

**ANALISIS PERBANDINGAN ALGORITMA DIJKSTRA DAN FLOYD
WARSHALL UNTUK MENCARI JARAK TERPENDEK PADA PENCARIAN
FASILITAS KESEHATAN**

SKRIPSI

DWI UTAMI PUTRI

191401013



**PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024**

**ANALISIS PERBANDINGAN ALGORITMA DIJKSTRA DAN FLOYD
WARSHALL UNTUK MENCARI JARAK TERPENDEK PADA PENCARIAN
FASILITAS KESEHATAN**

SKRIPSI

**Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah
Sarjana Ilmu Komputer**

**DWI UTAMI PUTRI
191401013**



**PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024**

PENGESAHAN

Judul : ANALISIS PERBANDINGAN ALGORITMA DIJKSTRA
DAN FLOYD WARSHALL UNTUK MENCARI JARAK
TERPENDEK PADA PENCARIAN FASILITAS
KESEHATAN

Kategori : SKRIPSI

Nama : DWI UTAMI PUTRI

Nomor Induk Mahasiswa : 191401013

Program Studi : SARJANA (S-1) ILMU KOMPUTER

Fakultas : ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA

Medan, 29 April 2024

Komisi Pembimbing :

Pembimbing II



Dewi Sartika Br Ginting, S.Kom, M.Kom
NIP. 199005042019032023

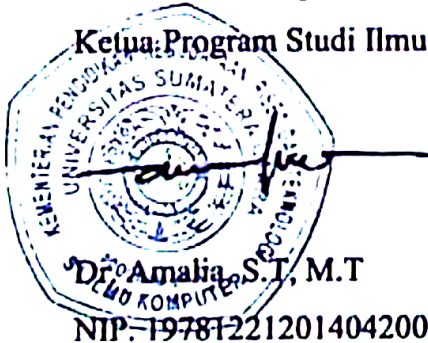
Pembimbing I



Dian Rachmawati S.Si, M.Kom
NIP. 198307232009122004

Diketahui/Disetujui Oleh

Ketua Program Studi Ilmu Komputer



Dr. Amalia S.T, M.T
NIP. 197812212014042001

PERNYATAAN**ANALISIS PERBANDINGAN ALGORITMA DIJKSTRA DAN FLOYD
WARSHALL UNTUK Mencari Jarak Terpendek pada Pencarian
Fasilitas Kesehatan****SKRIPSI**

Saya menegaskan bahwa karya skripsi ini merupakan hasil dari usaha dan pemikiran saya sendiri, dengan pengecualian beberapa kutipan dan rangkuman yang saya berikan sumbernya secara jelas.

Medan, 29 April 2024



Dwi Utami Putri

191401013

PENGHARGAAN

Alhamdulillah Rabbil 'Alamin, penulis mengucapkan segala puji dan rasa syukur kepada Allah SWT., Tuhan yang Maha Esa. Dengan limpahan berkat dan karunia-Nya, penulis berhasil menuntaskan skripsi ini yang merupakan syarat dalam meraih gelar Sarjana Komputer dari Program Studi S-1 Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi di Universitas Sumatera Utara. Shalawat serta salam senantiasa tercurah untuk Rasulullah yang mulia, Nabi Muhammad SAW, yang telah menjadi penuntun dan pembawa rahmat bagi seluruh umat manusia. Semoga keberkahan dan keselamatan senantiasa terlimpah kepada beliau dan para pengikutnya.

Penulis menyampaikan apresiasi yang tulus dan mendalam teruntuk:

1. Bapak Dr. Muryanto Amin, S.Sos, M.Si. yang menjabat sebagai Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia, B.Sc., M.Sc. sebagai Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
3. Ibu Dr. Amalia ST., M.T. selaku Ketua Program Studi S-1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
4. Ibu Dian Rachmawati S.Si, M.Kom selaku dosen pembimbing I yang telah banyak memberikan arahan, kritik, serta bimbingan yang berharga dalam penyusunan skripsi ini.
5. Ibu Dewi Sartika Br Ginting, S.Kom., M.Kom selaku dosen pembimbing II yang membrserikan bimbingan, kritik, motivasi, dan saran sangat membantu dalam menyelesaikan skripsi ini.
6. Ibu Sri Melvani Hardi, S.Kom., M.Kom, sebagai dosen pembimbing akademik yang sudah memberi bantuan, motivasi, serta saran dan petunjuk selama periode studi penulis.
7. Para dosen yang mengaajar di Program Studi S-1 Ilmu Komputer Universitas Sumatera Utara yang telah menyisihkan waktu dan energi mereka untuk memberikan pengajaran dan bimbingan yang berharga, sehingga penulis dapat mencapai ke tahap pembuatan skripsi ini.

8. Seluruh anggota staf administratif Fakultas Ilmu Komputer dan Teknologi Informasi di Universitas Sumatera Utara atas segala bantuan yang diberikan selama masa studi hingga proses penyusunan skripsi ini.
9. Keluarga penulis tercinta, Bapak Ahmad Jaya Putra, S.E., Ibu Rosilawaty, Abang Indra Yudhistira, S.Ds., serta adik penulis Syarafina Aulia dan Gibran Alfarisi, yang telah memberikan penulis inspirasi, dukungan, serta cinta dan kasih sayang penuh dalam menyelesaikan skripsi.
10. Teman seperjuangan penulis Nanda, Hera, Tasya, Qila, Baby, Puti, Alex, yang telah memberikan saran, kontribusi, dan masukan yang berharga kepada penulis.
11. Teman sepermainan penulis Shafira, Dicky, Zura, Nisa yang telah memberikan motivasi penulis untuk menyelesaikan skripsi.
12. Semua rekan seangkatan dalam Program Studi S-1 Ilmu Komputer Universitas Sumatera Utara tahun 2019, terutama teman sekelas dari Kom A, layak mendapat penghargaan atas dukungan dan kebersamaan mereka selama masa perkuliahan penulis.
13. Para pengurus IMILKOM periode 2022/2023 khususnya Departemen Seni dan Olahraga yang telah memberikan dukungan serta pengalaman berharga bagi penulis dalam berorganisasi.
14. Serta semua pihak yang sudah memberi dukungan serta bantuan, yang tidak mungkin disebutkan semuanya.

Semoga Allah SWT. memberikan balasan yang berlipat ganda untuk segala dukungan dan kontribusi yang telah diberikan oleh semua yang turut serta dalam proses penulisan skripsi. Harapannya hasil dari penelitian yang dilakukan akan membawa manfaat yang signifikan serta memberikan sumbangsih positif terhadap kemajuan ilmu pengetahuan, terutama dalam ranah Ilmu Komputer.

Medan, 29 April 2024



Dwi Utami Putri

191401013

ABSTRAK

Fasilitas kesehatan adalah sebuah penyedia pelayanan kesehatan kepada masyarakat yang berperan penting dalam sistem perawatan kesehatan, karena merupakan tempat di mana masyarakat mencari bantuan dan pengobatan untuk masalah kesehatan mereka. Pemilihan fasilitas kesehatan sebagai objek menunjukkan penelitian ini relevan dengan isu kesehatan, yang merupakan aspek penting dalam masyarakat. Penelitian ini ditujukan untuk menganalisis perbandingan antara Algoritma Dijkstra dan Algoritma Floyd Warshall dalam konteks pencarian fasilitas kesehatan di kota Medan. Fokus penelitian ini adalah memahami keefektifan dan keefisienan kedua algoritma dalam mengoptimalkan jarak terpendek menuju fasilitas kesehatan. Algoritma pencarian jarak terpendek, seperti Dijkstra dan Floyd Warshall dapat diterapkan untuk membantu meningkatkan aksesibilitas pelayanan kesehatan bagi masyarakat. Algoritma Dijkstra bersifat "*single-source shortest path*" yang mana mencari jarak terpendek dari satu titik ke semua titik lainnya. Sementara algoritma Floyd Warshall bersifat "*all-pairs shortest path*" yaitu mencari jarak terpendek antar semua pasangan simpul. Dikarenakan jenis graf yang digunakan pada penelitian ini memiliki hubungan terpusat dari titik awal ke titik-titik lainnya, maka penggunaan algoritma Dijkstra lebih efisien dengan kompleksitas waktu $\Theta(n^2)$ dan waktu tempuh 04.75 ms daripada algoritma Floyd Warshall dengan kompleksitas waktu $\Theta(n^3)$ waktu tempuh 08.32 ms.

Kata Kunci: Dijkstra, Floyd Warshall, *Graph Theory*, *Shortest Path*.

ABSTRACT

**COMPARATIVE ANALYSIS OF DIJKSTRA AND FLOYD WARSHALL
ALGORITHMS FOR FINDING SHORTEST PATH ON SEARCHING FOR
HEALTH FACILITY**

Healthcare facilities are providers of health services to the public, playing a vital role in the healthcare system as places where people seek assistance and treatment for their health issues. The choice of healthcare facilities as the subject indicates that this research is relevant to health issues, which are an important aspect of society. This study aims to analyze the comparison between the Dijkstra Algorithm and the Floyd Warshall Algorithm in the context of searching for healthcare facilities in the city of Medan. The focus of this research is to understand the effectiveness and efficiency of both algorithms in optimizing the shortest path to healthcare facilities. Shortest path algorithms, such as Dijkstra and Floyd Warshall, can be applied to help enhance the accessibility of healthcare services for the public. The Dijkstra Algorithm operates as a “single-source shortest path,” which searches for the shortest distance from one point to all other points. Meanwhile, the Floyd Warshall Algorithm operates as an “all-pairs shortest path,” searching for the shortest distance between all pairs of nodes. Due to the centralized nature of the graph used in this study, from the starting point to other points, the use of the Dijkstra Algorithm is more efficient with a time complexity of $\Theta(n^2)$ and a runtime of 04.75 ms, compared to the Floyd Warshall Algorithm with a time complexity of $\Theta(n^3)$ and a runtime of 08.32 ms.

Keywords: Dijkstra, Floyd Warshall, Graph Theory, Shortest Path.

DAFTAR ISI

PERSETUJUAN	iii
PERNYATAAN	iv
PENGHARGAAN	v
ABSTRAK	vii
ABSTRACT	viii
DAFTAR ISI	ix
DAFTAR TABEL	xii
DAFTAR GAMBAR	xiii
BAB I PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Rumusan Masalah	3
1.3. Batasan Masalah	3
1.4. Tujuan Penelitian	4
1.5. Manfaat Penelitian	4
1.6. Metodologi Penelitian	5
1.7. Sistematika Penulisan	5
BAB II LANDASAN TEORI	7
2.1. Teori Graf (<i>Graph Theory</i>)	7
2.2. Jenis-jenis Graf	7
2.3. Jarak Terpendek (<i>Shortest Path</i>)	9
2.4. Algoritma Dijkstra	10
2.5. Algoritma Floyd Warshall	14
2.6. Kompleksitas Algoritma	17
2.7. <i>Running Time</i>	18

2.8. Penelitian yang Relevan	19
BAB III ANALISIS DAN PERANCANGAN SISTEM	20
3.1. Analisis Sistem.....	20
3.1.1. Analisis Masalah	20
3.1.2. Analisis Kebutuhan	21
3.1.3. General Arsitektur	21
3.2. Pemodelan Sistem	22
3.2.1. <i>Use Case Diagram</i>	23
3.2.2. <i>Activity Diagram</i>	23
3.2.3. <i>Sequence Diagram</i>	24
3.3. Flowchart.....	25
3.3.1. Flowchart Sistem.....	25
3.3.2. Flowchart Algoritma Dijkstra	26
3.3.3. Flowchart Algoritma Floyd Warshall	26
3.4. Perancangan Antarmuka (<i>Interface</i>)	28
3.4.1. Rancangan Halaman Utama	28
3.4.2. Rancangan Halaman Hasil Pencarian	29
3.4.3. Rancangan Halaman Rute	29
BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM	31
4.1. Implementasi	31
4.1.1. Halaman Utama.....	31
4.1.2. Halaman Hasil Pencarian	32
4.1.3. Halaman Tampil Rute	33
4.2. Pengujian Sistem.....	33
4.2.1. Pengujian Implementasi Algoritma Dijkstra.....	34
4.2.2. Perhitungan Manual Algoritma Dijkstra.....	35
4.2.3. Pengujian Implementasi Algoritma Floyd Warshall.....	36

