PENCARIAN LOKASI APOTEK TERDEKAT MENGGUNAKAN ALGORITMA *FLOYD-WARSHAL*

Rully Mujiastuti, Rizky Purwanto, Sugiartowo
Universitas Muhammadiyah Jakarta

rully.mujiastuti@ftumj.ac.id

ABSTRAK

Apotek merupakan sebuah tempat bagi masyarakat untuk mendapatkan pelayanan kefarmasian dari apoteker. Pelayanan tersebut berupa penyaluran dan penjualan obat-obatan, baik melalui resep dokter (obat keras) maupun obat yang dijual bebas tanpa resep. Keberadaan apotek tersebut tersebar luas di berbagai wilayah, termasuk di Jakarta Timur. Pada penelitian ini, digunakan algoritma *Floyd Warshal* untuk mencari lokasi apotek yang terdekat. Algoritma *Floyd Warshal* merupakan algoritma pemrograman dinamis yang menghitung seluruh lintasan yang ada di dalam graf, dan mencari jarak terdekat antara *node* awal dan *node* tujuan melalui proses iterasi dengan nilai tidak berupa *negative cycle.* Dengan menggunakan input berupa lokasi user (sebagai lokasi awal menggunakan *Global Positioning System*), dan jenis obat, kemudian dilakukan pemrosesan data menggunakan algoritma *Floyd Warshal* dan dihasilkan output berupa peta, lokasi user dan apotek, pencarian apotek, daftar apotek terdekat pada lokasi pengguna, disertai no telp dan rute jalan menuju apotek serta menu about mengenai penjelasan aplikasi.

Kata Kunci: Apotek, Apotek Terdekat, Algoritma *Floyd Warshal*, Jarak Terdekat

ABSTRACT

Pharmacy is a place for people to get pharmaceutical services from pharmacists. The service is in the form of distribution and sale of medicines, either through doctor's prescriptions (hard drugs) or drugs that are sold freely without prescription. The existence of the pharmacy is widespread in various regions, including in East Jakarta. In this study, the Floyd Warshal algorithm was used to find the location of the nearest pharmacy. Floyd Warshal Algorithm is a dynamic programming algorithm that calculates all paths in a graph, and searches for the closest distance between the initial node and the destination node through an iteration process with the value not in the form of a negative cycle. By using the input in the form of a user location (as the initial location using the Global Positioning System), and the type of drug, then processing the data using the Floyd Warshal algorithm and producing output in the form of maps, user locations and pharmacies, pharmacy searches, the nearest pharmacy list at the user's location, accompanied phone number and route to the pharmacy and the menu about the explanation of the application.

.

Key words : *Pharmacy, Nearby Pharmacy, Floyd Warshal Algorithm, Nearby Routes*

1. Pendahuluan

Apotek merupakan sebuah tempat bagi masyarakat untuk mendapatkan pelayanan kefarmasian dari apoteker. Pelayanan tersebut berupa penyaluran dan penjualan obat-obatan, baik melalui resep dokter (obat keras) maupun obat yang dijual bebas tanpa resep.

Pengguna yang membutuhkan obat dapat berkunjung ke apotek untuk

mendapatkan obat yang diperlukan. Namun, terkadang obat yang dicari tidak tersedia pada apotek yang dituju, sehingga membutuhkan waktu dan informasi untuk mencarinya. Apalagi, jumlah apotek tersebar di wilayah yang cukup luas, maka diperlukan sebuah informasi untuk keberadaan lokasi apotek yang terdekat dengan jarak tempuh yang paling cepat untuk mendapatkan obat dengan segera.

Jarak tempuh yang terdekat dan informasi ketersediaan obat tersebut dapat dicari dengan menggunakan Algoritma *Floyd Warshal.* Algortitma ini menghitung seluruh lintasan yang ada di dalam graf mulai titik awal sampai titik tujuan. Pada titik lokasi awal menggunakan *Global Positioning System* (GPS), kemudian pada pengolahannya menggunakan Algoritma *Floyd Warshal* dan menuju titik tujuan dengan mengacu pada daftar apotek yang diurutkan dari lokasi terdekat berdasarkan lokasi *user* disertai dengan nomor telepon apotek dan rute jalan menuju apotik terdekat.

2. **Metode**

Data penelitian dikumpulkan melalui observasi, dokumentasi, literatur dan wawancara. Data yang dikumpulkan berupa daftar alamat apotek disertai dengan titik koordinat lokasi apotek. Variabel yang digunakan adalah jarak, misalnya dari *node* A (titik awal) dengan *node* J (titik tujuan) dengan satuan kilometer (Km).

