Penerapan Algoritma Dijkstra untuk Menentukan Rute Perjalanan Optimum di Kota Bandung

Lukas Kurnia Jonathan/13517006
Program Studi Teknik Informatika
Sekolah Teknik Elektro Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
lukasjonathan99@gmail.com

***Abstrak*—Sebuah kota tentunya terdiri dari berbagai sarana dan prasarana, tempat wisata, area perumahan, pusat perkantoran dan pusat perbelanjaan. Dalam melakukan aktivitas sehari-hari di sebuah kota, tentunya tidak pernah terlepas dari transportasi darat dan perpindahan dari suatu tempat ke tempat lain. Seringkali perjalanan dari suatu tempat ke tempat lainnya membutuhkan *cost* baik waktu, jarak yang cukup besar. Oleh karena itu, dibutuhkan suatu solusi agar rute perjalanan yang dipilih oleh seseorang dapat menghasilkan rute dengan *cost* optimum. Makalah ini akan membahas salah satu penggunaan Algoritma Greedy Dijkstra sebagai strategi untuk mengoptimasikan rute perjalanan di Kota Bandung.**

Kata kunci*— *Greedy, Dijkstra, Optimum, Perjalanan, Bandung

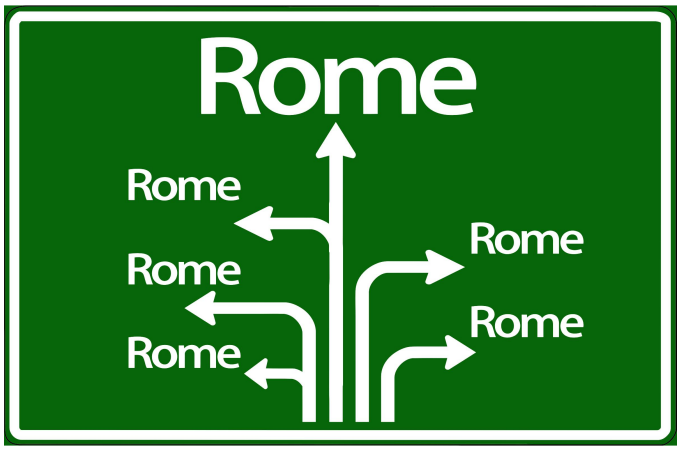
I. Pendahuluan

Manusia dalam menjalankan segala aktivitasnya tentu saja meemerlukan mobilisasi untuk berpindah dari suatu tempat ke tempat lain. Di sebuah kota yang sudah berkembang dengan banyak penduduk serta area permukiman yang banyak, tentunya tingkat mobilisasi akan semakin tinggi. Hal ini didukung dengan fasilitas yang terdapat dalam sebuah kota itu sendiri, seperti pusat perbelanjaan, pusat perkantoran, tempat wisata, pusat pendidikan dan sebagainya. Untuk dapat melakukan mobilisasi dari rumah atau tempat tinggal seseorang mencapai salah satu tempat yang diinginkan, diperlukan perjalanan, baik menggunakan kendaraan bermotor maupun berjalan kaki.

Masalah yang dapat muncul dan sering terjadi adalah menentukan jalan mana yang harus dipilih untuk bisa mencapai tujuan yang diinginkan. Hal ini dikarenakan banyak sekali jalan yang dapat dilalui dan ditempuh untuk mencapai tujuan yang kita inginkan. Sebagai contoh, kita dapat melihat jalan di kota Bandung. Untuk mencapai alun-alun dari kampus ITB, kita dapat menggunakan berbagai alternatif cara. Perbedaan pemilihan jalan untuk menempuh suatu perjalanan tertentu dapat menimbulkan berbagai dampak bagi pengguna jalan. Beberapa diantaranya adalah waktu tempuh perjalanan yang terlalu lama, jarak perjalanan yang terlalu jauh untuk mencapai tujuan, dan biaya perjalanan yang relatif mahal untuk mencapai tujuan yang diinginkan.

Melihat hal tersebut, penulis merasa tertarik untuk menyelesaikan persoalan pencarian rute perjalanan optimum untuk menempuh perjalanan di kota Bandung.

Salah satu cara untuk mendapatkan rute perjalanan yang optimum dengan *cost* yang minimum adalah menggunakan penerapan strategi algoritma, yaitu Algoritma Greedy. Salah satu algoritma Greedy yang memiliki solusi optimum dalam pencarian rute perjalanan optimum adalah Algoritma Dijkstra. Optimasi akan memudahkan perjalanan pengguna jalan untuk bisa melalui perjalanan dari satu tempat ke tempat lainnya dengan biaya (*cost*) yang minimum.