4.2.4. Perhitungan Manual Algoritma Floyd Warshall	37
4.3. Hasil Pengujian	41
4.4. Kompleksitas Algoritma	43
4.4.1. Kompleksitas Algoritma Dijkstra	44
4.4.2. Kompleksitas Algoritma Floyd Warshall.....	45
BAB V PENUTUP	47
5.1. KESIMPULAN	47
5.2. SARAN	48
DAFTAR PUSTAKA	49

DAFTAR TABEL

Tabel 2.1 Tabel Iterasi 1 Algoritma Floyd Warshall	15
Tabel 2.2 Tabel Iterasi 2 Algoritma Floyd Warshall	15
Tabel 2.3 Tabel Iterasi 3 Algoritma Floyd Warshall	15
Tabel 2.4 Tabel Iterasi 5 Algoritma Floyd Warshall	16
Tabel 2.5 Tabel Iterasi 5 Algoritma Floyd Warshall	16
Tabel 4.1 Langkah Perhitungan Algoritma Dijkstra	35
Tabel 4.2 Perhitungan Jarak Antar <i>Node</i>	36
Tabel 4.3 Inisialisasi Graf pada Algoritma Floyd Warshall.....	38
Tabel 4.4 Iterasi Perhitungan Algoritma Floyd Warshall	38
Tabel 4.5 Hasil Pengujian Acak Kedua Algoritma	41
Tabel 4.6 Kompleksitas Algoritma Dijkstra	44
Tabel 4.7 Kompleksitas Algoritma Floyd Warshall.....	45

DAFTAR GAMBAR

Gambar 2.1 Graf Sederhana	7
Gambar 2.2 Graf Ganda	8
Gambar 2.3 Graf Semu.....	8
Gambar 2.4 Graf Tak Berarah	8
Gambar 2.5 Graf Berarah	8
Gambar 2.6 Graf Berbobot.....	9
Gambar 2.7 Graf Tak Berbobot.....	9
Gambar 2.8 Jarak Terpendek.....	10
Gambar 2.9 Contoh Graf Algoritma Dijkstra.....	11
Gambar 2.10 Inisialisasi Algoritma Dijkstra.....	11
Gambar 2.11 Langkah 1 Algoritma Dijkstra.....	11
Gambar 2.12 Langkah 2 Algoritma Dijkstra.....	12
Gambar 2.13 Langkah 3 Algoritma Dijkstra.....	12
Gambar 2.14 Langkah 4 Algoritma Dijkstra.....	13
Gambar 2.15 Hasil Perhitungan Algoritma Dijkstra	13
Gambar 2.16 Contoh Graf Algoritma Floyd Warshall	14
Gambar 2.17 Grafik Notasi <i>Big-Omega</i>	17
Gambar 2.18 Grafik Notasi <i>Big-Theta</i>	18
Gambar 2.19 Grafik Notasi <i>Big-O</i>	18
Gambar 3.1 Diagram Ishikawa.....	20
Gambar 3.2 General Arsitektur	22
Gambar 3.3 <i>Use Case Diagram</i>	23
Gambar 3.4 <i>Activity Diagram</i>	24
Gambar 3.5 Sequence Diagram.....	24
Gambar 3.6 Flowchart Sistem	25
Gambar 3.7 Flowchart Algoritma Dijkstra.....	26
Gambar 3.8 Flowchart Algoritma Floyd Warshall.....	27
Gambar 3.9 Rancangan Halaman Utama	28
Gambar 3.10 Rancangan Halaman Hasil Pencarian.....	29
Gambar 3.11 Rancangan Halaman Tampil Rute.....	30
Gambar 4.1 Halaman Utama Aplikasi.....	31

Gambar 4.2 Halaman Hasil Pencarian.....	32
Gambar 4.3 Halaman Tampil Rute.....	33
Gambar 4.4 Hasil Jarak Terpendek Algoritma Dijkstra.....	34
Gambar 4.5 Tampilan Rute Hasil Pencarian	34
Gambar 4. 6 Graf Perhitungan Manual Algoritma Dijkstra	35
Gambar 4.7 Hasil Jarak Terpendek Algoritma Floyd Warshall	37
Gambar 4.8 Graf Perhitungan Manual Algoritma Floyd Warshall	37
Gambar 4.9 Grafik Perbandingan Waktu Tempuh Kedua Algoritma.....	42
Gambar 4.10 Grafik Rata-rata Waktu Tempuh Kedua Algoritma	43

BAB I

PENDAHULUAN

1.1. Latar Belakang

Aspek paling utama dikehidupan adalah kesehatan. Menerapkan pola hidup yang baik adalah cara utama untuk menjaga kesehatan. Selain menerapkan pola hidup yang baik, fasilitas kesehatan juga memegang peranan penting, yaitu sebagai pelayanan dalam usaha pencegahan, pemulihan, maupun penyembuhan (Andresman et al, 2021).

Fasilitas kesehatan adalah tempat yang menyediakan pelayanan kesehatan melalui tindakan preventif, kuratif, maupun rehabilitatif. Akses yang mudah dan cepat ke fasilitas kesehatan merupakan kunci untuk meningkatkan kualitas hidup masyarakat. Dalam kota besar seperti Medan, dengan tingginya jumlah penduduk dan kepadatan, menjadi kritis untuk memastikan aksesibilitas yang optimal ke fasilitas kesehatan. Ditambah lagi ketika dalam keadaan darurat, fasilitas kesehatan terdekatlah yang paling dibutuhkan. Terdapat beberapa jenis fasilitas kesehatan, maka pada penelitian ini penulis melakukan pencarian terhadap fasilitas kesehatan mencakup rumah sakit, puskesmas, praktik dokter, dan klinik.

Algoritma Dijkstra adalah salah satu teknik untuk menentukan jarak terpendek untuk graf berbobot dan digunakan untuk memecahkan masalah *single-source shortest path* (Mukhlif & Saif, 2020). Langkah pertama yaitu menentukan titik awal sebagai titik sumber dan tetapkan sebagai titik yg belum dikunjungi. Atur jarak ke titik awal sebagai 0. Dalam setiap iterasi, algoritma Dijkstra memilih titik dengan jarak terpendek yang belum dikunjungi. Kemudian, jarak ke titik-titik terhubung diperiksa dan diperbarui jika ditemukan jarak yang lebih pendek. Langkah ini berulang hingga semua titik dikunjungi atau mencapai titik tujuan (Pazil et al, 2020).

Pada penelitian sebelumnya oleh Behun et al. (2022), mengatakan bahwa algoritma Dijkstra dapat memindai semua jalur dan memilih jalur dengan jarak minimum secara optimal. Algoritma Dijkstra juga telah digunakan untuk menemukan rute terpendek dan terbaik untuk mengatur penerbangan antar kota dan

dalam waktu yang lebih singkat (Salem et al, 2022). Algoritma Dijkstra efektif dalam menghasilkan jarak terpendek dari titik asal ke titik lainnya secara relatif cepat. Namun, kelemahannya terletak pada ketidakefisienannya dalam menangani graf yang luas dengan banyak simpul dan sisi, serta membutuhkan memori penyimpanan yang signifikan untuk menyimpan jarak terpendek ke setiap titik.

Algoritma Floyd Warshall, sebagaimana dijelaskan oleh Dermawan (2019), merupakan metode pemrograman dinamis yang digunakan untuk menemukan jarak terpendek. Algoritma ini memproses *input* berupa graf berarah dan berbobot untuk menghasilkan bobot minimum setiap jalur antar semua pasangan simpul sebagai *output*, dilakukan secara serentak untuk semua pasangan (Nawagusti, 2018). Dalam menyelesaikan masalah, solusi yang diperoleh dianggap sebagai serangkaian keputusan yang terkait satu sama lain, di mana setiap solusi dibangun berdasarkan solusi dari tahap sebelumnya dan memungkinkan adanya berbagai solusi alternatif (Mukti, 2018).

Penelitian terdahulu oleh Musdalifah et al. (2018) membuktikan bahwa algoritma Floyd Warshall terbukti efektif dan efisien dalam meningkatkan jaringan distribusi beras dari petani ke konsumen. Penerapan algoritma Floyd Warshall menyediakan pilihan rute yang lebih optimal dan hemat waktu melalui perhitungan jarak terpendek, memungkinkan penentuan jalur yang optimal (Ridwan & Agustin, 2020). Keunggulan algoritma Floyd Warshall ini ialah implementasinya yang mudah dan efektif. Namun kelemahannya yaitu memiliki kompleksitas waktu yang relatif tinggi sehingga tidak cocok untuk menghitung data dalam jumlah besar (Furen & Yufang, 2021).

Mengacu pada latar belakang dan studi terdahulu yang telah dijelaskan, penulis mengajukan penelitian yang bertujuan untuk menganalisis dan membandingkan algoritma Dijkstra dengan algoritma Floyd Warshall dalam menemukan jarak terpendek ke fasilitas kesehatan di kota Medan. Fasilitas kesehatan yang hendak dicari meliputi rumah sakit, puskesmas, praktik dokter, dan klinik. Pada penelitian ini penulis juga menyesuaikan posisi *user* sebagai titik awal.

1.2. Rumusan Masalah

Rumusan masalah yang menjadi pusat kajian pada penelitian ini ialah kurangnya informasi mengenai lokasi fasilitas kesehatan terdekat di kota Medan serta sulitnya menentukan jarak terpendek dari lokasi awal *user* ke lokasi fasilitas kesehatan terdekat. Hal ini tentu saja membutuhkan waktu untuk mendapatkan informasi secara manual. Bahkan akan menyulitkan ketika seseorang sedang membutuhkan pertolongan darurat dan segera memerlukan tindakan medis dari fasilitas kesehatan terdekat.

1.3. Batasan Masalah

Batasan masalah yang ditetapkan untuk penulisan ini ialah :

1. Algoritma Dijkstra dan algoritma Floyd Warshall ialah metode yang diterapkan untuk menghitung jarak terpendek di fasilitas kesehatan yang berlokasi di kota Medan.
2. Jenis graf yang diterapkan merupakan graf berarah dan berbobot, di mana bobot tersebut mewakili jarak antara lokasi-lokasi.
3. Pencarian jarak terpendek hanya berlaku pada beberapa lokasi fasilitas kesehatan di kota Medan, terdapat dua puluh lokasi fasilitas kesehatan yang meliputi lima rumah sakit, lima puskesmas, lima praktik dokter, dan lima klinik. Daftar nama fasilitas kesehatan yang menjadi objek pada penelitian ini yaitu sebagai berikut :
 - a. RS Dr. Pirngadi
 - b. RS Imelda Pekerja Indonesia
 - c. RS USU
 - d. RS Columbia Asia Medan
 - e. RS Bunda Thamrin
 - f. Puskesmas Polonia
 - g. Puskesmas Medan Johor
 - h. Puskesmas Sentosa Baru
 - i. Puskesmas Kota Matsum
 - j. Puskesmas Padang Bulan
 - k. Praktik Dokter Sutanto AH

- l. Praktik Dokter Dr. Nining Julie Astuty
 - m. Praktik Dokter Dr. Basri Wijaya
 - n. Praktik Dokter Dr. Raharjo Suparto
 - o. Praktik Dokter Dr. H. Wahjudi
 - p. Klinik Millenium
 - q. Klinik Laisya
 - r. Klinik Mitra Sehat
 - s. Klinik Griya Medical Centre
 - t. Klinik Adinda
4. Parameter yang digunakan untuk membandingkan kinerja algoritma Dijkstra dan algoritma Floyd Warshall adalah jarak terpendek (km), waktu tempuh (*ms*), dan kompleksitas algoritma (*Big Θ*).
 5. Sistem tidak mempertimbangkan jenis keadaan darurat, jam operasional fasilitas kesehatan, kuota pasien fasilitas kesehatan, waktu dan kecepatan alat transportasi, kondisi lalu lintas dari lokasi awal ke lokasi tujuan, serta kondisi jalan disesuaikan dengan kondisi sebenarnya.
 6. Aplikasi *mobile* yang dikembangkan menggunakan *framework* Flutter dan dibangun dengan bahasa pemrograman Dart.

