapus data lama dan digantikan dengan data yang baru.
d. Ketika node tetangga sudah dikunjungi dan diproses, maka node tersebut diberi tanda sebagai node yang sudah pernah dikunjungi. Node yang sudah dikunjungi tidak perlu diproses kembali.
e. Set node yang belum dilewati dengan jarak terkecil sebagai node keberangkatan berikutnya, kemudia ulangi langkah (c).

Berikut pseudocode yang dapat digunakan untuk lebih memahami algoritma Dijkstra.

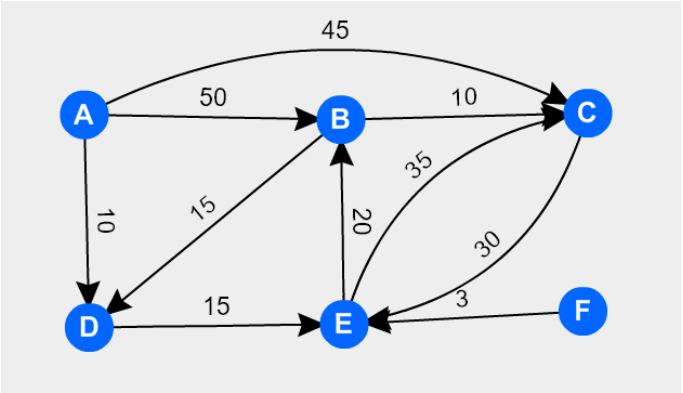
```
dist[s] ←0                              (distance to source vertex is zero)
for all v ∈ V-{s}
    do dist[v] ←∞                       (set all other distances to infinity)
S←∅                                     (S, the set of visited vertices is initially empty)
Q←V                                     (Q, the queue initially contains all vertices)
while Q ≠∅                              (while the queue is not empty)
do  u ← mindistance(Q,dist)             (select the element of Q with the min. distance
   S←S∪{u}                              (add u to list of visited vertices)
   for all v ∈ neighbors[u]
      do if  dist[v] > dist[u] + w(u, v)        (if new shortest path found)
           then    d[v] ←d[u] + w(u, v)        (set new value of shortest path)
                                                (if desired, add traceback code)
return dist
```

Gambar 9. Pseudocode algoritma dijkstra
(Sumber : [3])

III. PEMBAHASAN DAN ANALISIS ALGORITMA DIJKSTRA DALAM MENENTUKAN JARAK TERPENDEK

1. Analisis Algoritma Dijkstra Menentukan Jarak Terpendek

Pada bagian kali ini, penulis akan memberikan contoh dan menjelaskan bagaimana algoritma dijkstra diterapkan dalam menentukan jarak terpendek antara dua buah simpul yang diinginkan. Untuk itu penulis menggunakan salah satu kasus sebagai berikut.



Gambar 10. Contoh Permasalahan Graf G
(Sumber : buatan penulis)

Misalkan diberikan sebuah persoalan untuk mencari jarak terpendek dari simpul A ke semua simpul lainnya pada graf G . Maka dengan algorima dijkstra, kita akan memeroleh hasil sebagai berikut (dalam bentuk tabel).

Simpul yang dipilih	B	C	D	E	F
D	50	45	10	∞	∞
E	50	45	10	25	∞
B	45	45	10	25	∞
C	45	45	10	25	∞
F	45	45	10	25	∞

Tabel 1. Tabel hasil perhitungan menggunakan algoritma dijkstra
(Sumber : buatan penulis)
Warna kuning = Simpul yang sudah dikunjungi

Karena kita ingin mencari jarak terpendek dari simpul A ke simpul C, maka kita mengeset simpul keberangkatan awal kita dari simpul A dengan nilai 0 (nol). Selanjutnya dari simpul A, tinjau simpul yang bertetangga dengan simpul A lalu berikan nilai masing-masing simpul. Dalam hal ini, bobot dari simpul A ke B sebanyak 50, A ke C sebanyak 45, A ke D sebanyak 10. Sedangkan karena tidak ada sisi yang menghubungkan A dengan E dan F maka diberi nilai tak-hingga. Seluruh nilai kita tuliskan pada baris pertama Tabel 1.