Gambar 1. All-Roads-Lead-To-Rome. Sumber: https://heidelblog.net/2017/03/relevance-leads-back-to-rome/

II. Dasar Teori

A. Kota Bandung

Indonesia adalah negara kepulauan yang terdiri dari banyak sekali pulau. Indonesia memiliki 34 provinsi dengan beragam budaya yang tersebar di berbagai kota di seluruh Indonesia. Salah satu kota dengan budayanya yang merupakan tempat tinggal penulis adalah kota Bandung, terletak di provinsi Jawa Barat. Bandung merupakan kota dengan bentuk permukaan wilayah yang relatif cukup unik. Jika dilihat dari atas, Bandung menyerupai cekungan atau mangkuk yang besar diantara gunung-gunung tinggi.

Secara geografis, Bandung terletak di tengah Jawa Barat dengan ketinggian kurang lebih 768 m di atas permukaan laut. Kota Bandung sendiri memiliki luas wilayah sebesar 16.713 hektar yang terbagi atas 30 kecamatan, 151 kelurahan, 1.561

RW, dan 9.691 RT. Kecamatan yang terluas di kota Bandung terletak di daerah selatan Bandung, yaitu dengan luas 89 hektar.

Secara demografis dan pemerintahan, pada tahun 2012 penduduk yang tercatat di kota Bandung kurang lebih sebanyak 2.650.000 jiwa, terdiri dari sekitar 1.350.000 orang laki-laki dan 1.300.000 orang perempuan. Bandung dipimpin oleh walikota beserta wakilnya, dibantu 3 sekretaris daerah, 17 kepala dinas, 6 kepala badan, 8 kepala bagian, 1 kepala kantor, 4 perusahaan daerah, 1 inspektorat, dan 1 kepala satuan polisi pamong praja.

Jalanan yang terdapat di kota Bandung memiliki banyak macamnya, mulai dari jalan kecil untuk jalan pintas motor, jalan yang besar untuk *bypass,* jalan tol yang tidak bisa dilalui oleh kendaraan roda dua, jalan satu arah maupun dua arah. Tentunya akan banyak sekali variasi rute jalan yang dapat dipilih oleh pengguna jalan. Oleh karena itu, pada makalah ini penulis memutuskan untuk mengangkat topik mengenai kota Bandung, Jawa Barat.

Gambar 2. Kota Bandung.
Sumber: https://nasional.tempo.co/read/1108303/ketika-kota-bandung-menjadi-lebih-baik

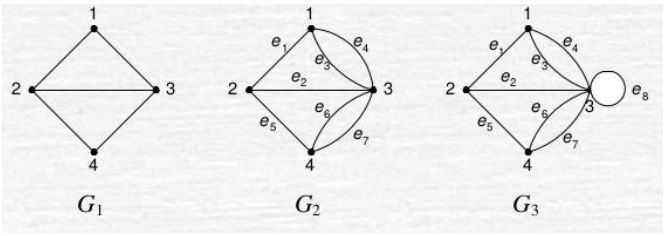
B. Teori Graf

Graf merupakan salah satu pokok bahasan yang sudah cukup tua namun memiliki banyak penerapan sekaligus manfaat. Salah satunya, dengan graf kita dapat merepresentasikan hubungan antara objek-objek diskrit. Graf memiliki dua buah atribut utama yaitu simpul (*vertex*) dan sisi (*edge)*. Sebagai contoh, sebuah negara dapat direpresentasikan dengan graf. Simpul menunjukkan suatu negara dan sisi menunjukkan *path* yang menghubungkan negara tersebut dengan negara lain.

Suatu graf G memiliki himpunan tidak kosong dari berbagai simpul V dan himpunan sisi E yang menghubungkan suatu simpul ke simpul lainnya. Berikut adalah notasi graf:

$$G = (V,E)$$

Berikut adalah beberapa contoh graf:



Gambar 3. beberapa contoh graf. Sumber: [1]

G1 adalah graf sederhana dengan V = {1,2,3,4} dan E = { (1,2), (2, 3), (1, 3), (1, 3), (2, 4), (3, 4), (3, 4), (3, 3) }. Graf juga dapat memiliki sisi ganda seperti graf G2 dimana sisi e3 (1,3) dan e4 (1,3) menghubungi dua buah simpul yang sama. Graf juga dapat memiliki sisi yang berhubungan dengan simpul itu sendiri membetuk sebuah *loop* pada graf G3.