1.4. Tujuan Penelitian

Berikut merupakan beberapa sasaran yang diharapkan dapat tercapai melalui studi ini:

1. Menetapkan jarak terpendek antar fasilitas kesehatan di kota Medan menggunakan algoritma Dijkstra dan algoritma Floyd Warshall.
2. Membandingkan performa berupa waktu tempuh (*ms*) dan kompleksitas algoritma (*Big Θ*) antara algoritma Dijkstra dan algoritma Floyd Warshall.

1.5. Manfaat Penelitian

Diharapkan, hasil dari penelitian ini akan memungkinkan pengguna untuk menemukan jarak terpendek ke fasilitas kesehatan di kota Medan dan memahami perbedaan antara algoritma Dijkstra dan Floyd Warshall dalam menentukan jarak terpendek.

1.6. Metodologi Penelitian

1. Studi Pustaka

Di bagian ini, penulis melakukan pencarian dan pengumpulan data yang diperlukan untuk penelitian. Data tersebut bisa berupa artikel ilmiah, buku, jurnal, makalah, atau sumber-sumber dari internet yang relevan dengan *graph theory*, jarak terpendek (*shortest path*), waktu tempuh, algoritma Dijkstra, dan algoritma Floyd Warshall.

2. Analisis dan Perancangan Sistem

Pada bagian ini, analisis dilakukan pada kebutuhan penelitian yang sudah direncanakan, yang meliputi pembuatan diagram Ishikawa, diagram *use case*, diagram aktivitas, diagram sekuens, diagram alur, dan desain antarmuka.

3. Implementasi Sistem

Dalam bagian ini, desain sistem yang telah dibuat diimplementasikan sebagai aplikasi *mobile*, menggunakan *framework* Flutter dan bahasa pemrograman Dart.

4. Pengujian Sistem

Dalam bagian ini, sistem yang sudah dikembangkan menjalani pengujian untuk memastikan bahwa hasilnya sesuai dengan tujuan penelitian yang telah ditetapkan.

5. Dokumentasi Sistem

Di bagian ini, proses pembuatan dokumentasi dari analisis hingga pengujian sistem dilakukan, yang nantinya akan disusun menjadi laporan atau skripsi.

1.7. Sistematika Penulisan

Dalam penulisan skripsi ini, penulis menyusun struktur pembahasannya secara lebih teratur dengan mengorganisasi menjadi lima bab yang sistematis, yang meliputi:

BAB 1 PENDAHULUAN

Bagian ini menguraikan latar belakang penelitian, perumusan masalah, batasan masalah, tujuan dari penelitian, keuntungan

yang diharapkan, metodologi yang diterapkan, dan tata letak keseluruhan struktur penulisan skripsi.

BAB 2 LANDASAN TEORI

Bagian ini menguraikan konsep-konsep dasar yang menjadi landasan dalam menjalankan penelitian ini.

BAB 3 ANALISIS DAN PERANCANGAN SISTEM

Bagian ini memaparkan analisis sistem serta desain atau model perancangan dari program yang sedang dikembangkan, termasuk algoritma yang diaplikasikan dalam program tersebut.

BAB 4 IMPLEMENTASI DAN PENGUJIAN SISTEM

Bagian ini mengulas mengenai hasil uji sistem dan pelaksanaan program yang telah direncanakan sebelumnya, serta melakukan analisis terhadap program tersebut.

BAB 5 KESIMPULAN DAN SARAN

Bagian ini menyediakan rangkuman dari semua diskusi yang telah disampaikan, termasuk saran yang ditujukan untuk pembaca atau pengembang untuk penelitian lanjutan.

BAB II

LANDASAN TEORI

2.1. Teori Graf (*Graph Theory*)

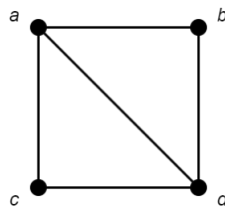
Teori graf adalah cabang matematika yang mempelajari graf, dimana graf merupakan sebuah pasangan yang melibatkan kumpulan objek yang berbentuk simpul (*vertex*) yang tersambung oleh kumpulan sisi (*edge*). Graf ini dinyatakan sebagai $G = (V, E)$, di mana $V = \{v_1, v_2, \dots, v_n\}$ adalah himpunan dari simpul-simpul (*vertices*), dan $E = \{e_1, e_2, \dots, e_n\}$ adalah himpunan dari sisi-sisi (*edges*) yang menghubungkan pasangan simpul.

2.2. Jenis-jenis Graf

Berdasarkan adanya sisi ganda atau *loop* pada sebuah graf, graf dapat diklasifikasikan menjadi dua tipe:

1. Graf Sederhana (*Simple Graph*)

Simple Graph adalah jenis graf di mana setiap pasang simpul terhubung oleh satu sisi saja dan tidak memiliki *loop*. Sebagai contoh, gambar 2.1 menunjukkan graf sederhana dengan simpul $V(G) = \{a, b, c, d\}$.

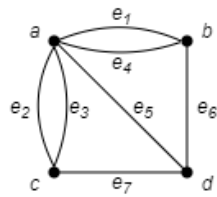


Gambar 2.1 Graf Sederhana

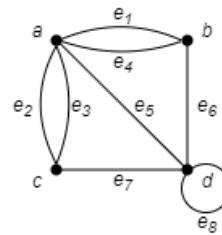
2. Graf Tak Sederhana (*Unsimple Graph*)

Graf tak sederhana yaitu jenis graf yang terdiri dari sisi ganda atau gelang, terbagi menjadi dua jenis: *multigraph*, yang memiliki sisi ganda, dan *pseudograph*, yang termasuk kedua sisi ganda dan gelang. Jika ada beberapa sisi yang menghubungkan dua simpul, maka itu disebut sebagai graf ganda. Sementara itu, graf semu adalah jenis graf yang memiliki sisi gelang, yaitu

sisinya dimulai dan diakhiri pada simpul yang sama. Gambar 2.2 dan gambar 2.3 berikut memberikan bentuk dari graf tak sederhana.



Gambar 2.2 Graf Ganda

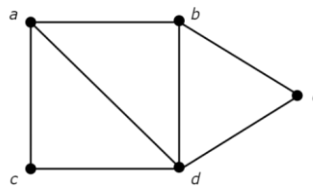


Gambar 2.3 Graf Semu

Graf bisa diklasifikasikan menjadi dua tipe berdasarkan orientasi arah sisinya, yakni:

1. Graf Tak Berarah (*Undirected Graph*)

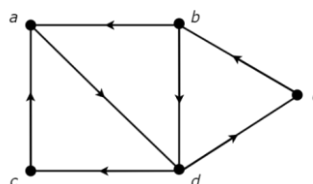
Graf tak berarah ialah jenis graf yang sisinya tak menunjukkan arah, menghubungkan dua simpul tanpa memperhatikan urutan pasangan simpul tersebut (*unordered pairs*), jadi sisi $(a, b) = (b, a)$. Berikut adalah bentuk graf tak berarah yang akan ditunjukkan pada gambar 2.4.



Gambar 2.4 Graf Tak Berarah

2. Graf Berarah (*Directed Graph*)

Graf berarah adalah jenis graf yang setiap sisinya menunjuk ke arah tertentu. Graf berarah biasanya diidentifikasi dengan adanya tanda panah di setiap sisinya sebagai petunjuk simpul awal dan simpul tujuan. Maka, pada graf berarah setiap simpul merupakan pasangan yang terurut, artinya sisi $(a, b) \neq (b, a)$. Gambar 2.5 dibawah ini adalah contoh dari graf berarah.

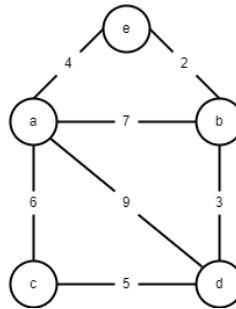


Gambar 2.5 Graf Berarah

Dilihat dari keberadaan bobot pada sisinya, graf diklasifikasikan menjadi dua jenis, yaitu :

1. Graf Berbobot (*Weighted Graph*)

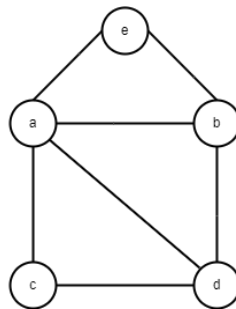
Graf berbobot ialah graf di mana nilai atau bobot ditetapkan pada setiap sisinya. Sebagai ilustrasi, gambar 2.6 menampilkan contoh graf berbobot.



Gambar 2.6 Graf Berbobot

2. Graf Tak Berbobot (*Unweighted Graph*)

Graf tak berbobot merupakan variasi graf dimana sisi-sisinya tidak diberi nilai atau bobot. Bentuk dari graf tak berbobot ditunjukkan pada gambar 2.7.

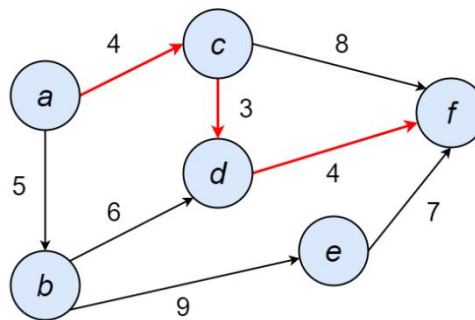


Gambar 2.7 Graf Tak Berbobot

2.3 Jarak Terpendek (*Shortest Path*)

Jarak terpendek adalah jarak paling singkat yang diperlukan untuk berpindah dari satu lokasi ke lokasi lainnya. Masalah pencarian jarak terpendek biasanya direpresentasikan melalui graf. Menurut Rinaldi Munir (2009: 412), graf yang digunakan untuk memecahkan masalah ini adalah graf berbobot (*weighted graph*), di mana istilah "terpendek" umumnya menunjukkan usaha untuk meminimalkan bobot atau panjang lintasan dalam graf.

Contoh dari *shortest path* ditunjukkan pada gambar 2.8 dibawah ini, dimana jarak terpendek pada graf adalah $a-c-d-f$ dengan total bobot 11.



Gambar 2.8 Jarak Terpendek

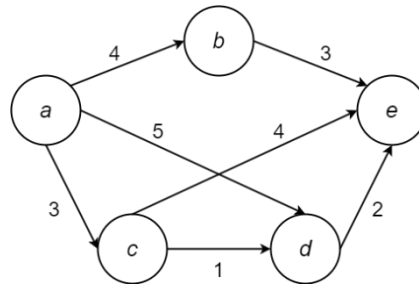
2.4 Algoritma Dijkstra

Algoritma Dijkstra diaplikasikan guna mendapat rute terpendek dari satu titik ke titik lain dalam graf dengan syarat semua bobot harus non-negatif. Bobot dan jalur titik perlu ditukar dengan semua titik untuk menentukan titik mana yang memiliki bobot terkecil. Dijkstra adalah algoritma sederhana dan lugas berdasarkan pendekatan *greedy*. (Ray et al, 2022).

Langkah penyelesaian menggunakan algoritma Dijkstra yakni :

1. Inisialisasi jarak awal untuk semua titik kecuali titik sumber sebagai tak terhingga.
2. Tetapkan jarak awal titik sumber ke dirinya sendiri sebagai 0.
3. Pilih titik dengan jarak terpendek yang belum dikunjungi.
4. Hitung jarak baru dari titik awal ke titik terhubung dengan melalui titik saat ini.
5. Apabila jarak yang baru ditemukan lebih pendek daripada jarak yang telah ada sebelumnya, maka lakukan pembaruan pada jarak tersebut.
6. Lakukan pengulangan langkah 3 hingga 5 sampai setiap titik telah dikunjungi atau tidak ada lagi titik yang terhubung.
7. Jalur terpendek dari titik awal ke setiap titik dapat ditentukan dengan melihat jalur yang disimpan selama proses algoritma.