Langkah yang digunakan untuk pencarian rute terpendeknya adalah sebagai berikut :

1. Merepresentasikan graf ke dalam bentuk tabel.
2. Menjadikan masing-masing *node* sebagai penghubung dalam rute.
3. Mencari rute baru dengan menjadikan masing-masing *node* sebagai penghubung

3. **Landasan Teori**

A. Algoritma *Floyd Warshal.*

Algoritma *Floyd-Warshall* adalah matriks hubung graf berarah berlabel, dan keluarannya adalah path terpendek. Dalam usaha untuk mencari path terpendek, Algoritma *Floyd-Warshall* memulai iterasi dari titik awalnya kemudian memperpanjang path dengan mengevaluasi titik demi titik hingga mencapai tujuan dengan jumlah bobot yang seminimum mungkin. (Hasibuan, 2016)

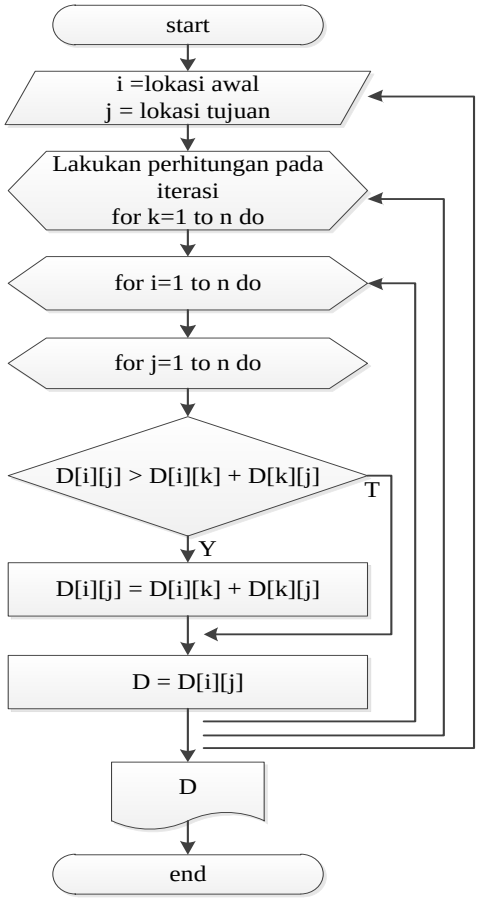
Menurut Sjukani (2007) dalam Faruk (2015), Algoritma *Floyd-Warshall* tidak sekedar mencari lintasan terpendek antara dua buah simpul tertentu, tetapi langsung membuat tabel lintasan terpendek antar simpul. Algoritma *Floyd-Warshall* merupakan salah satu varian pemrograman dinamis, yaitu suatu metode yang melakukan pemecahan masalah dengan memandang solusi yang akan diperoleh sebagai suatu keputusan yang saling terkait. Artinya solusi-solusi tersebut dibentuk dari solusi yang berasal dari tahap sebelumnya dan ada kemungkinan solusi lebih dari satu.

B. Deskripsi Algoritma *Floyd-Warshall*

Dasar dari Algoritma *Floyd-Warshall* menurut Hasibuan (2016), yaitu :

1. Asumsikan semua simpul graf G adalah V = {1, 2, 3, 4, 5, …, n}.
2. Untuk setiap pasangan simpul awal dan tujuan (i, j) diperhatikan semua lintasan dari i ke j dimana semua simpul pertengahan diambil dari {1, 2, …, k}, dan p adalah lintasan berbobot minimum diantara semuanya.
3. Algoritma ini mengeksploitasi relasi antara lintasan p dan lintasan terpendek dari i ke j dengan semua simpul pertengahan berada pada himpunan {1, 2, …, k-1}
4. Relasi tersebut bergantung apakah k adalah simpul pertengahan pada lintasan p

Gambar 1 berikut adalah flowchart algoritma *Floyd-Warshall*



Gambar 1. *Flowchart* Algoritma *Floyd-Warshall*

C. Karakteristik Algoritma *Floyd-Warshall*

Terdapat beberapa karakteristik yang dimiliki oleh Algoritma *Floyd-Warshall* menurut Putra (2015), antara lain :

1. Persoalan dibagi atas beberapa tahap, yang setiap tahapnya hanya akan diambil satu keputusan.
2. Masing-masing tahap terdiri atas sejumlah status yang saling berhubungan dengan status tersebut. Status yang dimaksud di sini adalah berbagai kemungkinan masukan yang ada pada tahap tersebut.
3. Ketika masuk ke suatu tahap, hasil keputusan akan transformasi.
4. Bobot pada suatu tahap akan meningkat secara teratur seiring bertambahnya jumlah tahapan.
5. Bobot yang ada pada suatu tahap tergantung dari bobot tahapan yang telah berjalan dan bobot pada tahap itu sendiri.
6. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan pada tahap sebelumnya.
7. Terdapat hubungan rekursif yang menyatakan bahwa keputusan terbaik dalam setiap status pada tahap k akan memberikan keputusan terbaik untuk setiap status pada tahap k + 1.
8. Prinsip optimalitas berlaku pada persoalan yang dimaksud.

D. **GRAF**

Menurut Munir (2005) dalam Lukmana (2017), Graf adalah kumpulan *node* atau *vertex* yang dihubungkan satu sama lain melalui sisi/*edge*, yang digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Graf G didefinisikan sebagai pasangan himpunan (V ,E), ditulis dengan notasi G (V,E), yang dalam hal ini.

- V adalah himpunan tidak kosong dari simpul-simpul (titik/*vertex*/*node*).
- E adalah himpunan sisi (rusuk/*edge*) yang menghubungkan sepasang simpul.