1) Jenis-jenis Graf:

a) Graf Sederhana (simple graph)

Graf sederhana adalah graf yang tidak memiliki loop atau sisi ganda. G1 pada gambar 3 adalah salah satu contoh graf sederhana.

b) Graf tak-sederhana (unsimple graph)

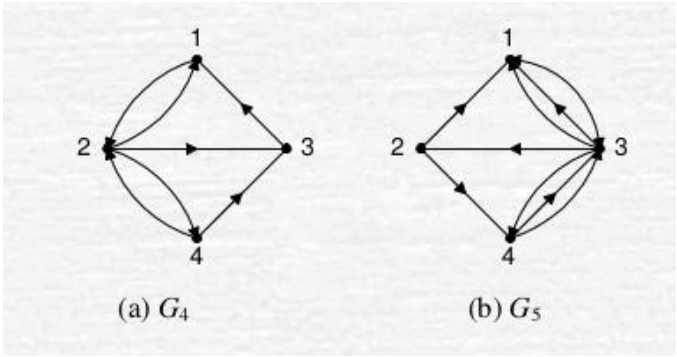
Berkebalikan dengan graf sederhana, graf tak-sederhana adalah graf yang memiliki loop atau sisi ganda pada sisi di graf tersebut. G2 dan G3 pada gambar 3 adalah contoh graf sederhana

c) Graf tak-berarah (undirected graph)

Graf tak-berarah adalah graf dengan sisi yang tidak mempunyai orientasi. Pada gambar 3, semua graf adalah graf tak-berarah

d) Graf berarah (directed graph)

Graf berarah adalah graf yang setiap sisinya memeiliki arah orientasi. Gambar 4 adalah contoh graf berarah



Gambar 4. Graf berarah. Sumber: [1]

2) Terminologi Graf:

a) Ketetanggan (Adjacent)

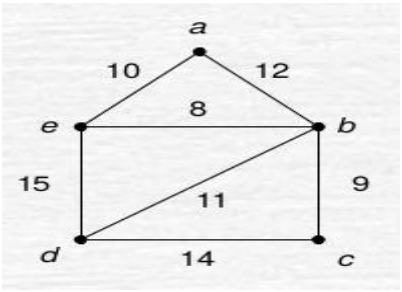
Pada sebuah graf, kedua buah simpul dapat dikatakan bertetanggaan jika keduanya terhubung secara langsung.

b) Bersisian (Incidency)

Pada sebuah graf, untuk sembarang sisi e = (vi , vj) dikatakan bersisian jika e bersisian dengan simpul vi dan e bersisian dengan simpul vj.

c) Graf berbobot (Weighted graph)

Graf berbobot adalah suatu graf yang setiap sisinya diberikan bobot tertentu. Gambar 5 adalah contoh graf berbobot.



Gambar 5. Graf berbobot. Sumber : [1]

Dalam merepresentasikan hubungan antara suatu simpul dengan simpul lain dapat dilakukan dengan merepresentasikan dalam sebuah matriks. Secara umum, terdapat dua buah matriks untuk merepresentasikan graf:

1) Matriks ketetangaan (Adjacency Matrices)

Matriks ketetanggaan yang dapat digambarkan jika kedua simpul memiliki hubungan ketertanggaan. Contoh gambar 6 berdasarkan gambar 5

	a	b	c	d	e
a	0	12	0	0	10
b	12	0	9	11	8
c	0	9	0	14	0
d	0	11	14	0	15
e	10	8	0	15	0

Gambar 6. Adjacency matrices dengan bobot. Sumber: buatan penulis

2) Matriks bersisian (Incidency Matrices)

Matriks bersisian merepresentasikan sisi yang berhubungan dengan simpul pada graf. Contoh gambar 7 berdasarkan graf G2 gambar 3

	e1	e2	e3	e4	e5	e6	e7
1	1	0	1	1	0	0	0
2	1	1	0	0	1	0	0
3	0	1	1	1	0	1	1
4	0	0	0	0	1	1	1

Gambar 7. Incidency Matrices. Sumber : buatan penulis

Pada makalah ini, graf yang digunakan adalah graf berbobot dengan simpul menunjukkan persimpangan dari suatu jalan. Sedangkan representasi yang digunakan dalam eksperimen menggunakan matriks berketetanggan (adjacency matrices) untuk menggambarkan keterhubungan antara suatu simpul dengan simpul lain.