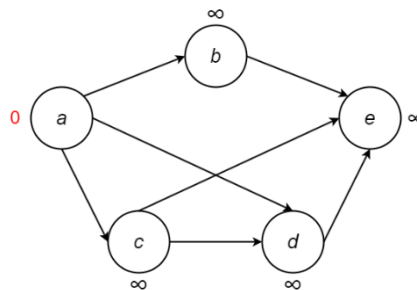
Berikut merupakan contoh perhitungan pencarian jarak terpendek yang dapat diselesaikan dengan menggunakan algoritma Dijkstra :



Gambar 2.9 Contoh Graf Algoritma Dijkstra

Gambar 2.9 diatas ialah graf yang terdiri dari simpul $a-b-c-d-e$ dimana a sebagai titik sumber dan titik lainnya sebagai titik tujuan pada pencarian jarak terpendek.

Inisialisasi:

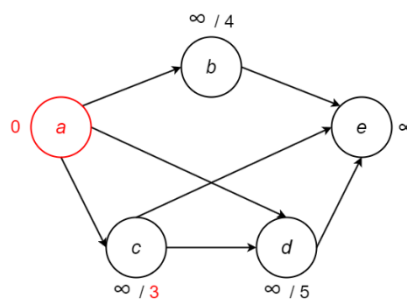


Gambar 2.10 Inisialisasi Algoritma Dijkstra

Inisialisasi jarak ke titik lainnya yang belum dikunjungi dengan tak hingga (∞), dan titik sumber sendiri dengan nilai 0 seperti pada gambar 2.10 diatas.

Langkah 1 :

	a	b	c	d	e
Dari a	0	$\min(\infty, 0+4) = 4$	$\min(\infty, 0+3) = 3$	$\min(\infty, 0+5) = 5$	∞

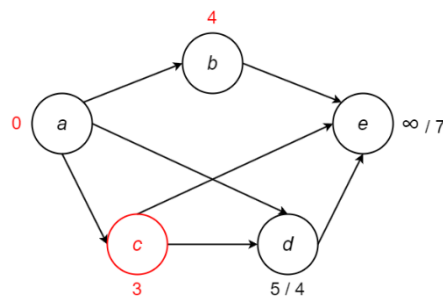


Gambar 2.11 Langkah 1 Algoritma Dijkstra

Gambar 2.11 tersebut ialah langkah pertama pencarian jarak terpendek, yaitu pilih titik a sebagai titik saat ini. Perbarui jarak dari titik a menuju titik tetangga yakni titik b , c , dan d lalu isi titik lainnya yang tidak terhubung (e) dengan tak hingga (∞). Karena jarak yang paling kecil adalah jarak pada titik c , maka beri tanda untuk nilai 3 di titik c .

Langkah 2 :

	a	b	c	d	e
Dari c	0	4	3	$\min(5, 3+1) = 4$	$\min(\infty, 3+4) = 7$

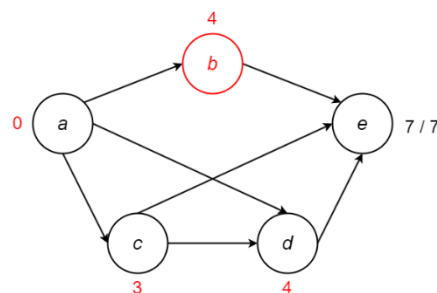


Gambar 2.12 Langkah 2 Algoritma Dijkstra

Gambar 2.12 menunjukkan titik saat ini yaitu titik c yang terhubung ke titik tetangganya, d dan e . Karena nilai jarak titik d dan e saat ini lebih kecil dari sebelumnya, maka perbarui.

Langkah 3 :

	a	b	c	d	e
Dari b	0	4	3	4	$\min(7, 4+3) = 7$

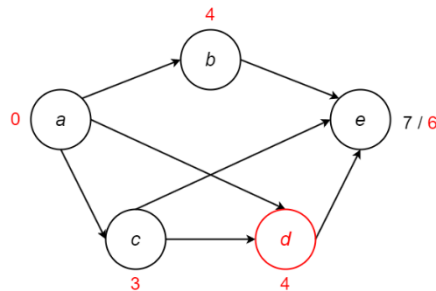


Gambar 2.13 Langkah 3 Algoritma Dijkstra

Langkah selanjutnya dapat dilihat pada gambar 2.13 diatas, dimana titik b sebagai titik saat ini karena sebelumnya merupakan titik dengan jarak terkecil yang belum dikunjungi. Selanjutnya perbarui jarak dari titik saat ini ke titik tetangga yaitu e .

Langkah 4 :

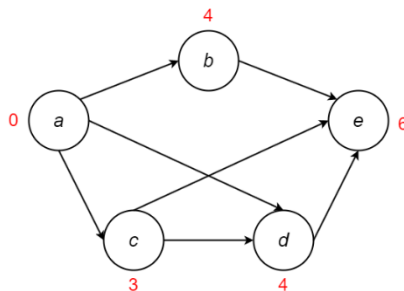
	a	b	c	d	e
Dari d	0	4	3	4	$\min(7, 4+2)=6$



Gambar 2.14 Langkah 4 Algoritma Dijkstra

Dapat dilihat pada gambar 2.14 bahwa titik terakhir yang belum dikunjungi ialah titik d , kemudian perbarui jarak ke titik tetangganya yaitu titik e .

Langkah 5 :



Gambar 2.15 Hasil Perhitungan Algoritma Dijkstra

Gambar 2.15 diatas merupakan hasil dari perhitungan dengan Algoritma Dijkstra, maka proses selesai karena tidak ada titik tetangga yang belum dikunjungi.

Dengan demikian, hasil perhitungannya sebagai berikut:

1. Jarak terpendek dari titik a ke b yakni 4
2. Jarak terpendek dari titik a ke c yakni 3
3. Jarak terpendek dari titik a ke d yakni 4
4. Jarak terpendek dari titik a ke e yakni 6

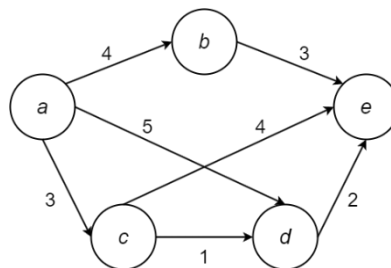
2.5 Algoritma Floyd Warshall

Algoritma Floyd Warshall merupakan teknik populer untuk penentuan jarak terpendek pada graf berarah dan berbobot. Graf ini boleh mencakup bobot negatif pada sisinya, namun siklus dengan bobot negatif tidak diizinkan. Algoritma ini bertujuan mengoptimalkan total bobot yang paling rendah dengan mengkalkulasi bobot terkecil antar setiap pasangan titik (Kamayudi, 2006).

Langkah-langkah penyelesaian algoritma Floyd Warshall untuk mencari jarak terpendek adalah sebagai berikut :

1. Bentuk n matriks sesuai dengan iterasi k , dimana k adalah banyak titik yang dimiliki graf
2. Isi diagonal (\backslash) atau setiap pasangan titik yang sama dengan nilai 0 (misalnya titik a dengan a , b dengan b , dan seterusnya). Sel lainnya diisi dengan tak terhingga (∞)
3. Pasangkan antara garis vertikal dan horizontal yang memiliki nilai kurang dari tak hingga ($<\infty$). Bandingkan nilai hasil penjumlahan pasangan garis dan apabila nilai lebih kecil dari nilai di titik jumpa, maka ganti. Jika tidak, maka nilai tidak berubah.

Berikut adalah contoh penerapan algoritma Floyd Warshall untuk mencari jarak terpendek.



Gambar 2.16 Contoh Graf Algoritma Floyd Warshall

Permasalahan pada gambar 2.16 diatas adalah pencarian jarak terpendek dari titik awal yaitu a ke titik lainnya. Adapun langkah-langkah penyelesaiannya, yaitu:

Iterasi 1 :

Tabel 2.1 Tabel Iterasi 1 Algoritma Floyd Warshall

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>
<i>a</i>	0	4	3	5	∞
<i>b</i>	∞	0	∞	∞	3
<i>c</i>	∞	∞	0	1	4
<i>d</i>	∞	∞	∞	0	2
<i>e</i>	∞	∞	∞	∞	0

Pada iterasi pertama, tidak ditemukan pasangan dengan nilai yang lebih kecil dari tak terhingga ($<\infty$). Sehingga tidak ada nilai yang berubah.

Iterasi 2 :

Tabel 2.2 Tabel Iterasi 2 Algoritma Floyd Warshall

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>
<i>a</i>	0	4	3	5	∞ 7
<i>b</i>	∞	0	∞	∞	3
<i>c</i>	∞	∞	0	1	4
<i>d</i>	∞	∞	∞	0	2
<i>e</i>	∞	∞	∞	∞	0

Pada iterasi kedua, terdapat pasangan dengan nilai kurang dari tak terhingga ($<\infty$), sehingga kedua nilai ditambah lalu nilai pada baris *a* di kolom *e* berubah.

Iterasi 3 :

Tabel 2.3 Tabel Iterasi 3 Algoritma Floyd Warshall

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>
<i>a</i>	0	4	3	5 4	7
<i>b</i>	∞	0	∞	∞	3
<i>c</i>	∞	∞	0	1	4
<i>d</i>	∞	∞	∞	0	2
<i>e</i>	∞	∞	∞	∞	0

Pada iterasi ketiga, terdapat pasangan dengan nilai hasil penjumlahannya lebih kecil dari nilai sebelumnya. Maka nilai pada baris a kolom d berubah.

Iterasi 4 :

Tabel 2.4 Tabel Iterasi 4 Algoritma Floyd Warshall

	a	b	c	d	e
a	0	4	3	4	7 6
b	∞	0	∞	∞	3
c	∞	∞	0	1	4 3
d	∞	∞	∞	0	2
e	∞	∞	∞	∞	0

Pada iterasi keempat, pasangan baris a dan c , di kolom e menghasilkan nilai penjumlahan lebih kecil. Maka nilai pada baris a dan c kolom e berubah.

Iterasi 5 :

Tabel 2.5 Tabel Iterasi 5 Algoritma Floyd Warshall

	a	b	c	d	e
a	0	4	3	4	6
b	∞	0	∞	∞	3
c	∞	∞	0	1	3
d	∞	∞	∞	0	2
e	∞	∞	∞	∞	0

Pada iterasi kelima, tidak ada pasangan dengan nilai yang lebih kecil dari tak terhingga ($<\infty$). Sehingga tidak ada nilai yang berubah. Iterasi selesai dan didapatkanlah bobot terkecil dari semua titik.

Berdasarkan iterasi yang telah dilakukan di atas, maka didapatkan bobot hasil sebagai berikut:

1. Jarak terpendek dari titik a ke b yakni 4
2. Jarak terpendek dari titik a ke c yakni 3
3. Jarak terpendek dari titik a ke d yakni 4
4. Jarak terpendek dari titik a ke e yakni 6

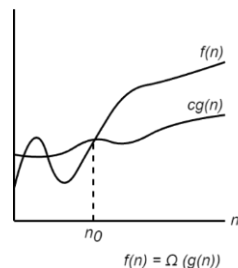
2.6 Kompleksitas Algoritma

Kompleksitas algoritma mengukur jumlah komputasi yang diperlukan oleh algoritma untuk memecahkan sebuah masalah. Algoritma yang ideal tidak hanya perlu memberikan solusi yang benar, tetapi juga harus efisien dalam hal sumber daya yang digunakan. Algoritma dapat dikatakan efisien apabila waktu dan ruang yang dibutuhkan ialah minimum. Performa algoritma diukur berdasarkan kebutuhan waktu dan ruang yang diperlukan untuk memproses jumlah data masukan yang diberikan, biasanya disimbolkan dengan n . Semakin besar nilai n , maka semakin besar pula ukuran waktu dan ruang yang dibutuhkan. Kompleksitas waktu $T(n)$ menilai jumlah operasi yang dibutuhkan oleh sebuah algoritma relatif terhadap ukuran input n . Sementara itu, kompleksitas ruang $S(n)$ adalah pengukuran terhadap jumlah total memori yang diperlukan oleh algoritma berdasarkan ukuran input n (Munir, 2009).