Vertex pada graf dapat merupakan objek sembarang seperti kota, atom-atom suatu zat, nama anak, jenis buah, komponen alat elektronik dan sebagainya. *Edge* dapat menunjukan hubungan sembarang seperti rute penerbangan, jalan raya, sambungan telepon, ikatan kimia, dan lain-lain. Jika terdapat sebuah rusuk e yang menghubungkan *vertex* v dan w, ditulis *edge* (v,w).

Komponen-komponen graf menurut Saputra (2011) adalah :

1. *Vertex*
   Vertex (simpul atau titik) yang disimbolkan dengan V adalah himpunan simpul yang terbatas dan tidak kosong. Jumlah simpul pada graf dapat dinyatakan dengan n = |V|.
2. *Edge*
   Edge (sisi, busur atau rusuk) yang disimbolkan dengan E adalah himpunan

rusuk yang menghubungkan sepasang simpul. Jumlah rusuk dan graf dinyatakan dengan m = |E|.

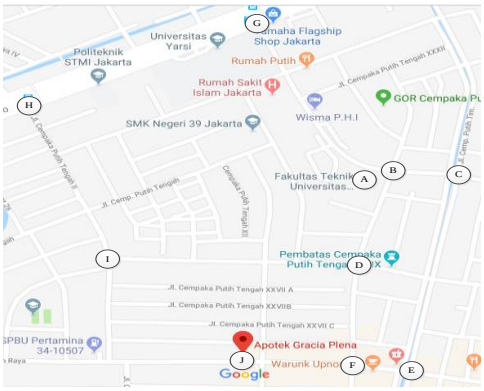
3. *Degree*
   Degree (Derajat) suatu simpul yang disimbolkan dengan d(v) adalah jumlah rusuk yang berada pada simpul tersebut.
4. *Size*
   Size (ukuran) dari suatu *graf* adalah banyaknya simpul yang dimiliki.

4.Hasil dan Pembahasan

Pencarian apotek terdekat ini menggunakan metode *Floyd-Warshall* dalam menentukan rute jalan terdekat yang akan dipilih. Metode *Floyd-Warshall* melakukan pemecahan masalah dengan memandang solusi yang akan diperoleh sebagai suatu keputusan yang saling terkait. Maksudnya, solusi-solusi dibentuk dari solusi yang berasal dari tahap sebelumnya dan ada kemungkinan solusi lebih dari satu.

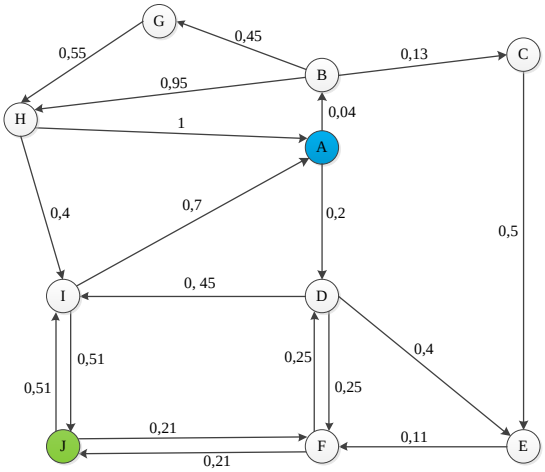
Sebagai contoh adalah untuk mencari jarak terdekat dari titik awal user menggunakan GPS di Cempaka Putih, menuju titik tujuan apotek terdekat yaitu Apotek Gracia Plena. Kemudian pencarian rute tersebut dibuat menjadi sebuah graf berarah, dengan *node* A (warna biru terang) sebagai titik awal (*start*) dan *node* J (warna hijau terang) merupakan titik tujuan dengan jarak tempuh satuan kilometer (km).

Gambar menggunakan GPS tersebut dapat dilihat pada gambar 4.1 dan gambar pencarian rute berupa graf dapat dilihat pada gambar 4.2 seperti berikut ini.



Gambar 2 Pencarian lokasi apotek dengan GPS (*Google Maps*)

Pada gambar di atas, titik awal adalah Fakultas Teknik UMJ dengan *node* A, dan titik tujuan Apotek Gracia Plena dengan *node* J. Kemudian digambarkan dengan graf berikut ini :



Gambar 3. Percobaan pencarian rute (dalam Km)

Adapun penjelasan tiap *node*, mulai *node* A (titik awal) hingga *node* J (titik tujuan) seperti tercantum pada tabel1 berikut ini :

Tabel 1. Hasil Representasi Titik pada *Google Maps*

Titik	Keterangan
A	Fakultas Teknik Universitas Muhammadiyah Jakarta
B	Persimpangan Jl. Cempaka Putih Tengah XXVII dengan Jl. Cempaka Putih Tengah XXX
C	Persimpangan Jl. Cempaka Putih Tengah XXX dengan Jl. Cempaka Putih Timur
D	Persimpangan Jl. Cempaka Putih Tengah XVII dengan Jl. Cempaka Putih Tengah XXVII
E	Persimpangan Jl. Cempaka Putih Timur dengan Jl. Cempaka Putih Raya
F	Persimpangan Jl. Cempaka Putih Tengah XXVII dengan Jl. Cempaka Putih Raya
G	Persimpangan Jl. Cempaka Putih Tengah XXX dengan Jl. Letjen Suprapto
H	Persimpangan Jl. Letjen Suprapto dengan Jl. Cempaka Putih Tengah II
I	Persimpangan Jl. Cempaka Putih Tengah XVII dengan Jl. Cempaka Putih Tengah II dan Jl. Cempaka Putih Raya
J	Apotek Gracia Plena