C. Algoritma Greedy

Algoritma Greedy merupakan strategi algoritma yang digunakan untuk memecahkan persoalan optimasi. Persoalan optimasi dapat dibagi menjadi dua macam, yaitu Maksimasi (maximization) dan Minimisasi (minimization). Pada strategi greedy, solusi yang dibuat dilakukan langah per langkah dengan setiap langkah mengambil keputusan terbaik yang dapat diambil (optimum lokal) dengan harapan di akhir langkah, dapat mengarah ke solusi optimum global.

Elemen-Elemen pada Algoritma Greedy:

1) Himpunan Kandidat, C

Himpunan kandidat adalah himpunan yang berisi elemen-elemen yang dapat membentuk solusi.

2) Himpunan Solusi, S

Himpunan solusi adalah himpunan yang sesuai dengan solusi dari persoalan yang diinginkan. Himpunan solusi merupakan subset dari himpunan kandidat

3) Fungsi Seleksi

Fungsi seleksi adalah suatu fungsi yang digunakan untuk menyeleksi kandidat yang menghasilkan solusi optimum. Pada setiap langkah greedy, fungsi seleksi akan selalu digunakan. Hasil dari langkah sebelumnya tidak akan dipertimbangkan lagi untuk pencarian hasil optimum di langkah selanjutnya sesuai dengan prinsip greedy. Output yang diharapkan dari fungsi seleksi pada umumnya mencari nilai maksimum atau minimum.

4) Fungsi Kelayakan

Fungsi kelayakan adalah fungsi yang digunakan untuk memeriksa kelayakan dari suatu kandidat yang dipilih. Syarat kelayakan dapat didefinisikan sesuai persoalan. Tugas dari fungsi kelayakan adalah memastikan tidak ada kandidat yang melanggar batasan (*constraint*) yang ada. Jika ada kandidat yang tidak layak maka kandidat tersebut tidak akan dipertimbangkan lagi.

5) Fungsi Objektif

Tujuan atau sasaran yang ingin didapatkan. Misalkan meminimumkan jumlah koin, memaksimalkan keuntungan.

Pada Algoritma Greedy, solusi optimum global belum tentu solusi optimum(terbaik) yang dapat didapat. Kita dapat melakukan perbandingan dengan solusi optimum dari Algoritma Brute Force (tidak dibahas di makalah ini). Oleh karena itu, solusi pada Algoritma Greedy dapat disebut sub-optimum atau pseudo-optimum.

Hal ini terjadi karena Algoritma Greedy tidak beroperasi secara menyeluruh untuk semua alternatif solusi. Selain itu, fungsi seleksi yang digunakan dapat bervariasi, sehingga perlu dilakukan pemilihan fungsi yang paling tepat agar bisa menghasilkan hasil optimal.

Skema umum Algoritma Greedy[1]:

```
Function greedy(input C: himpunan_kandidat)->
himpunan_kandidat


Deklarasi
```

```
  X : kandidat
  S: himpunan_kandidat
Algoritma
  S <-- {}  {Inisialisasi dengan kosong}
  while (not SOLUSI(S)) and (C != {} ) do
    x <-- SELEKSI(C){ pilih sebuah kandidat dari C}
    C <-- C -{x}{ elemen himpunan kandidat berkurang satu }
      if LAYAK(S  U {x}) then
          S<--S U {x}
      endif
   endwhile
   {SOLUSI(S) or C = {} }
   if SOLUSI(S) then
        return S
    else
        write('tidak ada solusi')
    endif
```

D. Algoritma Dijkstra

Algoritma Dijkstra merupakan salah satu algoritma pencarian rute terpendek yang optimal.

1) Sejarah Algoritma Dijkstra

Algoritma Dijkstra ditemukan oleh Edsger W. Dijkstra (1930 - 2002). Beliau merupakan salah satu orang yang berpengaruh di bidang computer science. Beberapa kontribusi scientific yang diberikan oleh beliau antara lain: desain algoritma, bahasa pemrograman, desain program, sistem operasi, dan proses distribusi. Beliau banyak mendapatkan hadiah dan penghargaan untuk kontribusi yang ia kerjakan termasuk penghargaan tertinggi pada tahun 1972 di bidang computer science yaitu ACM Turing Award.

2) Algoritma Dijkstra

Algoritma ini digunakan untuk mendapatkan jarak terpendek dari suatu graf berarah (juga benar untuk graf tidak berarah). Salah satu representasi dari grafik yang digunakan untuk Algoritma Dijkstra adalah menggunakan graf berarah dengan simpul V dan sisi E.