Kinerja sebuah algoritma akan tampak apabila ukuran n yang diproses besar. Oleh karena itu, diperlukan notasi kompleksitas algoritma yang memperlihatkan kinerja algoritma untuk n yang besar. Notasi tersebut dinamakan dengan kompleksitas waktu asimptotik. Kompleksitas waktu asimptotik terbagi menjadi tiga jenis, yakni keadaan terbaik (*best case*) yang direpresentasikan oleh simbol $\Omega(g(n))$ (*Big-Omega*), keadaan rata-rata (*average case*) yang ditandai dengan $\Theta(g(n))$ (*Big-Theta*), dan keadaan terburuk (*worst case*) yang dinyatakan dengan $O(g(n))$ (*Big-O*) (Azizah, 2013).

1. Notasi Ω (*Big-Omega*)

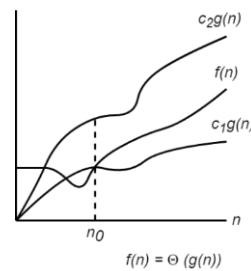
Gambar 2.17 berikut ini menggambarkan notasi Ω sebagai batas bawah dari sebuah fungsi $f(n)$, dimana didefinisikan $f(n) = \Omega(g(n))$ jika ada konstanta positif n_0 dan c , sehingga untuk semua n yang lebih besar atau sama dengan n_0 , $f(n)$ selalu sama dengan atau lebih besar dari $cg(n)$.



Gambar 2.17 Grafik Notasi *Big-Omega*

2. Notasi Θ (*Big-Theta*)

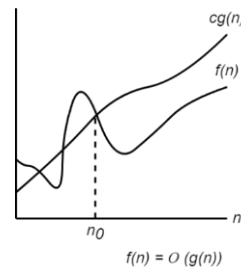
Gambar 2.18 di bawah ini menunjukkan notasi Θ yang membatasi suatu fungsi $f(n)$ agar berada dalam batas faktor konstan. Dinyatakan bahwa $f(n) = \Theta(g(n))$ ketika ada konstanta positif n_0 , c_1 , dan c_2 , sehingga untuk semua n yang lebih besar atau sama dengan n_0 , nilai $f(n)$ berada tepat di $c_1g(n)$, tepat di $c_2g(n)$, atau di antara $c_1g(n)$ dan $c_2g(n)$.



Gambar 2.18 Grafik Notasi *Big-Theta*

3. Notasi O (*Big-O*)

Pada ilustrasi di gambar 2.19, terlihat bahwa untuk semua nilai n pada tepat dan di sebelah kanan n_0 , nilai fungsi $f(n)$ berada tepat atau di bawah $cg(n)$. $f(n) = O(g(n))$ mengindikasikan bahwa $f(n)$ adalah anggota himpunan $O(g(n))$.



Gambar 2.19 Grafik Notasi *Big-O*

2.7 *Running Time*

Waktu tempuh (*running time*) ialah durasi yang diperlukan sebuah algoritma untuk menyelesaikan sebuah siklus proses dalam program dan merupakan faktor penting dalam menentukan efisiensi dan kinerja dari sebuah program. Oleh karena itu, analisis *running time* sangat penting untuk menilai kinerja suatu program, menentukan seberapa efisien program tersebut beroperasi, dan berusaha untuk menaksir lamanya kebutuhan durasi untuk menyelesaikan tugas yang diberikan.

Running time bisa diukur dengan beberapa cara, seperti menggunakan waktu *wall-clock* atau waktu prosesor, dan dapat dinyatakan dalam beberapa unit, seperti *milisecond*, detik, atau menit. Analisis *running time* juga dapat membantu dalam menentukan *trade-off* antara kecepatan dan memori, memungkinkan para pengembang untuk memilih solusi yang paling sesuai untuk masalah yang dihadapi.

2.8 Penelitian yang Relevan

Ada beberapa studi terdahulu yang berkaitan dengan penelitian yang akan dijalankan, seperti :

1. Pada penelitian oleh (Pazil et al, 2020) dengan judul “*Shortest Path from Bandar Tun Razak to Berjaya Times Square using Dijkstra Algorithm*” dikatakan bahwa algoritma Dijkstra telah berhasil digunakan untuk mengoptimalkan jalur dengan membatasi jalan memutar maksimum dan biaya dari Bandar Tun Razak ke Berjaya Times Square.
2. Pada penelitian oleh (Mukhlif & Saif, 2020) yang berjudul “*Comparative Study On Bellman-Ford And Dijkstra Algorithms*” menghasilkan kesimpulan bahwa algoritma Dijkstra lebih baik dalam menentukan jarak terpendek dan beban lalu lintas yang lebih sedikit.
3. Pada penelitian oleh (Sitompul et al., 2018) dengan judul “Implementasi Algoritma Floyd Warshall dalam Menentukan Jalur Terbaik *Driver Pastifresh.Id*” menghasilkan kesimpulan Algoritma Floyd Warshall dapat memperbaiki matriks jarak untuk menemukan rute terpendek dari PastiFresh ke pelanggan dengan melalui semua titik *vendor* yang ditetapkan. Implementasi algoritma ini telah terbukti menjadi opsi optimal dalam menentukan jalur, membantu para pengemudi PastiFresh.id dalam proses pengiriman.
4. Pada penelitian oleh (Furen & Yufang, 2021) yang berjudul “*Research on New Energy Electric Shared Bus Route Optimization Based on Floyd Algorithm*” menunjukkan bahwa algoritma Floyd Warshall mudah dan efisien untuk optimasi skema rute bus, yang secara efektif dapat mengurangi biaya operasi perusahaan dan meningkatkan daya saingnya.

BAB III

ANALISIS DAN PERANCANGAN SISTEM

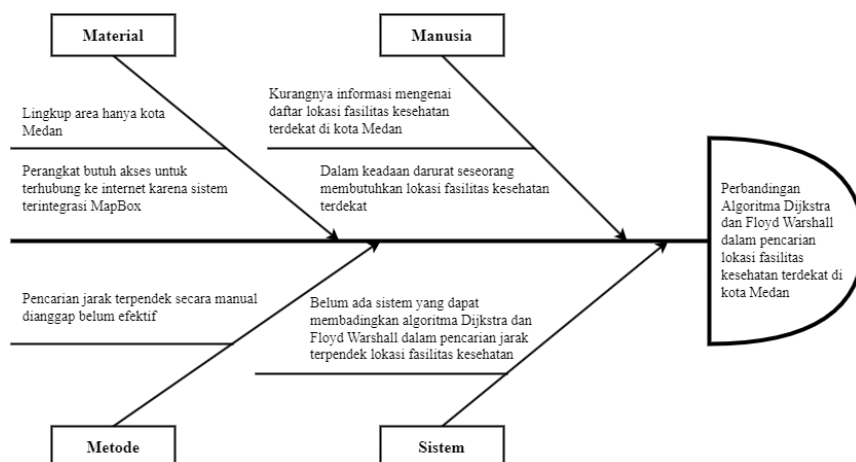
3.1. Analisis Sistem

Fokus dari tahap analisis sistem dalam penelitian yakni mengenali komponen-komponen yang dibutuhkan untuk mengembangkan sistem. Analisis sistem terdiri dari dua aspek utama, yakni analisis masalah dan analisis kebutuhan. Analisis masalah berfokus pada mengidentifikasi serta memahami penyebab suatu masalah serta upaya pencarian solusi untuk mengatasi masalah tersebut. Sebaliknya, analisis kebutuhan bertujuan untuk menentukan data dan proses yang diperlukan dalam mengembangkan sistem yang dirancang.

3.1.1. Analisis Masalah

Ketidakkampuan mendapatkan informasi yang memadai tentang lokasi fasilitas kesehatan terdekat menyulitkan seseorang untuk menentukan jarak terdekat dari lokasi pengguna ke fasilitas kesehatan khususnya di kota Medan. Maka dari itu, penulis bermaksud melakukan penelitian untuk mencari jarak terdekat ke fasilitas kesehatan di kota Medan dengan membandingkan dua algoritma, yaitu algoritma Dijkstra dan Floyd Warshall, dalam proses pencarian jarak terdekat.

Dalam penelitian ini, masalah didefinisikan dengan menggunakan diagram Ishikawa yang digunakan untuk menggambarkan secara visual hubungan sebab-akibat dari suatu masalah. Gambar 3.1 di bawah ini mengilustrasikan akar masalah yang sedang dihadapi dan akan diselesaikan dalam studi ini.



Gambar 3.1 Diagram Ishikawa.

3.1.2. Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk mengenali data dan proses yang dibutuhkan dalam perancangan sistem yang sedang dalam tahap pengembangan. Proses ini terdiri dari dua aspek utama, yaitu kebutuhan fungsional dan kebutuhan non-fungsional.

1. Kebutuhan Fungsional

Operasi-operasi yang harus dilaksanakan oleh sistem yang sedang dalam proses pengembangan termasuk dalam kategori kebutuhan fungsional. Dalam kerangka penelitian ini, kebutuhan fungsional sistem yang akan dibuat yaitu:

- a. Sistem harus dapat membaca lokasi awal pengguna
- b. Sistem harus dapat mengenali 20 titik lokasi fasilitas kesehatan di kota Medan
- c. Sistem harus dapat melakukan pencarian jarak terdekat dengan menggunakan algoritma Dijkstra dan Floyd Warshall untuk mendapatkan hasil pencarian graf.
- d. Sistem harus dapat menampilkan jarak terdekat dan waktu tempuh (*runtime*) dari setiap algoritma.

2. Kebutuhan Non-fungsional

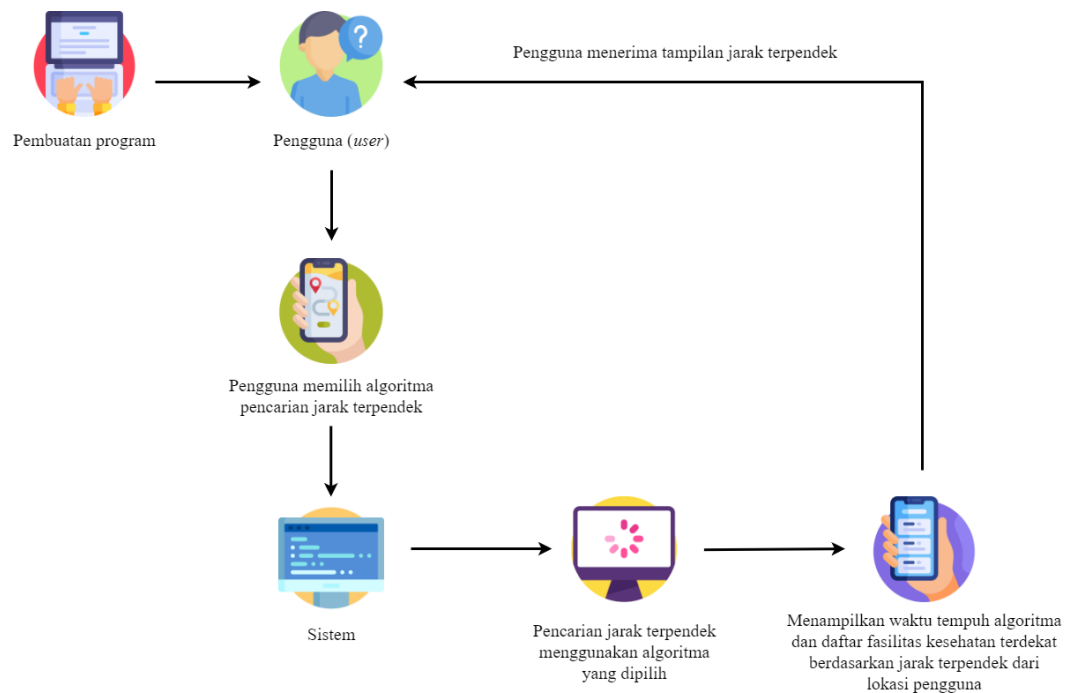
Kebutuhan non-fungsional merujuk pada karakteristik, atribut, kinerja, dan pembatasan dari fungsi yang ditawarkan oleh sistem. Berikut adalah beberapa contoh kebutuhan non-fungsional untuk sistem yang sedang dikembangkan dalam penelitian ini:

- a. Batasan titik pada penelitian ini hanya berada di 20 lokasi fasilitas kesehatan di kota Medan
- b. Tampilan pada sistem *user friendly* sehingga mudah dipahami pengguna.
- c. Tidak membutuhkan biaya yang besar karena tidak memerlukan perangkat tambahan.
- d. Perangkat harus terhubung ke internet untuk dapat menjalankan sistem.