Dengan menggunakan algoritma Floyd Warshall, maka dilakukan langkah berikut ini :

1. Membuat tabel sebagai representasi dari graf. Seperti pada tabel 2 berikut ini.

Tabel 2. Penghitungan rute *Floyd-Warshall* Tahap Awal (dalam km)

	A	B	C	D	E	F	G	H	I	J
A	0	**0,04**	∞	**0,2**	∞	∞	∞	∞	∞	∞
B	∞	0	**0,13**	∞	∞	∞	**0,45**	**0,95**	∞	∞
C	∞	∞	0	∞	**0,5**	∞	∞	∞	∞	∞
D	∞	∞	∞	0	**0,4**	**0,25**	∞	∞	**0,45**	∞
E	∞	∞	∞	∞	0	**0,11**	∞	∞	∞	∞
F	∞	∞	∞	**0,25**	∞	0	∞	∞	∞	**0,21**
G	∞	∞	∞	∞	∞	∞	0	**0,55**	∞	∞
H	**1**	∞	∞	∞	∞	∞	∞	0	**0,4**	∞
I	**0,7**	∞	∞	∞	∞	∞	∞	∞	0	**0,51**
J	∞	∞	∞	∞	∞	**0,21**	∞	∞	**0,51**	0

Keterangan :

- Angka yang ditebalkan adalah angka terkecil pada rute baru
- Parameter huruf pada kolom paling kiri adalah *node* awal, sementara baris paling atas adalah *node* tujuan.
- Nilai baru tidak dimasukkan ke dalam tabel karena nilai lebih besar dari nilai yang sudah ada sebelumnya

2. Langkah ke-2 masing-masing *node* dijadikan sebagai penghubung antar *node* guna mencari rute baru beserta nilainya yang akan dilalui. Kemudian akan didapatkan hasil seperti tabel 3 berikut ini.

Tabel 3. Iterasi ke-1 (dalam km)

	A	B	C	D	E	F	G	H	I	J
A	0	0,04	∞	0,2	∞	∞	∞	∞	∞	∞
B	∞	0	0,13	∞	∞	∞	0,45	0,95	∞	∞
C	∞	∞	0	∞	0,5	∞	∞	∞	∞	∞
D	∞	∞	∞	0	0,4	0,25	∞	∞	0,45	∞
E	∞	∞	∞	∞	0	0,11	∞	∞	∞	∞
F	∞	∞	∞	0,25	∞	0	∞	∞	∞	0,21
G	∞	∞	∞	∞	∞	∞	0	0,55	∞	∞
H	1	**1,04**	∞	**1,2**	∞	∞	∞	0	0,4	∞
I	0,7	**0,74**	∞	**0,9**	∞	∞	∞	∞	0	0,51
J	∞	∞	∞	∞	∞	0,21	∞	∞	0,51	0

Terdapat nilai yang berubah yaitu nilai rute H – B, H – D, I – B, dan I – D. hal ini dikarenakan *Node* A sebagai penghubung sehingga rute baru tercipta, yaitu:

1. H – A – B = 1 + 0,04 = 1,04
2. H – A – D = 1 + 0,2 = 1,2
3. I – A – B = 0,7 + 0,04 = 0,74
4. I – A – D = 0,7 + 0,2 = 0,9

3. Langkah ke-3 memasukkan nilai rute baru yang tercipta pada *node* B ke dalam tabel 4. Rute baru tercipta, yaitu :
   1. A – B – C = 0,04 + 0,13 = 0,17
   2. A – B – G = 0,04 + 0,45 = 0,49
   3. A – B – H = 0,04 + 0,95 = 0,99
   4. H – A – B – C = 1 + 0,04 + 0,13 = 1,17
   5. H – A – B – G = 1 + 0,04 + 0,45 = 1,49
   6. I – A – B – C = 0,7 + 0,04 + 0,13 = 0,87
   7. I – A – B – G = 0,7 + 0,04 + 0,45 = 1,19

8. I – A – B – H = 0,7 + 0,04 + 0,95 = 1,69

Tabel 4. Iterasi ke-2 (dalam km)

	A	B	C	D	E	F	G	H	I	J
A	0	0,04	**0,17**	0,2	∞	∞	**0,49**	**0,99**	∞	∞
B	∞	0	0,13	∞	∞	∞	0,45	0,95	∞	∞
C	∞	∞	0	∞	0,5	∞	∞	∞	∞	∞
D	∞	∞	∞	0	0,4	0,25	∞	∞	0,45	∞
E	∞	∞	∞	∞	0	0,11	∞	∞	∞	∞
F	∞	∞	∞	0,25	∞	0	∞	∞	∞	0,21
G	∞	∞	∞	∞	∞	∞	0	0,55	∞	∞
H	1	1,04	**1,17**	1,2	∞	∞	**1,49**	0	0,4	∞
I	0,7	0,74	**0,87**	0,9	∞	∞	**1,19**	**1,69**	0	0,51
J	∞	∞	∞	∞	∞	0,21	∞	∞	0,51	0