Langkah-langkah yang dilakukan pada Algoritma Dijkstra dalam mencari jalur terpendek antara dua buah simpul adalah sebagai berikut:

1. Menandai semua simpul yang ada menjadi belum dikunjungi

2. Menentukan titik yang menjadi titik awal dan mengeset titik awal menjadi 0, sedangkan titik yang lain tak hingga.

3. Dari semua simpul yang ada dan belum dikunjungi, cari nilai bobot yang paling kecil. Dari simpul tersebut, cari seluruh tetangga yang terhubung dan belum dikunjungi, kemudan bandingkan bobot pada simpul tetangga dengan jarak dari simpul dipilih ke simpul tetangga ditambah bobot pada simpul dipilih.

4. Jika hasil penjumlahan lebih kecil, maka bobot pada simpul tetangga diupdate dengan hasil perbandingan yang lebih kecil

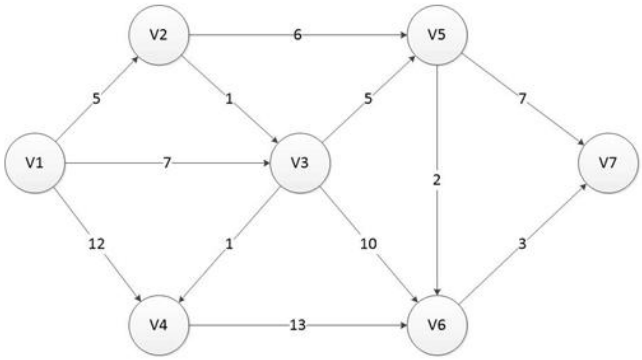
5. Tandai simpul yang dipilih dengan "dikunjungi"

6. Lakukan langkah ke-3 hingga seluruh simpul sudah dikunjungi

Pseudo-code Algoritma Dijkstra

```
procedure Dijkstra (input G: weighted_graph, input a: intial_vertex)
Deklarasi:
   S : himpunan simpul solusi
   L : array[1..n] of real { L(z) berisi panjang lintasan terpendek dari a ke z}
Algoritma
   for i = 1 to n
      L(vi = tak hingga)
   end for
   L(a)= 0; S = { }
   while z not in S do
   U = simpul yang bukan di dalam S dan memiliki L(u) minimum
   S = S U {u}
   for semua simpul v yang tidak terdapat di dalam S
      if L(u) + G(u,v) < L(v) then L(v)    L(u) + G(u,v)
   end for
   end while
```

Contoh persoalan Dijkstra:



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

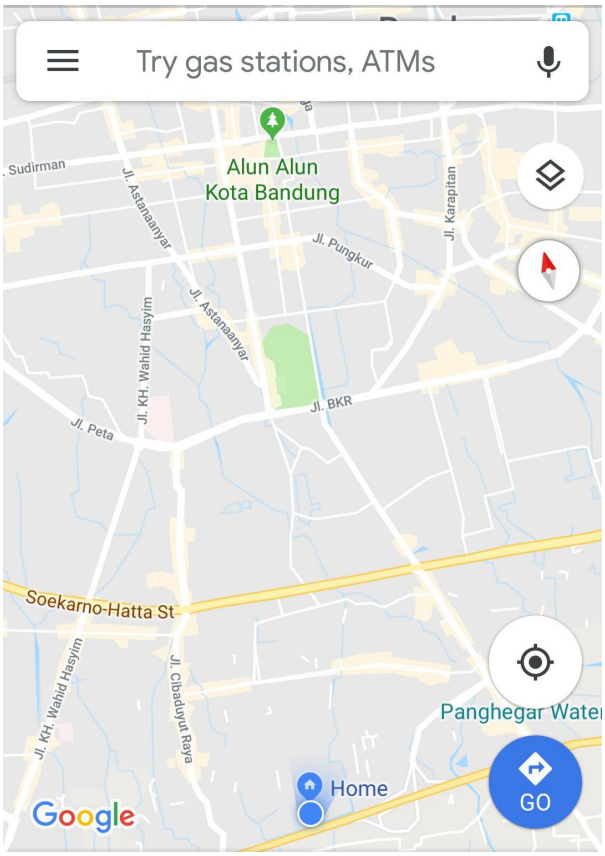
Gambar 8. Sekilas contoh Algoritma Dijkstra. Sumber: http://mti.binus.ac.id/2017/11/28/algoritma-dijkstra/

III. Batasan Masalah

Dalam penelitian kali ini, penulis membatasi masalah yang akan dibahas, diselesaikan, dan diuji. Batasan ini diperlukan agar penyelesaian masalah dapat dilakukan dengan lebih mudah dan lebih singkat. Ke depannya, batasan masalah ini dapat lebih ditingkatkan dalam penelitian di masa yang akan datang. Beberapa batasan masalah yang diambil oleh penulis antara lain:

A. Daerah Titik Sampel

Karena pada praktiknya secara nyata kota Bandung adalah kota yang cukup besar seperti dijelaskan pada teori dasar, maka penulis mengambil sampel untuk melakukan pengujian dari rumah penulis menuju landmark dari kota Bandung, yaitu Alun-alun Bandung.