3.1.3. General Arsitektur

General arsitektur adalah sebuah model perancangan atau kerangka kerja yang mendeskripsikan alur dan proses pada sistem, serta menampilkan interaksi antar

komponennya. Gambar 3.2 berikut ini menggambarkan rancangan umum arsitektur sistem yang digunakan dalam penelitian ini.



Gambar 3.2 General Arsitektur

Penjelasan proses general arsitektur sistem yang telah dijabarkan pada gambar 3.2 diatas, yaitu :

1. Pengguna memilih antara algoritma Dijkstra atau algoritma Floyd Warshall sebagai metode pencarian jarak terdekat untuk menemukan fasilitas kesehatan yang paling dekat dari lokasi pengguna.
2. Sistem kemudian memproses pencarian jarak terdekat menggunakan algoritma yang dipilih.
3. Setelah proses selesai dilakukan, sistem akan menampilkan waktu tempuh (ms) berdasarkan algoritma yang dipilih serta menampilkan jarak terdekat lokasi fasilitas kesehatan dari lokasi pengguna.

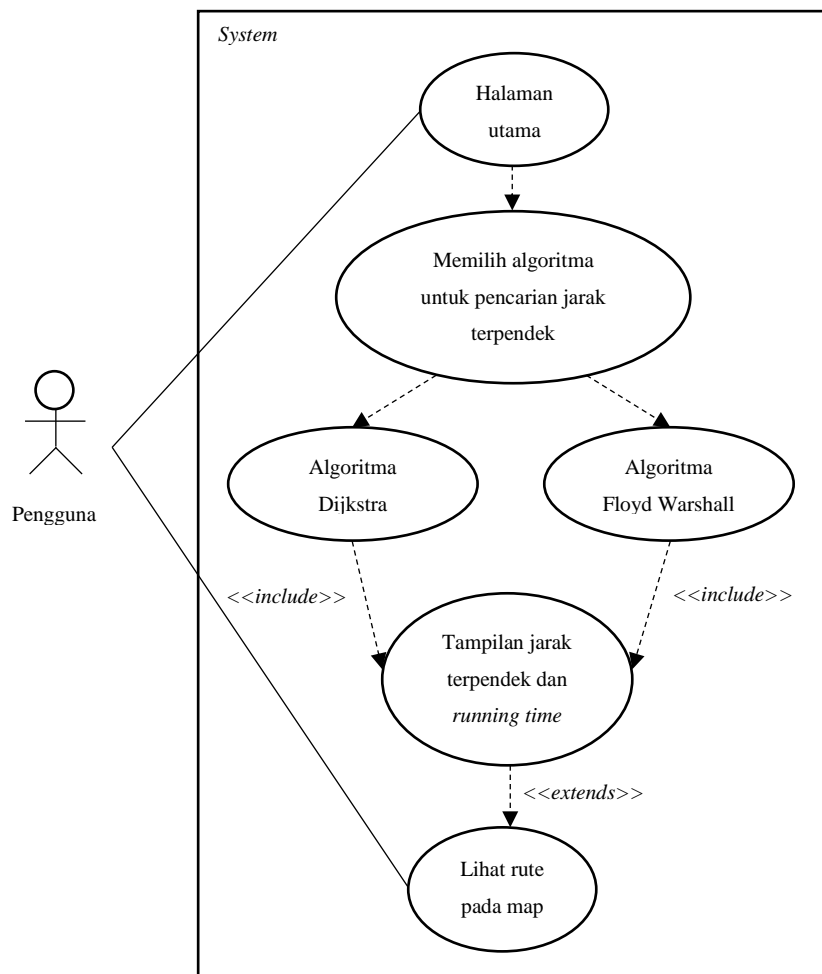
3.2. Pemodelan Sistem

Pemodelan sistem adalah penjabaran interaksi antara pengguna dan sistem. Dalam penelitian ini, UML (*Unified Modelling Language*) digunakan dengan tiga diagram utama, yakni *use case*, *activity*, dan *sequence*. Masing-masing diagram berperan

dalam menggambarkan fungsionalitas, alur kerja, dan interaksi objek secara sekuensial untuk memahami dan merancang sistem.

3.2.1. Use Case Diagram

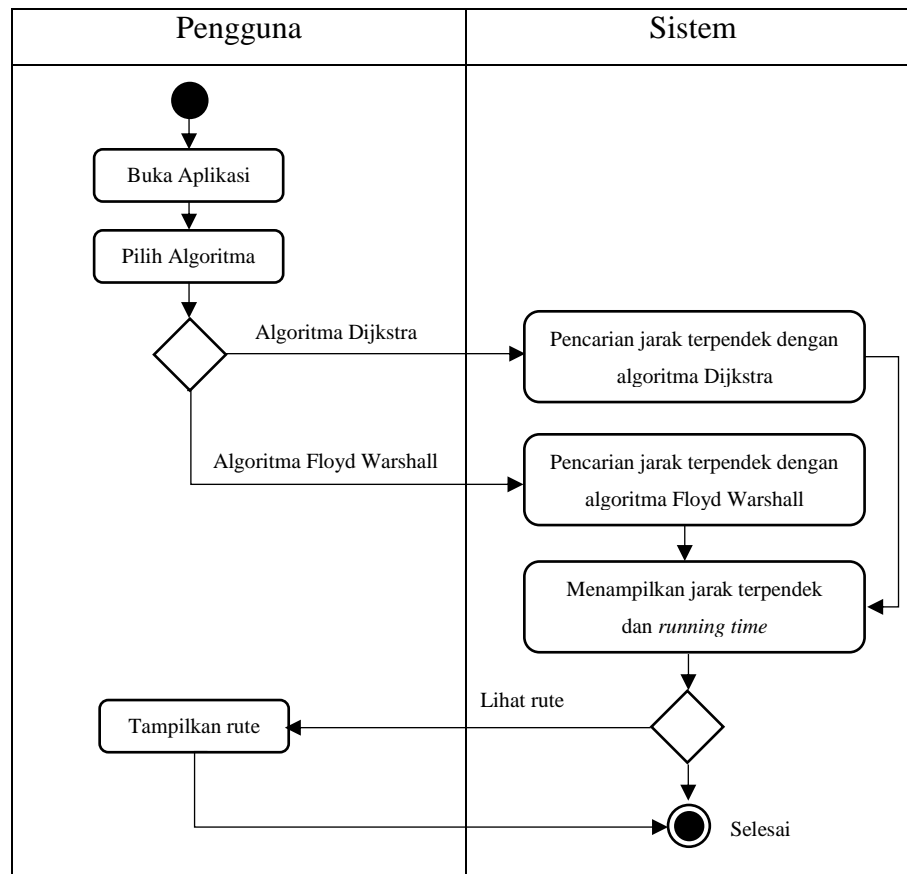
Use case diagram yaitu bentuk pemodelan diagram yang menjelaskan gambaran umum tentang interaksi atau kegiatan utama yang dilakukan oleh pengguna atau sistem itu sendiri. *Use case diagram* dalam pengembangan sistem penelitian ini disajikan dalam gambar 3.3 berikut:



Gambar 3.3 Use Case Diagram

3.2.2. Activity Diagram

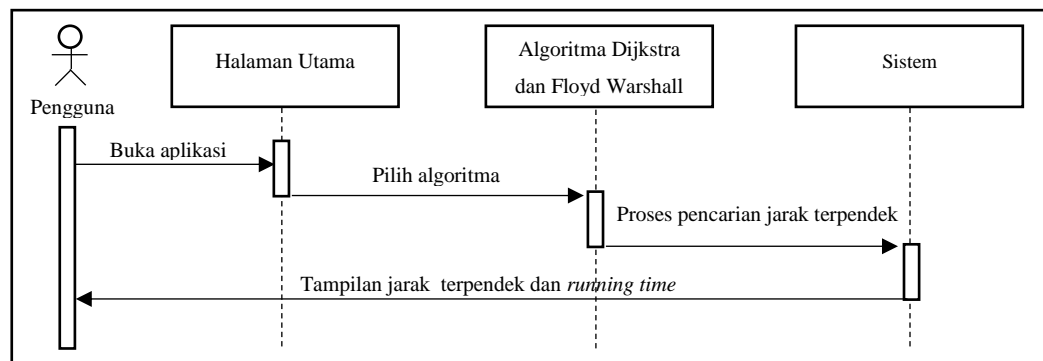
Activity diagram adalah gambaran visual yang menunjukkan aktivitas sistem dari awal hingga selesai. Gambar 3.4 menampilkan diagram aktivitas untuk sistem yang direncanakan dalam penelitian ini.



Gambar 3.4 Activity Diagram

3.2.3. Sequence Diagram

Sequence diagram adalah representasi visual yang menampilkan hubungan antara objek dalam sistem sesuai dengan urutan waktu terjadiannya. Gambar 3.5 menunjukkan *sequence diagram* untuk sistem yang sedang dikembangkan pada studi ini.



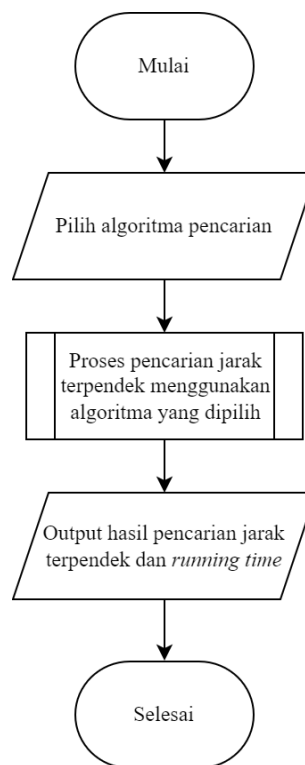
Gambar 3.5 Sequence Diagram

3.3. Flowchart

Diagram alur (*flowchart*) merupakan representasi grafis yang menggunakan simbol-simbol geometris, panah, dan teks untuk memvisualisasikan urutan logika atau proses dalam suatu program.