4. Langkah ke-4 memasukkan nilai rute baru yang tercipta pada *node* C ke dalam tabel 5. Rute baru tercipta, yaitu :

   1. A – B – C – E = 0,04 + 0,13 + 0,5 = 0,67
   2. B – C – E = 0,13 + 0,5 = 0,63
   3. H – A – B – C – E = 1 + 0,04 + 0,13 + 0,5 = 1,67
   4. I – A – B – C – E = 0,7 + 0,04 + 0,13 + 0,5 = 1,37

Tabel 5. Iterasi ke-3 (dalam km)

	A	B	C	D	E	F	G	H	I	J
A	0	0,04	0,17	0,2	**0,67**	∞	0,49	0,99	∞	∞
B	∞	0	0,13	∞	**0,63**	∞	0,45	0,95	∞	∞
C	∞	∞	0	∞	0,5	∞	∞	∞	∞	∞
D	∞	∞	∞	0	0,4	0,25	∞	∞	0,45	∞
E	∞	∞	∞	∞	0	0,11	∞	∞	∞	∞
F	∞	∞	∞	0,25	∞	0	∞	∞	∞	0,21
G	∞	∞	∞	∞	∞	∞	0	0,55	∞	∞
H	1	1,04	1,17	1,2	**1,67**	∞	1,49	0	0,4	∞
I	0,7	0,74	0,87	0,9	**1,37**	∞	1,19	1,69	0	0,51
J	∞	∞	∞	∞	∞	0,21	∞	∞	0,51	0

5. Langkah ke-5 memasukkan nilai rute baru yang tercipta pada *node* D ke dalam tabel 6. Rute baru tercipta, yaitu :
   1. A – D – E = 0,2 + 0,4 = 0,6
   2. A – D – F = 0,2 + 0,25 = 0,45
   3. A – D – I = 0,2 + 0,45 = 0,65
   4. F – D – E = 0,25 + 0,4 = 0,65
   5. F – D – I = 0,25 + 0,45 = 0,7
   6. H – A – D – E = 1 + 0,2 + 0,4 = 1,6
   7. H – A – D – F = 1 + 0,2 + 0,25 = 1,45
   8. I – A – D – E = 0,7 + 0,2 + 0,4 = 1,3
   9. I – A – D – F = 0,7 + 0,2 + 0,25 = 1,15

Tabel 6. Iterasi ke-4 (dalam km)

	A	B	C	D	E	F	G	H	I	J
A	0	0,04	0,17	0,2	**0,6**	**0,45**	0,49	0,99	**0,65**	∞
B	∞	0	0,13	∞	0,63	∞	0,45	0,95	∞	∞
C	∞	∞	0	∞	0,5	∞	∞	∞	∞	∞
D	∞	∞	∞	0	0,4	0,25	∞	∞	0,45	∞
E	∞	∞	∞	∞	0	0,11	∞	∞	∞	∞
F	∞	∞	∞	0,25	**0,65**	0	∞	∞	**0,7**	0,21
G	∞	∞	∞	∞	∞	∞	0	0,55	∞	∞
H	1	1,04	1,17	1,2	**1,6**	**1,45**	1,49	0	0,4	∞
I	0,7	0,74	0,87	0,9	**1,3**	**1,15**	1,19	1,69	0	0,51
J	∞	∞	∞	∞	∞	0,21	∞	∞	0,51	0

6 Langkah ke-6 memasukkan nilai rute baru yang tercipta pada *node* E ke dalam tabel 7. Rute baru tercipta, yaitu :

   1. B – C – E – F = 0,13 + 0,5 + 0,11 = 0,74
   2. C – E – F = 0,5 + 0,11 = 0,61

Tabel 7. Iterasi ke-5 (dalam km)

	A	B	C	D	E	F	G	H	I	J
A	0	0,04	0,17	0,2	0,6	0,45	0,49	0,99	0,65	∞
B	∞	0	0,13	∞	0,63	**0,74**	0,45	0,95	∞	∞
C	∞	∞	0	∞	0,5	**0,61**	∞	∞	∞	∞
D	∞	∞	∞	0	0,4	0,25	∞	∞	0,45	∞
E	∞	∞	∞	∞	0	0,11	∞	∞	∞	∞
F	∞	∞	∞	0,25	0,65	0	∞	∞	0,7	0,21
G	∞	∞	∞	∞	∞	∞	0	0,55	∞	∞
H	1	1,04	1,17	1,2	1,6	1,45	1,49	0	0,4	∞
I	0,7	0,74	0,87	0,9	1,3	1,15	1,19	1,69	0	0,51
J	∞	∞	∞	∞	∞	0,21	∞	∞	0,51	0