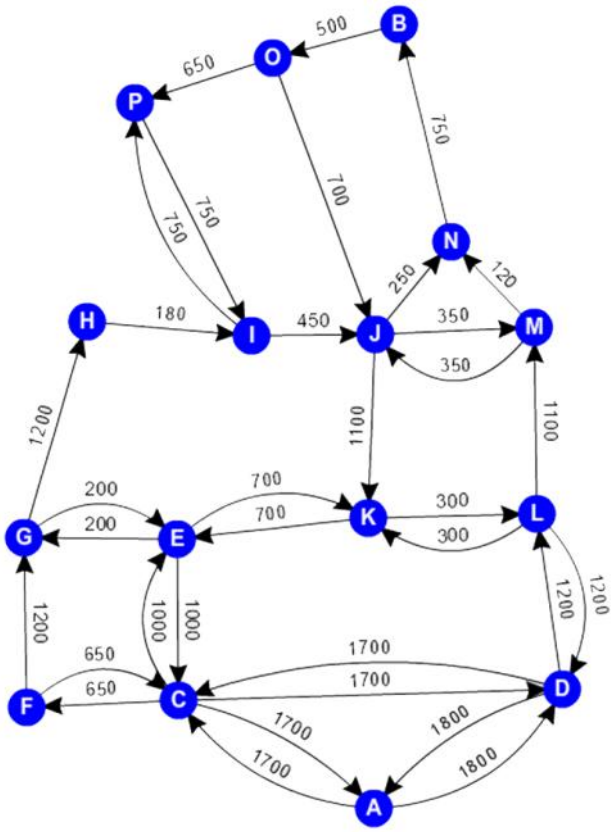


Gambar 9. Peta rumah penulis ke Alun-Alun Bandung. Sumber: Google Maps

Poin berwarna biru merepresentasikan rumah penulis, sedangkan poin berwarna hijau alun-alun kota Bandung.

B. Simpul dan Bobot

Untuk memudahkan dalam melakukan komputasi dan mengamati keterhubungan pada peta, penulis melakukan representasi peta dari rumah penulis ke alun-alun kota Bandung dalam sebuah graf berarah yang memiliki bobot jarak dalam satuan meter.



Gambar 10. Representasi graf peta rumah penulis ke Alun-Alun Bandung. Sumber: buatan penulis

Legend:

A = Rumah penulis

B = Alun-alun kota Bandung

C = Perempatan Cibaduyut - Leuwi Panjang

D = Perempatan Moh. Toha – Soekarno-Hatta

E = Pertigaan Leuwi Panjang - Peta

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 = Pertigaan Dewi Sartika - Pungkur

O = Perempatan Asia Afrika – Otista

P = Perempatan Sudirman – Astana Anyar

IV. IDE PENYELESAIAN MASALAH

Dalam menyelesaikan persoalan untuk mendapatkan rute perjalanan optimum di Kota Bandung, Ide penyelesaian masalah yang digunakan adalah dengan menggunakan pendekatan Algoritma Dijkstra.

Pseudo code Algoritma Dijkstra yang digunakan untuk melakukan eksperimen menyelesaikan masalah sebagai berikut:

```
Function Dijkstra (input start: node_asal)

Deklarasi:

listDistance : inisialisasi list yang berisi bobot per simpul

listRute : inisialisasi list berisi rute per simpul

listVisited : insialisasi list boolean belum dikunjungi

U : integer {indeks minimumm}


Algoritma

listDistance[0] = 0 {simpul awal selalu 0, sisanya tak hingga}


for semua simpul di graf

   U = bobot simpul minimum yang belum dikunjungi

   listVisited[u] = True

   listRute[u] = listRute[u] + u {listRute untuk simpul ke-u ditambahkan u}

   for tetangga dari simpul

      if bobot tetangga > bobot simpul + distance[u][tetangga] then

           bobot tetangga = bobot simpul + distance[u][tetangga]

           listrute[tetangga] = listrute[u]

       Endif

   endfor

endfor

return listDistance,listRute
```

Source code dari program dapat dilihat melalui tautan yang penulis lampirkan di bagian Apendiks.