Sekarang kita meninjau baris pertama dan mencari simpul yang belum dikunjungi dengan bobot paling kecil. Kita mendapati simpul D dengan bobot 10. Pilih simpul D dan jadikan simpul keberangkatan, setelah itu diberikan warna kuning sebagai penanda.

Selanjutnya kita mengulangi proses yang sama, dari simpul D kita mencari simpul yang bertetangga dan terhubung langsung dari simpul D. Kita mendapati simpul E. Karena 10 +15 < ∞ (nilai simpul D + bobot sisi (D,E) < nilai E) maka nilai E diganti dan disimpan menjadi 25 pada baris kedua Tabel 1. Kembali lakukan pemilihan simpul yang belum dikunjungi dengan bobot paling kecil, maka didapati simpul E.

Simpul E dijadikan simpul keberangkatan, kemudian kita kembali mencari simpul tetangga yang belum pernah dikunjungi dan terhubung langsung dari simpul E, kita mendapati simpul B dan simpul C. Pada simpul B, karena 25 + 20 < 50 (nilai simpul E + bobot sisi (E,B) < nilai simpul B) maka nilai simpul B

diganti menjadi 45. Akan tetapi, pada simpul C dikarenakan 25 + 35 > 45 (nilai simpul E + bobot sisi (E,C) > nilai simpul C) maka nilai simpul C tetap dan tidak berubah. Di baris ketiga (Tabel 1) kembali dicek simpul yang belum pernah dikunjungi dengan bobot minimum, maka kita mendapatkan simpul B dan C.

Penulis memilih simpul B (boleh juga memilih C karena memiliki bobot yang sama besar) kemudian kembali melakukan pengecekan simpul yang belum pernah dikunjungi dan dapat dicapai dari simpul B sehingga didapati simpul C. Karena 45 + 10 > 45 maka nilai simpul C tidak diganti nilainya.

Selanjutnya, kita akan memilih simpul C pada baris yang keempat (Tabel 1) karena belum pernah dikunjungi dan memiliki bobot mnimum. Kembali dilakukan pengecekan dan kita mendapatkan simpul E dapat dicapai dari simpul C. Akan tetapi, dikarenakan simpul E sudah pernah dikunjungi maka tidak lagi diproses.

Pada akhirnya kita mengunjungi simpul terakhir yaitu simpul F dengan bobot minimum dan belum pernah dikunjungi. Dikarenakan tidak terdapat simpul lain yang belum dikunjungi dan bisa dicapai dari simpul F, maka seluruh proses algoritma dijkstra selesai.

Dari tabel tersebut dapat kita simpulkan, jarak terpendek dari simpul A ke masing-masing simpul adalah

A ke B = 45 (A,D,E,B)
A ke C = 45 (A,C)
A ke D = 10 (A,D)
A ke E = 25 (A,D,E)
A ke F = ∞ (tidak ada rute dari A ke F)

Kita dapat melihat bahwa dari simpul A ke simpul lain tidak hanya dapat dicapai dengan satu buah jalan atau lintasan saja, tetapi beberapa lintasan. Sebagai contoh dari A ke B dapat melalui (A,B) , (A,D,E,B) , ataupun (A,C,E,B). Semua benar, akan tetapi algoritma dijkstra menghasilkan hasil paling optimal yaitu jarak terpendek sebesar 45 melalui lintasan (A,D,E,B).

2. Menentukan Jarak Tempuh Perjalanan Terpendek di Kota Bandung

Seperti yang dijelaskan pada bagian II, kota Bandung adalah kota yang memilki bentuk permukaan wilayah yang cukup unik dengan cekungan dan pusatnya di tengah kota. Di Bandung banyak sekali tempat yang bisa dicapai dari berbagai jalan yang berbeda. Selain itu, dikarenakan banyak jalan yang satu arah (*one way*) di kota Bandung maka rute untuk perjalanan pulang dan pergi dapat berbeda.

Pada bagian kali ini, penulis akan menggunakan algoritma dijkstra untuk menentukan rute perjalanan terpendek di kota Bandung.