7. Langkah ke-7 memasukkan nilai rute baru yang tercipta pada *node* F ke dalam tabel 8. Rute baru tercipta, yaitu :
   1. A – D – F – J = 0,2 + 0,25 + 0,21= 0,66
   2. B – C – E – F – D = 0,13 + 0,5 + 0,11 + 0,25 = 0,99
   3. B – C – E – F – J = 0,13 + 0,5 + 0,11 + 0,21 = 0,95
   4. B – C – E – F – D – I = 0,13 + 0,5 + 0,11 + 0,25 + 0,45 = 1,44
   5. C – E – F – D = 0,5 + 0,11 + 0,25 = 0,86
   6. C – E – F – J = 0,5 + 0,11 + 0,21 = 0,82
   7. C – E – F – D – I = 0,5 + 0,11 + 0,25 + 0,45 = 1,31

8. D – F – J = 0,25 + 0,21 = 0,46
9. E – F – D = 0,11 + 0,25 = 0,36
10. E – F – J = 0,11 + 0,21 = 0,32
11. E – F – D – I = 0,11 + 0,25 + 0,45 = 0,81
12. H – A – D – F – J = 1 + 0,2 + 0,25 + 0,21 = 1,66
13. J – F – D = 0,21 + 0,25 = 0,46
14. J – F – D – E = 0,21 + 0,25 + 0,4 = 0,86

Tabel 8. Iterasi ke-6 (dalam km)

	A	B	C	D	E	F	G	H	I	J
A	0	0,04	0,17	0,2	0,6	0,45	0,49	0,99	0,65	***0,66***
B	∞	0	0,13	**0,99**	0,63	0,74	0,45	0,95	**1,44**	**0,95**
C	∞	∞	0	**0,86**	0,5	0,61	∞	∞	**1,31**	**0,82**
D	∞	∞	∞	0	0,4	0,25	∞	∞	0,45	**0,46**
E	∞	∞	∞	**0,36**	0	0,11	∞	∞	**0,81**	**0,32**
F	∞	∞	∞	0,25	0,65	0	∞	∞	0,7	0,21
G	∞	∞	∞	∞	∞	∞	0	0,55	∞	∞
H	1	1,04	1,17	1,2	1,6	1,45	1,49	0	0,4	**1,66**
I	0,7	0,74	0,87	0,9	1,3	1,15	1,19	1,69	0	0,51
J	∞	∞	∞	**0,46**	**0,86**	0,21	∞	∞	0,51	0

8. Langkah ke-8 tidak ada perubahan karena hasil rute baru yang melalui *node* G nilainya lebih besar dari rute-rute sebelumnya. Nilai dalam tabel 9 masih sama seperti tabel ke-8

Tabel 9. Iterasi ke-7 (dalam km)

	A	B	C	D	E	F	G	H	I	J
A	0	0,04	0,17	0,2	0,6	0,45	0,49	0,99	0,65	***0,66***
B	∞	0	0,13	**0,99**	0,63	0,74	0,45	0,95	**1,44**	**0,95**
C	∞	∞	0	**0,86**	0,5	0,61	∞	∞	**1,31**	**0,82**
D	∞	∞	∞	0	0,4	0,25	∞	∞	0,45	**0,46**
E	∞	∞	∞	**0,36**	0	0,11	∞	∞	**0,81**	**0,32**
F	∞	∞	∞	0,25	0,65	0	∞	∞	0,7	0,21
G	∞	∞	∞	∞	∞	∞	0	0,55	∞	∞
H	1	1,04	1,17	1,2	1,6	1,45	1,49	0	0,4	**1,66**
I	0,7	0,74	0,87	0,9	1,3	1,15	1,19	1,69	0	0,51
J	∞	∞	∞	**0,46**	**0,86**	0,21	∞	∞	0,51	0

9. Langkah ke-9 memasukkan nilai rute baru yang tercipta pada *node* H ke dalam tabel 10. Rute baru tercipta, yaitu :
   1. B – H – A = 0,95 + 1 = 1,95
   2. G – H – A = 0,55 + 1 = 1,55
   3. G – H – I = 0,55 + 0,4 = 0,95
   4. G – H – A – B = 0,55 + 1 + 0,04 = 1,59
   5. G – H – A – D = 0,55 + 1 + 0,2 = 1,75
   6. G – H – A – B – C = 0,55 + 1 + 0,04 + 0,13 = 1,72
   7. G – H – A – D – E = 0,55 + 1 + 0,2 + 0,4 = 2,15
   8. G – H – A – D – F = 0,55 + 1 + 0,2 + 0,25 = 2
   9. G – H – A – D – F – J = 0,55 + 1 + 0,2 + 0,25 + 0,21 = 2,21

Tabel 10. Iterasi ke-8 (dalam km)