V. HASIL PENGUJIAN DAN ANALISIS

Untuk melakukan pengujian digunakan matriks ketetanggaan yang merepresentasiikan graf pada gambar 10.

```
[
[0, 0, 1700, 1800, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 500, 0],
[1700, 0, 0, 1700, 1000, 650, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[1800, 0, 1700, 0, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0],
[0, 0, 1000, 0, 0, 0, 200, 0, 0, 0, 700, 0, 0, 0, 0],
[0, 0, 650, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 200, 0, 0, 1200, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 180, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 450, 0, 0, 0, 0, 750],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1100, 0, 350, 250, 0, 0],
[0, 0, 0, 0, 700, 0, 0, 0, 0, 0, 0, 300, 0, 0, 0, 0],
[0, 0, 0, 1200, 0, 0, 0, 0, 0, 0, 300, 0, 1100, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 350, 0, 0, 0, 120, 0, 0],
[0, 750, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 700, 0, 0, 0, 0, 0, 650],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 750, 0, 0, 0, 0, 0, 0]
]
```

Gambar 11. Matriks ketetanggan untuk kasus alun-alun.
Sumber: buatan penulis

Hasil yang dikeluarkan dari program sebagai berikut:

```
File  Edit  View  Search  Terminal  Help
(base) lukaskurnia@Ipray-J22:~/Lukas/Coding/IF/Stima/Makalah/D
Jarak optimum dari Rumah Penulis ke seluruh tempat lain:
1 Rumah Penulis = 0 dengan rute :
- Rumah Penulis
2 Alun-alun kota Bandung = 4970 dengan rute :
- Rumah Penulis
- Perempatan Moh. Toha - Soekarno-Hatta
- Perempatan BKR - Moh. Toha
- Pertigaan Moh. Toha - Pungkur
- Pertigaan Dewi Sartika - Pungkur
- Alun-alun kota Bandung
3 Perempatan Cibaduyut - Leuwi Panjang = 1700 dengan rute :
- Rumah Penulis
- Perempatan Cibaduyut - Leuwi Panjang
4 Perempatan Moh. Toha - Soekarno-Hatta = 1800 dengan rute :
- Rumah Penulis
- Perempatan Moh. Toha - Soekarno-Hatta
5 Pertigaan Leuwi Panjang - Peta = 2700 dengan rute :
- Rumah Penulis
- Perempatan Cibaduyut - Leuwi Panjang
- Pertigaan Leuwi Panjang - Peta
6 Perempatan Kopo - Soekarno-Hatta = 2350 dengan rute :
- Rumah Penulis
- Perempatan Cibaduyut - Leuwi Panjang
- Perempatan Kopo - Soekarno-Hatta
7 Perempatan Peta - Kopo = 2900 dengan rute :
- Rumah Penulis
- Perempatan Cibaduyut - Leuwi Panjang
- Pertigaan Leuwi Panjang - Peta
- Perempatan Peta - Kopo
8 Pertigaan Pasir Koja - Kopo = 4100 dengan rute :
- Rumah Penulis
- Perempatan Cibaduyut - Leuwi Panjang
- Pertigaan Leuwi Panjang - Peta
- Perempatan Peta - Kopo
- Pertigaan Pasir Koja - Kopo
```

```
9 Pertigaan Pasir Koja - Astana Anyar = 4280 dengan rute :
- Rumah Penulis
- Perempatan Cibaduyut - Leuwi Panjang
- Pertigaan Leuwi Panjang - Peta
- Perempatan Peta - Kopo
- Pertigaan Pasir Koja - Kopo
- Pertigaan Pasir Koja - Astana Anyar
10 Perempatan Pungkur - Otista = 4450 dengan rute :
- Rumah Penulis
- Perempatan Moh. Toha - Soekarno-Hatta
- Perempatan BKR - Moh. Toha
- Pertigaan Moh. Toha - Pungkur
- Perempatan Pungkur - Otista
11 Perempatan Peta - Inhoftank = 3300 dengan rute :
- Rumah Penulis
- Perempatan Moh. Toha - Soekarno-Hatta
- Perempatan BKR - Moh. Toha
- Perempatan Peta - Inhoftank
12 Perempatan BKR - Moh. Toha = 3000 dengan rute :
- Rumah Penulis
- Perempatan Moh. Toha - Soekarno-Hatta
- Perempatan BKR - Moh. Toha
13 Pertigaan Moh. Toha - Pungkur = 4100 dengan rute :
- Rumah Penulis
- Perempatan Moh. Toha - Soekarno-Hatta
- Perempatan BKR - Moh. Toha
- Pertigaan Moh. Toha - Pungkur
14 Pertigaan Dewi Sartika - Pungkur = 4220 dengan rute :
- Rumah Penulis
- Perempatan Moh. Toha - Soekarno-Hatta
- Perempatan BKR - Moh. Toha
- Pertigaan Moh. Toha - Pungkur
- Pertigaan Dewi Sartika - Pungkur
15 Perempatan Asia Afrika - Otista = 5470 dengan rute :
- Rumah Penulis
- Perempatan Moh. Toha - Soekarno-Hatta
- Perempatan BKR - Moh. Toha
- Pertigaan Moh. Toha - Pungkur
- Pertigaan Dewi Sartika - Pungkur
- Alun-alun kota Bandung
- Perempatan Asia Afrika - Otista
16 Perempatan Sudirman - Astana Anyar = 5030 dengan rute :
- Rumah Penulis
- Perempatan Cibaduyut - Leuwi Panjang
- Pertigaan Leuwi Panjang - Peta
- Perempatan Peta - Kopo
- Pertigaan Pasir Koja - Kopo
- Pertigaan Pasir Koja - Astana Anyar
- Perempatan Sudirman - Astana Anyar
(base) lukaskurnia@Ipray-J22:~/Lukas/Coding/IF/Stima/Makalah/
```