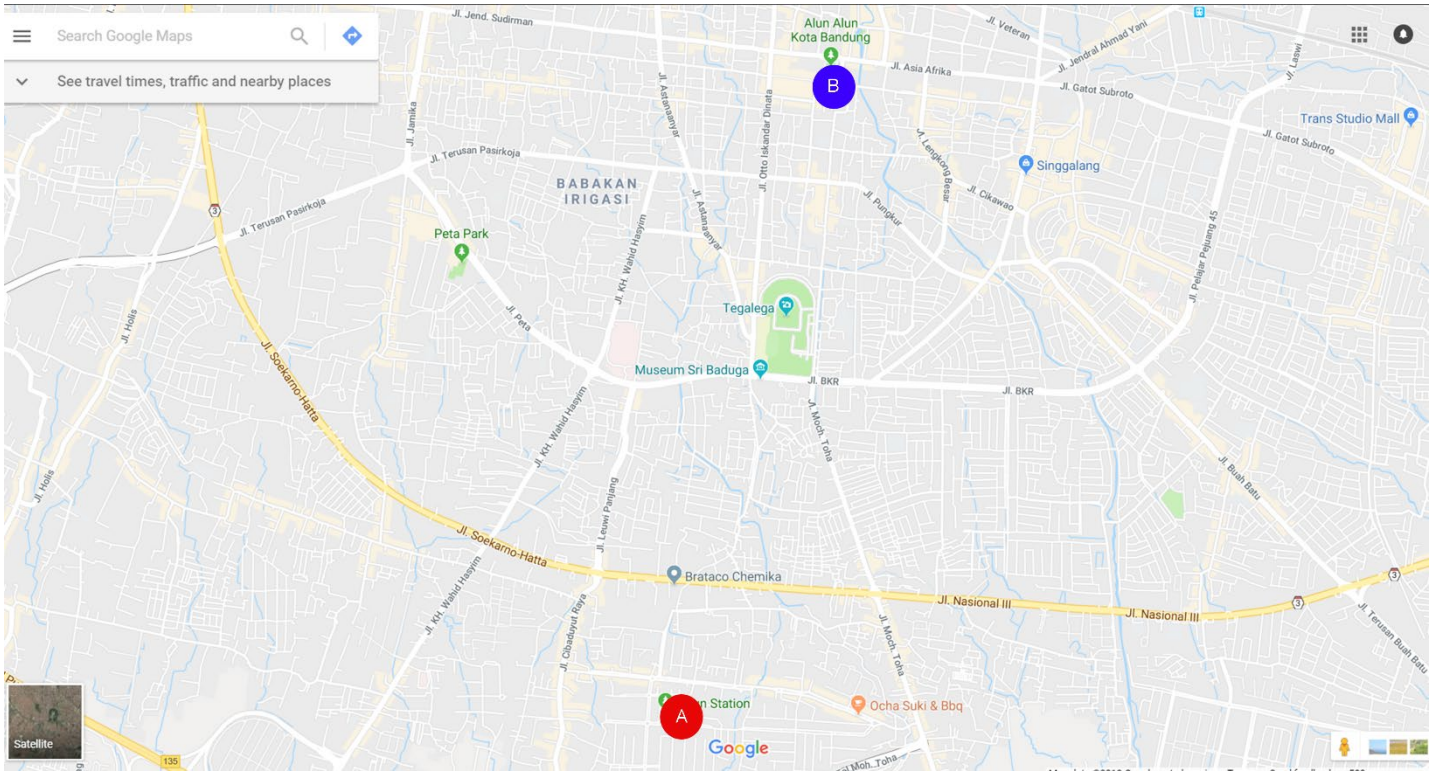
2.1 Batasan Penelitian

Dikarenakan Bandung kota yang cukup luas, maka dari itu penulis akan menggambarkan peta Bandung dengan simpul simpul di beberapa titik titik tertentu atau jalan-jalan besar di kota Bandung. Penelitian ini dimulai dengan titik keberangkatan awal dari rumah penulis dikarenakan rumah penulis kebetulan terletak tepat di perbatasan ujung selatan kota Bandung.