	A	B	C	D	E	F	G	H	I	J
A	0	0,04	0,17	0,2	0,6	0,45	0,49	0,99	0,65	*0,66*
B	**1,95**	0	0,13	0,99	0,63	0,74	0,45	0,95	1,44	0,95
C	∞	∞	0	0,86	0,5	0,61	∞	∞	1,31	0,82
D	∞	∞	∞	0	0,4	0,25	∞	∞	0,45	0,46
E	∞	∞	∞	0,36	0	0,11	∞	∞	0,81	0,32
F	∞	∞	∞	0,25	0,65	0	∞	∞	0,7	0,21
G	**1,55**	**1,59**	**1,72**	**1,72**	**2,15**	**2**	0	0,55	**0,95**	**2,21**
H	1	1,04	1,17	1,2	1,6	1,45	1,49	0	0,4	1,66
I	0,7	0,74	0,87	0,9	1,3	1,15	1,19	1,69	0	0,51
J	∞	∞	∞	0,46	0,86	0,21	∞	∞	0,51	0

10. Langkah ke-10 memasukkan nilai rute baru yang tercipta pada *node* I ke dalam tabel 9. Rute baru tercipta, yaitu :
    1. C – E – F – D – I – A = 0,5 + 0,11 + 0,25 + 0,45 + 0,7 = 2,01
    2. C – E – F – D – I – A – B = 0,5 + 0,11 + 0,25 + 0,45 + 0,7 + 0,04 = 2,05
    3. C – E – F – D – I – A – B – G = 0,5 + 0,11 + 0,25 + 0,45 + 0,7 + 0,04 + 0,45 = 2,5
    4. C – E – F – D – I – A – B – H = 0,5 + 0,11 + 0,25 + 0,45 + 0,7 + 0,04 + 0,95 = 3
    5. D – I – A = 0,45 + 0,7 = 1,15
    6. D – I – A – B = 0,45 + 0,7 + 0,04 = 1,19
    7. D – I – A – B – C = 0,45 + 0,7 + 0,04 + 0,13 = 1,32
    8. D – I – A – B – G = 0,45 + 0,7 + 0,04 + 0,45 = 1,64
    9. D – I – A – B – H = 0,45 + 0,7 + 0,04 + 0,95 = 2,14
    10. E – F – D – I – A = 0,11 + 0,25 + 0,45 + 0,7 = 1,51
    11. E – F – D – I – A – B = 0,11 + 0,25 + 0,45 + 0,7 + 0,04 = 1,55
    12. E – F – D – I – A – B – C = 0,11 + 0,25 + 0,45 + 0,7 + 0,04 + 0,13 = 1,68
    13. E – F – D – I – A – B – G = 0,11 + 0,25 + 0,45 + 0,7 + 0,04 + 0,45 = 2
    14. E – F – D – I – A – B – H = 0,11 + 0,25 + 0,45 + 0,7 + 0,04 + 0,95 = 2,5
    15. F – D – I – A = 0,25 + 0,45 + 0,7 = 1,40
    16. F – D – I – A – B = 0,25 + 0,45 + 0,7 + 0,04 = 1,44
    17. F – D – I – A – B – C = 0,25 + 0,45 + 0,7 + 0,04 + 0,13 = 1,57
    18. F – D – I – A – B – G = 0,25 + 0,45 + 0,7 + 0,04 + 0,45 = 1,89
    19. F – D – I – A – B – H = 0,25 + 0,45 + 0,7 + 0,04 + 0,95 = 2,39

20. G – H – I – J = 0,55 + 0,4 + 0,51 = 1,46
21. H – I – J = 0,4 + 0,51 = 0,91
22. J – I – A = 0,51 + 0,7 = 1,21
23. J – I – A – B = 0,51 + 0,7 + 0,04 = 1,25
24. J – I – A – B – C = 0,51 + 0,7 + 0,04 + 0,13 = 1,38
25. J – I – A – B – G = 0,51 + 0,7 + 0,04 + 0,45 = 1,7
26. J – I – A – B – H = 0,51 + 0,7 + 0,04 + 0,95 = 2,2

Tabel 11 Iterasi ke-9 (dalam km)

	A	B	C	D	E	F	G	H	I	J
A	0	0,04	0,17	0,2	0,6	0,45	0,49	0,99	0,65	*0,66*
B	1,95	0	0,13	0,99	0,63	0,74	0,45	0,95	1,44	0,95
C	**2,01**	**2,05**	0	0,86	0,5	0,61	**2,5**	**3**	1,31	0,82
D	**1,19**	**1,32**	**1,32**	0	0,4	0,25	**1,64**	**2,14**	0,45	0,46
E	**1,51**	**1,55**	**1,68**	0,36	0	0,11	**2**	**2,5**	0,81	0,32
F	**1,40**	**1,44**	**1,57**	0,25	0,65	0	**1,89**	**2,39**	0,7	0,21
G	1,55	1,59	1,72	1,72	2,15	2	0	0,55	0,95	**1,46**
H	1	1,04	1,17	1,2	1,6	1,45	1,49	0	0,4	**0,91**
I	0,7	0,74	0,87	0,9	1,3	1,15	1,19	1,69	0	0,51
J	**1,21**	**1,25**	**1,38**	0,46	0,86	0,21	**1,7**	**2,2**	0,51	0