Gambar 12. Hasil ouput program. Sumber: buatan penulis

Berdasarkan hasil yang dikeluarkan program, kita dapat melihat jarak tempuh optimum dari titik awal yaitu rumah penulis ke seluruh titik pada graf. Kita dapat mengambil kesimpulan dari no 2 pada gambar 12, jarak dari rumah penulis ke Alun-Alun kota Bandung sebesar 4970m atau sekitar 4.97km dengan melalui rute optimum yaitu:

- Rumah Penulis
- Perempatan Moh. Toha – Soekarno-Hatta
- Perempatan BKR – Moh. Toha
- Pertigaan Moh. Toha – Pungkur
- Pertigaan Dewi Sartika – Pungkur
- Alun-alun kota Bandung

VI. Kesimpulan dan Saran

Algoritma Dijkstra dapat menjadi salah satu solusi untuk menentukan rute perjalanan optimal di kota Bandung. Dengan menggunakan strategi greedy pada algoritma dijkstra, kita dapat menelusuri simpul(persimpangan) satu persatu yang paling optimal hingga memetakan seluruh simpul(persimpangan) yang dilalui dengan jarak terpendek.

Ke depannya, batasan masalah yang dibahas di makalah ini dapat lebih ditingkatkan dalam penelitian di masa yang akan datang.

VII. Apendiks

Implementasi program yang dibuat penulis beserta hasil percobaan dapat diakses pada GitHub melalui tautan https://github.com/lukaskurnia/Dijkstra-Algorithm . Program ditulis dengan bahasa Python3 sehingga diperlukan Python3 untuk mengeksekusi program.

VIII. Ucapan Terimakasih

Pertama-tama penulis mengucapkan terimakasih kepada Tuhan Yang Maha Esa, sehingga penulis dapat menyelesaikan makalah ini dengan baik. Selain itu, tidak lupa penulis juga mengucapkan terimakasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T, Ibu Dr. Masayu Leylia Khodra, dan Ibu Dr. Nur Ulfa Maulidevi, S.T sebagai dosen mata kuliah strategi algoritma di program studi Teknik Informatika Institut Teknologi Bandung yang sudah membimbing penulis dalam mengikuti pelajaran selama satu semester ini. Kemudian, penulis juga mengucapkan terimakasih kepada kedua orang tua, keluarga, teman-teman penulis atas dukungan dan dorongannya untuk tetap semangat mengerjakan makalah ini.

Referensi

[1] Munir, Rinaldi. Diktat Kuliah IF2120 Matematika Diskrit. Program Studi Teknik Informatika ITB. 2015.

[2] Munir, Rinaldi. Diktat Kuliah IF2211 Strategi Algortima. Program Studi Teknik Informatika ITB. 2019.

[3] https://ppid.bandung.go.id/profil-kota-bandung/ . Profil Kota Bandung. Diakses pada 25 April 2019, pukul 23:01

[4] http://mti.binus.ac.id/2017/11/28/algoritma-dijkstra/ . Algoritma Dijkstra. Diakses pada 26 April 2019, pukul 00:47

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, 26 April 2019

Lukas Kurnia Jonathan
13517006