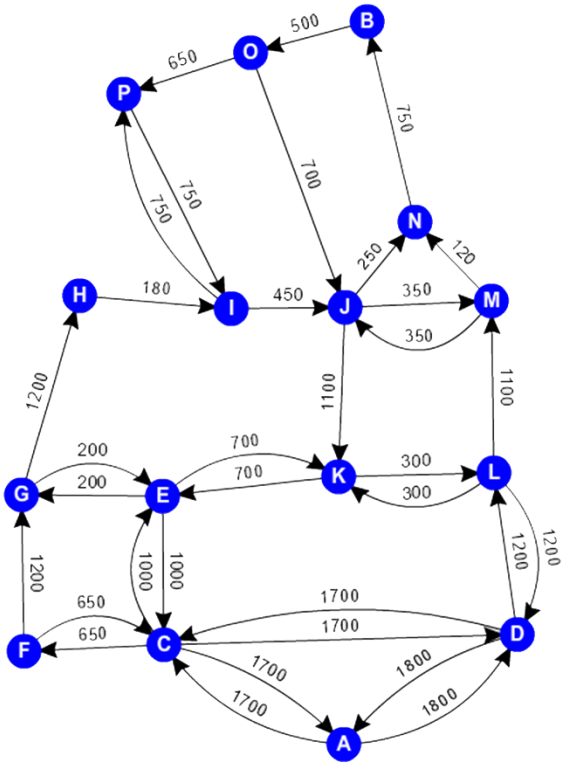
2.2 Hasil Penelitian

Misalkan kita akan mencari jarak tempuh terdekat dari rumah penulis ke alun-alun kota Bandung. Gambaran umum daerah yang akan dilalui oleh penulis dapat dilihat pada gambar 11.

Gambar 11. Titik A adalah rumah penulis, Titik B adalah alun-alun kota Bandung (Sumber: maps.google.com)



Untuk memudahkan, peta pada gambar 11 akan disederhanakan dan dibuat dalam representasi graf berarah.



Gambar 12. Representasi graf berarah pada peta gambar 11
(Sumber : buatan penulis)

Keterangan:
A = Rumah penulis
B = Alun-alun kota Bandung
C = Perempatan Cibaduyut (patung sepatu)
D = Perempatan Moh. Toha – Soekarno-Hatta
E = Pertigaan Leuwi Panjang
F = Perempatan Kopo – Soekarno-Hatta
G = Perempatan Peta – Kopo
H = Pertigaan Pasir Koja – Kopo
I = Pertigaan Pasir Koja – Astana Anyar
J = Perempatan Pungkur – Otista
K = Perempatan Peta – Inhoftank
L = Perempatan BKR – Moh. Toha
M = Pertigaan Moh. Toha – Pungkur
N = Jalan Dewi Sartika
O = Perempatan Asia Afrika – Otista
P = Perempatan Sudirman – Astana Anyar
*Bobot yang dituliskan pada graf menunjukkan jarak antara dua buah tempat dalam besaran meter (m)

Misalkan kita berada di simpul A, yaitu rumah penulis dan ingin menempuh perjalanan ke alun-alun kota Bandung, kita mengeset node A sebagai titik awal keberangkatan.

Dengan menggunakan algoritma dijkstra maka diperoleh hasil sebagai berikut:

(Tabel dibagi menjadi 3 tingkat dikarenakan space yang sempit)

Sisi yang dipilih	B	C	D	E	F
C	∞	1700	1800	∞	∞
D	∞	1700	1800	2700	2350
F	∞	1700	1800	2700	2350
E	∞	1700	1800	2700	2350
G	∞	1700	1800	2700	2350
L	∞	1700	1800	2700	2350
K	∞	1700	1800	2700	2350
M	∞	1700	1800	2700	2350
H	∞	1700	1800	2700	2350
N	∞	1700	1800	2700	2350
I	4970	1700	1800	2700	2350
J	4970	1700	1800	2700	2350
B	4970	1700	1800	2700	2350
P	4970	1700	1800	2700	2350
O	4970	1700	1800	2700	2350
Sisi yang dipilih	G	H	I	J	K
C	∞	∞	∞	∞	∞
D	∞	∞	∞	∞	∞
F	∞	∞	∞	∞	∞
E	3550	∞	∞	∞	∞
G	2900	∞	∞	∞	3400
L	2900	4100	∞	∞	3400
K	2900	4100	∞	∞	3200
M	2900	4100	∞	∞	3200