3.3.1. Flowchart Sistem

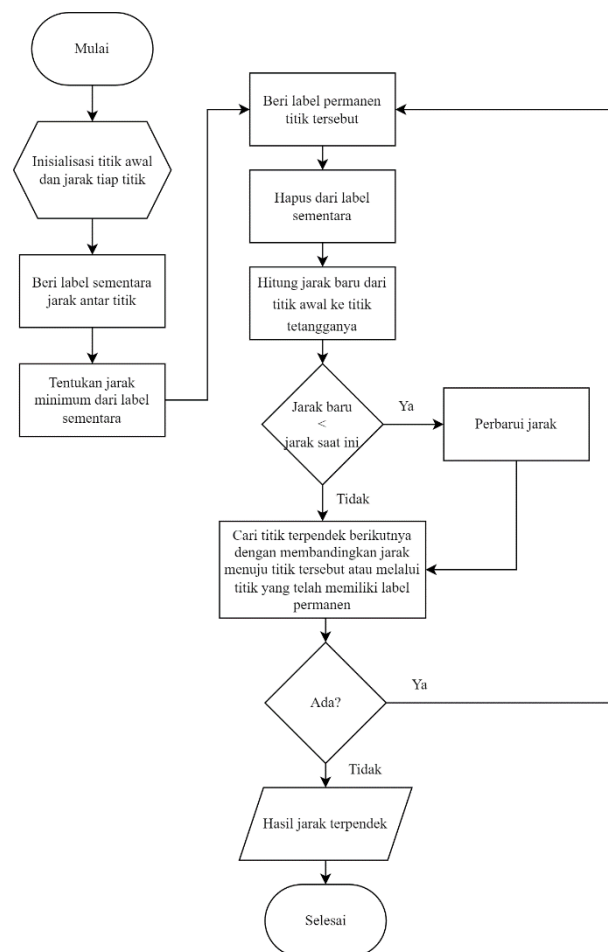
Diagram alur (*flowchart*) yang ditampilkan dalam gambar 3.6 menampilkan proses kerja program aplikasi sistem yang sedang dikembangkan pada penelitian ini. Pertama, pengguna akan memilih algoritma yang akan dipakai untuk menemukan jarak terpendek. Kemudian, sistem akan menggunakan algoritma Dijkstra atau Floyd Warshall untuk melakukan pencarian jarak terpendek ke fasilitas kesehatan terdekat dari lokasi pengguna. Setelah pencarian selesai, hasil jarak terpendek beserta waktu eksekusinya akan ditampilkan oleh sistem.



Gambar 3.6 *Flowchart* Sistem

3.3.2. Flowchart Algoritma Dijkstra

Pada gambar 3.7 dibawah ini, *flowchart* diawali dengan menginisialisasi titik awal dan jarak setiap titik. Selanjutnya beri label sementara titik yang belum dikunjungi. Langkah berikutnya adalah tentukan titik dengan jarak minimum dari label sementara. Beri label permanen pada titik tersebut dan hapus dari label sementara. Hitung jarak baru dari titik awal ke titik tetangganya, perbarui jika ditemukan yang lebih pendek. Cari titik terpendek berikutnya dari titik yang belum dikunjungi, jika tidak ada lagi titik yang belum dikunjungi maka dihasilkan pencarian jarak terpendek dan proses selesai.

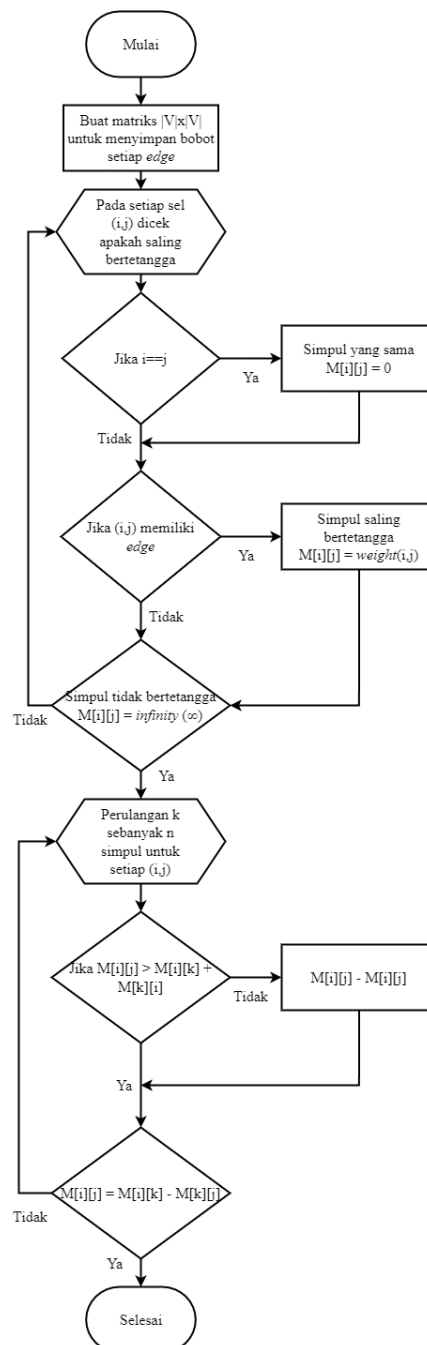


Gambar 3.7 Flowchart Algoritma Dijkstra

3.3.3. Flowchart Algoritma Floyd Warshall

Menurut diagram alur pada gambar 3.8, langkah pencarian jarak terpendek menggunakan algoritma Floyd Warshall dimulai dengan mendefinisikan matriks

berukuran $V \times V$ yang mencerminkan jarak antara setiap simpul. Dalam matriks ini, jika ada koneksi langsung antara dua simpul, bobotnya adalah $\text{weight}(I,j)$. Jika tidak ada koneksi langsung, bobotnya diatur sebagai tak terbatas (∞). Untuk setiap pasangan simpul, dilakukan perhitungan menggunakan rumus $M[i,j] > M[i,k] + M[k,j]$. Jika kondisi ini terpenuhi, nilai $M[i,j]$ akan diperbarui menjadi $M[i,k] + M[k,j]$. Proses ini diulangi dari iterasi k hingga n untuk semua nilai $M[i,j]$ hingga simpul ke- n .



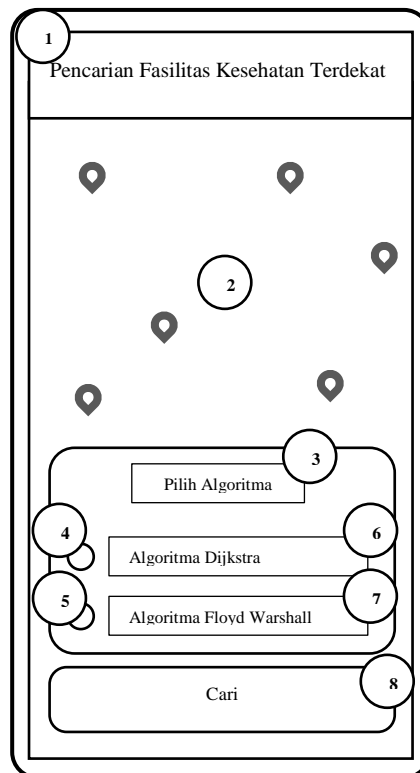
Gambar 3.8 Flowchart Algoritma Floyd Warshall

3.4. Perancangan Antarmuka (*Interface*)

Interface ialah representasi kerangka dasar dari rancangan sistem yang bertindak sebagai platform untuk interaksi antara pengguna dan sistem yang sedang dibuat.

3.4.1. Rancangan Halaman Utama

Gambar 3.9 dibawah ini menggambarkan halaman utama sekaligus tampilan awal ketika pengguna membuka aplikasi.



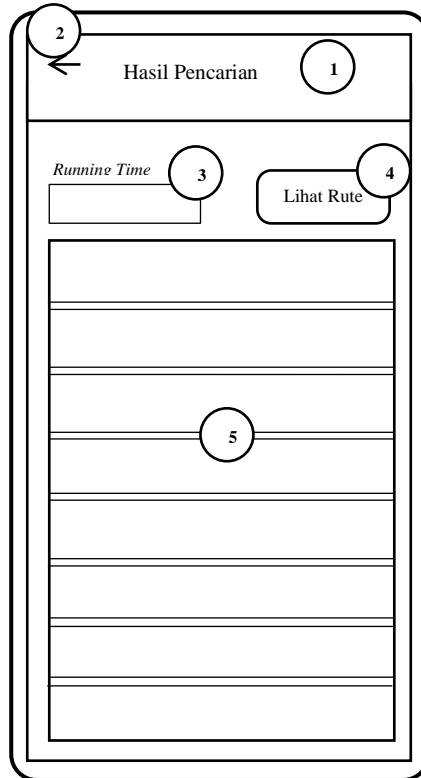
Gambar 3.9 Rancangan Halaman Utama

Penjelasan gambar :

1. *App Bar Title* : Tampilan judul halaman utama
2. *FlutterMap* : Menyediakan map untuk titik-titik fasilitas kesehatan
3. *Text Widget* : Tampilan teks judul untuk memilih algoritma
4. *Radio Button* : Tombol untuk memilih algoritma Dijkstra
5. *Radio Button* : Tombol untuk memilih algoritma Floyd Warshall
6. *Text Widget* : Tampilan teks untuk pilihan Algoritma Dijkstra
7. *Text Widget* : Tampilan teks untuk pilihan Algoritma Floyd Warshall
8. *Button* : Tombol untuk melakukan pencarian jarak terdekat

3.4.2. Rancangan Halaman Hasil Pencarian

Halaman hasil pencarian yang dimuat dalam gambar 3.10 di bawah ini berperan dalam menampilkan hasil pencarian jarak terdekat dan waktu eksekusi dari algoritma yang telah dipilih.



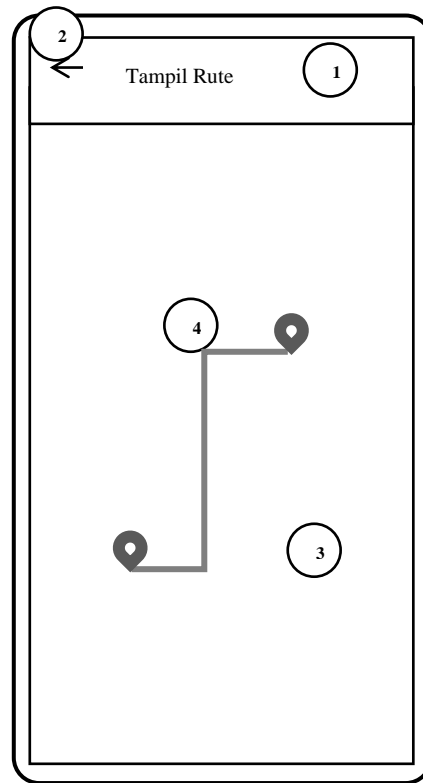
Gambar 3.10 Rancangan Halaman Hasil Pencarian

Penjelasan gambar :

1. *App Bar Title* : Tampilan judul halaman pencarian
2. *Button* : Tombol yang mengarahkan kembali ke halaman utama
3. *Text Widget* : Tampilan hasil *running time*
4. *Button* : Tombol untuk melihat rute di halaman tampil rute
5. *List View* : Tampilan daftar fasilitas kesehatan terdekat beserta jaraknya

3.4.3. Rancangan Halaman Rute

Gambar 3.11 berikut merupakan halaman yang menampilkan rute menuju fasilitas kesehatan terdekat yang sebelumnya telah dihasilkan oleh algoritma pencarian yang dipilih.



Gambar 3.11 Rancangan Halaman Tampil Rute

Penjelasan gambar :

1. *App Bar Title* : Tampilan judul halaman rute
2. *Button* : Tombol yang mengarahkan kembali ke halaman pencarian
3. *Mapbox Map* : Menyediakan map untuk menampilkan *marker* dan rute
4. *LineLayer* : Lapisan garis pada map untuk merepresentasikan rute

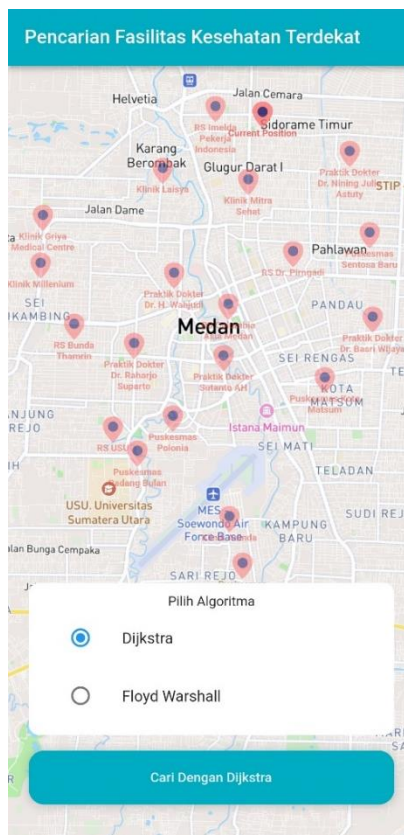
BAB IV

IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1. Implementasi

Penelitian ini mengimplementasikan Algoritma Dijkstra dan Algoritma Floyd Warshall. Bahasa pemrograman Dart dan *framework* Flutter digunakan untuk membangun sistem. Adapun penggunaan map dalam sistem ini direalisasikan melalui API yang ditawarkan oleh Mapbox. Dalam penelitian ini, spesifikasi perangkat yang digunakan untuk mengimplementasikan aplikasi *mobile* ialah Android versi 12. Aplikasi dijalankan melalui Visual Studio Code dan laptop yang memiliki spesifikasi Windows 11 Processor Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz, 2592 Mhz, 6 Core(s), 12 Logical Processor(s).