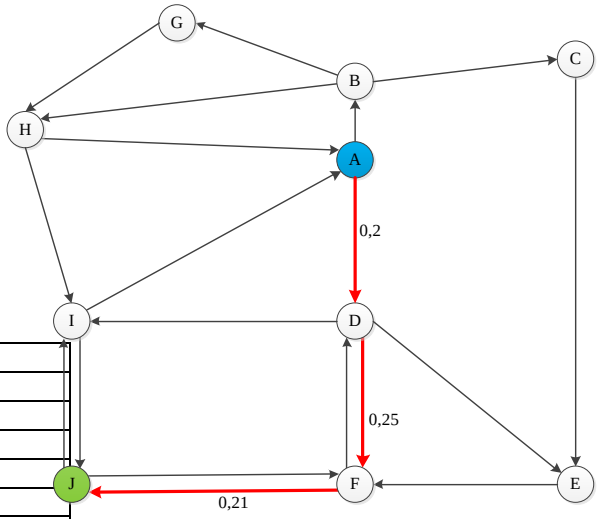
11. Langkah ke-11 tidak ada perubahan karena hasil rute baru yang melalui *node* G nilainya lebih besar dari rute-rute sebelumnya. Nilai dalam tabel 12 masih sama seperti tabel ke-11

Tabel 12. Iterasi ke-10 (dalam km)

	A	B	C	D	E	F	G	H	I	J
A	0	0,04	0,17	0,2	0,6	0,45	0,49	0,99	0,65	*0,66*
B	1,95	0	0,13	0,99	0,63	0,74	0,45	0,95	1,44	0,95
C	2,01	2,05	0	0,86	0,5	0,61	2,5	3	1,31	0,82
D	1,19	1,32	1,32	0	0,4	0,25	1,64	2,14	0,45	0,46
E	1,51	1,55	1,68	0,36	0	0,11	2	2,5	0,81	0,32
F	1,40	1,44	1,57	0,25	0,65	0	1,89	2,39	0,7	0,21
G	1,55	1,59	1,72	1,72	2,15	2	0	0,55	0,95	1,46
H	1	1,04	1,17	1,2	1,6	1,45	1,49	0	0,4	0,91
I	0,7	0,74	0,87	0,9	1,3	1,15	1,19	1,69	0	0,51
J	1,21	1,25	1,38	0,46	0,86	0,21	1,7	2,2	0,51	0

Tabel 12 diatas merupakan tabel hasil akhir dari penghitungan rute menggunakan metode *Floyd-Warshall*. Berdasarkan tabel tersebut rute terpendek dari titik awal (Fakultas Teknik Universitas Muhammadiyah Jakarta) menuju titik tujuan (Apotek Gracia Plena) dari gambar sebelumnya adalah melalui rute A – D – F – J dengan total bobot jarak sebesar 0,2 + 0,25 + 0,21 = 0,66 km atau nilainya setara dengan 660 meter.

Gambar 3 berikut merupakan rute hasil dari perhitungan dengan metode *Floyd-Warshall*.



Gambar 4. Rute hasil perhitungan Algoritma *Floyd Warshall* (dalam km)

5. KESIMPULAN

1. Dapat disimpulkan bahwa telah terbentuk rute –rute terpendek dengan menggunakan Algoritma *Floyd Warshall.*
2. Rute terdekat yang dapat ditempuh adalah **A – D – F – J**. Yaitu *node* A (Fakultas Teknik UMJ) ke *node* B (Persimpangan Jl. Cempaka Putih Tengah XVII dengan Jl. Cempaka Putih Tengah XXVII), kemudian melintasi *node* F (Persimpangan Jl. Cempaka Putih Tengah XXVII dengan Jl. Cempaka Putih Raya) dan berakhir di *node* J (Apotek Gracia Plena).

6. DAFTAR PUSTAKA

Faruk, A. (2015). *Pembuatan Aplikasi Pencarian Rumah Sakit BPJS Berbasis Android dengan Metode Floyd-Warshall Studi Kasus Wilayah Jakarta Utara*. Jakarta: Tugas Akhir Program Teknik Informatika Universitas Muhammadiyah Jakarta.

Google Maps. (2013, September). [Online] diakses, from Google Maps: https: //www.google.co.id/maps/@-6.2081813,106.9318739,14z

Hasibuan, A. R. (2016). Penerapan Algoritma Floyd Warshall untuk Menentukan Jalur

Terpendek dalam Pengiriman Barang. *Jurnal Riset Komputer (JURIKOM), Vol. 3 No. 6*, 20.

Lukmana, A. G. (2017). *Aplikasi Trade Center Mini Maps Menggunakan Metode Floyd-Warshall.* Jakarta: Tugas Akhir Program Teknik Informatika Universitas Muhammadiyah Jakarta.

Putra, Z. (2015). Sistem Pencarian Jalur Terpendek di Kota Medan Menggunakan Algoritma Floyd-Warshall. *Repository USU*, 23.

Purwanto, R(2017). Implementasi Algoritma*Floyd-Warshall* Untuk Mencari Lokasi Apotek Terdekat Di Wilayah Jakarta Timur

Saputra, R. (2011). Sistem Informasi Geografis Pencarian Rute Optimum Obyek Wisata Kota Yogyakarta dengan Algoritma Floyd-Warshall. *Jurnal Matematika Vol. 14, No. 1*.