H	2900	4100	∞	4450	3200
N	2900	4100	4280	4450	3200
I	2900	4100	4280	4450	3200
J	2900	4100	4280	4450	3200
B	2900	4100	4280	4450	3200
P	2900	4100	4280	4450	3200
O	2900	4100	4280	4450	3200
Sisi yang dipilih	L	M	N	O	P
C	∞	∞	∞	∞	∞
D	∞	∞	∞	∞	∞
F	3000	∞	∞	∞	∞
E	3000	∞	∞	∞	∞
G	3000	∞	∞	∞	∞
L	3000	∞	∞	∞	∞
K	3000	4100	∞	∞	∞
M	3000	4100	∞	∞	∞
H	3000	4100	4220	∞	∞
N	3000	4100	4220	∞	∞
I	3000	4100	4220	∞	∞
J	3000	4100	4220	∞	5030
B	3000	4100	4220	∞	5030
P	3000	4100	4220	5470	5030
O	3000	4100	4220	5470	5030

Tabel 2. Tabel Hasil perhitungan algoritma dijkstra pada gambar 12 (sumber : buatan penulis)

Berdasarkan hasil yang sudah didapat dari penelitian tersebut, kita dapat mengetahui jarak terpendek dari rumah penulis ke alun-alun kota Bandung adalah 4970 m atau kurang lebih 4.97 km. Adapun jalan yang dilalui penulis adalah perempatan Moh. Toha – Soekarno-Hatta, kemudian melewati perempatan BKR – Moh. Toha, pertigaan Moh. Toha – Pungkur, jalan Dewi Sartika, lalu tiba di alun-alun kota Bandung. (A,D,L,M,N,B).

Hasil yang didapat dari penelitian di atas hanya memperhitungkan jarak antara satu titik ke titik lainnya. Oleh karena itu hanya bisa menentukan rute atau jalan dengan jarak terpendek, bukan rute tercepat atau terefektif. Algoritma ini masih bisa dikembangkan dan disempurnakan dengan memerhitungkan kondisi keadaan di jalan. Misal jalan yang macet, terjadi kecelakaan, jalan yang besar atau kecil, sehingga jalan terpendek yang didapat juga bisa menjadi jalan yang tercepat dan terefektif bagi pengendara atau pejalan kaki.

IV. KESIMPULAN

Graf memiliki banyak aplikasi dan kegunaan untuk memecahkan berbagai persoalan. Dengan menggunakan salah satu algoritma pencarian lintasan terpendek, yaitu algoritma Dijkstra, kita dapat menemukan *shortest path* untuk perjalanan kita ketika beraktivitas sehari-hari. Terkhususnya bagi kita baik mahasiswa maupun masyarakat yang tinggal di wilayah perkotaan yang luas dan membutuhkan tingkat mobilitas yang tinggi seperti kota Bandung.

V. UCAPAN TERIMA KASIH

Pertama-tama, penulis mengucapkan puji syukur kepada Tuhan yang Maha Esa karena kasih-Nya telah memberikan kesempatan untuk menyelesaikan makalah ini. Penulis juga mengucapkan terima kasih kepada orangtua, teman-teman penulis yang memberikan bantuan doa dan inspirasi untuk pembuatan makalah ini. Penulis juga mengucapkan 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 yang bersama-sama mengajarkan mahasiswa/i untuk dapat memahami tentang matematika diskrit. Tidak lupa, penulis juga mengucapkan banyak terima kasih kepada seluruh sumber informasi, referensi yang membantu penulis dalam menyelesaikan makalah ini. Semoga Tuhan membalas semua kebaikan yang sudah diberikan oleh seluruh pihak.

REFERENCES

[1] Munir, Rinaldi. Matematika Diskrit Edisi 3. Bandung : Penerbit Informatika, 2010. hlm 356-376, 412-421.

[2] PPID Kota Bandung. Profil Kota Bandung. [Internet]. https://ppid.bandung.go.id/profil-kota-bandung/ diakses 7 Desember 2018 pukul 15.03

[3] Dijkstra Algorithm. [Internet]. http://math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Melissa.pdf diakses 8 Desember 2018 pukul 22.16

[4] Algoritma Dijkstra. Binus University. [Internet]. http://mti.binus.ac.id/2017/11/28/algoritma-dijkstra/ diakses 8 Desember 2018 pukul 22.37

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, 9 Desember 2018

Lukas Kurnia Jonathan
13517006