4.1.1. Halaman Utama



Gambar 4.1 Halaman Utama Aplikasi

Halaman yang ditampilkan pada gambar 4.1 sebelumnya merupakan halaman awal ketika pengguna membuka aplikasi. Halaman ini menampilkan titik-titik fasilitas kesehatan di kota Medan serta titik lokasi pengguna. Terdapat tombol radio untuk memilih algoritma dan tombol cari untuk memulai pencarian jarak terpendek menggunakan algoritma yang dipilih.

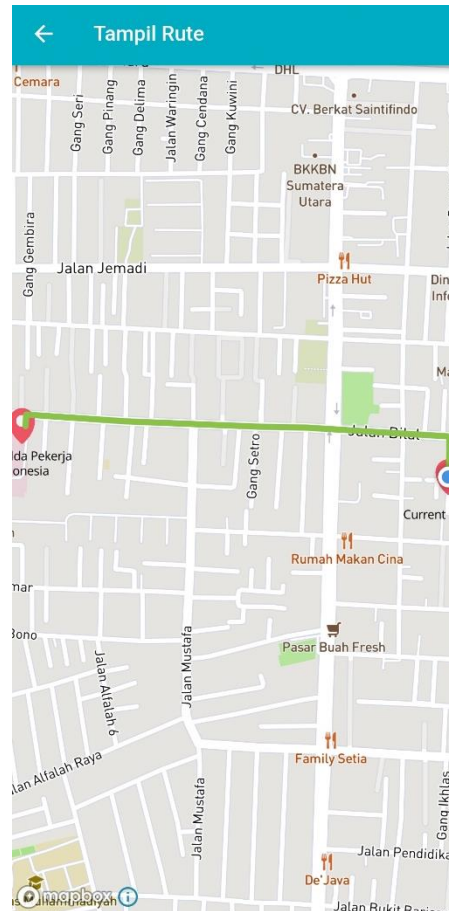
4.1.2. Halaman Hasil Pencarian



Gambar 4.2 Halaman Hasil Pencarian

Gambar 4.2 di atas menampilkan hasil pencarian jarak terpendek dari algoritma yang dipilih. Halaman ini menampilkan *running time* (ms), jarak terpendek fasilitas kesehatan dengan satuan kilometer (km), serta tombol lihat rute untuk mengarahkan ke halaman tampil rute.

4.1.3. Halaman Tampil Rute



Gambar 4.3 Halaman Tampil Rute

Halaman pada gambar 4.3 di atas menampilkan rute menuju fasilitas kesehatan terdekat yang dihasilkan dari pencarian jarak terpendek pada perhitungan algoritma sebelumnya.

4.2. Pengujian Sistem

Proses uji sistem dilaksanakan selepas berhasil mengimplementasikan sistem, dan bertujuan untuk memverifikasi bahwa sistem yang dikembangkan mampu menunjukkan jarak terpendek ke fasilitas kesehatan terdekat. Pada tahap ini, terdapat 20 lokasi fasilitas kesehatan yang digunakan sebagai titik uji. Algoritma Dijkstra dan Floyd Warshall digunakan untuk mencari jarak terpendek dari posisi awal pengguna ke semua lokasi lain dalam graf. Hasil jarak terpendek kemudian ditampilkan dalam bentuk daftar dan diurutkan berdasarkan nilai jaraknya.

4.2.1. Pengujian Implementasi Algoritma Dijkstra



Gambar 4.4 Hasil Jarak Terpendek Algoritma Dijkstra

Hasil pengujian algoritma Dijkstra berupa list daftar fasilitas kesehatan yang telah diurutkan dari yang terdekat dari lokasi pengguna terlihat pada gambar 4.4 di atas. Diperoleh fasilitas kesehatan terdekat dari lokasi pengguna berjarak 1.004 kilometer dan waktu tempuh algoritma mencari jarak terdekat dari titik awal ke titik lainnya ialah 04.75 *milisecond*.

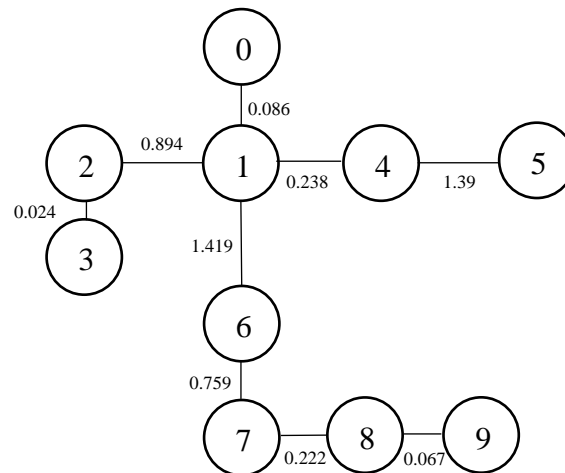
Untuk tampilan rute yang merupakan hubungan *node* dan *edge* dari daftar fasilitas kesehatan terdekat dapat dilihat pada gambar 4.5 dibawah ini.



Gambar 4.5 Tampilan Rute Hasil Pencarian

4.2.2. Perhitungan Manual Algoritma Dijkstra

Pada gambar 4.6, titik awal adalah lokasi pengguna. Dihasilkan lokasi fasilitas kesehatan terdekat pertama dari hasil pencarian yakni RS Imelda Pekerja Indonesia. Pada perhitungan manual ini penulis menghitung jarak dengan sampel 3 titik fasilitas kesehatan terdekat yang dihasilkan pada list.



Gambar 4.6 Graf Perhitungan Manual Algoritma Dijkstra

Gambar 4.6 diatas menampilkan inisialisasi graf dengan *node* awal adalah 0 (lokasi pengguna) dan *node* tujuan adalah 3 (RS Imelda Pekerja Indonesia), 5 (Klinik Mitra Sehat), dan 9 (Praktik Dokter Dr. Julie Astuty).

Selanjutnya melakukan iterasi untuk melakukan perhitungan jarak menggunakan algoritma Dijkstra.

Tabel 4.1 Langkah perhitungan algoritma Dijkstra

Iterasi pertama										
N	0	1	2	3	4	5	6	7	8	9
0	0*	0.086*	∞	∞	∞	∞	∞	∞	∞	∞
Iterasi ke-2										
N	0	1	2	3	4	5	6	7	8	9
0	0*	0.086*	∞	∞	∞	∞	∞	∞	∞	∞
1	0*	∞	0.98	∞	0.324*	∞	1.505	∞	∞	∞
4	0*	∞	∞	∞	∞	1.714	∞	∞	∞	∞
Iterasi ke-3										
N	0	1	2	3	4	5	6	7	8	9
0	0*	0.086*	∞	∞	∞	∞	∞	∞	∞	∞
1	0*	∞	0.98*	∞	0.324*	∞	1.505	∞	∞	∞

4	0*	∞	∞	∞	∞	1.714	∞	∞	∞	∞
2	0*	∞	∞	1.004*	∞	∞	∞	∞	∞	∞
Iterasi ke-4										
N	0	1	2	3	4	5	6	7	8	9
0	0*	0.086*	∞	∞	∞	∞	∞	∞	∞	∞
1	0*	∞	0.98*	∞	0.324*	∞	1.505*	∞	∞	∞
4	0*	∞	∞	∞	∞	1.714*	∞	∞	∞	∞
2	0*	∞	∞	1.004*	∞	∞	∞	∞	∞	∞
6	0*	∞	∞	∞	∞	∞	∞	2.264*	∞	∞
7	0*	∞	∞	∞	∞	∞	∞	∞	2.486*	∞
8	0*	∞	∞	∞	∞	∞	∞	∞	∞	2.553*

Berdasarkan tabel di atas, diperoleh hasil perhitungan jarak sebagai berikut.

Tabel 4.2 Perhitungan Jarak Antar *Node*

dari 0 ke 1 berjarak 0.086
dari 1 ke 2 berjarak 0.98
dari 1 ke 4 berjarak 0.324
dari 4 ke 5 berjarak 1.714
Tujuan ditemukan: Klinik Mitra Sehat pada <i>node</i> 5
dari 2 ke 3 berjarak 1.004
Tujuan ditemukan: RS Imelda Pekerja Indonesia pada <i>node</i> 3
dari 6 ke 7 berjarak 2.264
dari 7 ke 8 berjarak 2.486
dari 8 ke 9 berjarak 2.553
Tujuan ditemukan: Praktik Dokter Dr. Julie Astuty pada <i>node</i> 9

4.2.3. Pengujian Implementasi Algoritma Floyd Warshall

Pengujian untuk algoritma Floyd Warshall dilakukan dengan titik awal berupa lokasi pengguna sama seperti titik awal untuk algoritma Dijkstra. Hal ini dilakukan agar kedua algoritma dapat dibandingkan dengan graf yang sama.

Gambar 4.7 menampilkan hasil dari pengujian yang menerapkan algoritma Floyd Warshall.

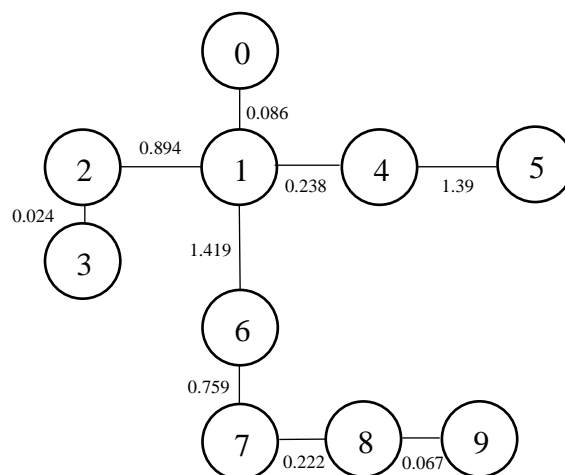


Gambar 4.7 Hasil Jarak Terpendek Algoritma Floyd Warshall

Pada pengujian algoritma Floyd Warshall diatas dihasilkan fasilitas kesehatan terdekat dari lokasi pengguna berjarak 1.004 kilometer dan waktu tempuh algoritma ialah 08.32 *milisecond*.

4.2.4. Perhitungan Manual Algoritma Floyd Warshall

Graf pada gambar 4.8, menggambarkan titik awal adalah lokasi pengguna. Berlokasi pada titik yang sama ketika pengujian dilakukan dengan algoritma Dijkstra. Penggunaan titik awal yang sama dimaksud agar kedua algoritma dapat dibandingkan dengan jumlah *node* dan *weight* yang sama. Dari lokasi awal, didapat tiga lokasi vaksinasi terdekat yakni RS Imelda Pekerja Indonesia, Klinik Mitra Sehat, dan Praktik Dokter Dr. Julie Astuty.



Gambar 4.8 Graf Perhitungan Manual Algoritma Floyd Warshall

Berikut ini adalah urutan proses dalam melakukan perhitungan menggunakan algoritma Floyd Warshall.

Langkah 1

Melakukan inisialisasi graf ke dalam matriks.

Tabel 4.3 Inisialisasi Graf pada Algoritma Floyd Warshall

N	0	1	2	3	4	5	6	7	8	9
0	0	0.086	∞	∞	∞	∞	∞	∞	∞	∞
1	∞	0	0.894	∞	0.238	∞	1.419	∞	∞	∞
2	∞	∞	0	0.024	∞	∞	∞	∞	∞	∞
3	∞	∞	∞	0	∞	∞	∞	∞	∞	∞
4	∞	∞	∞	∞	0	1.39	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	0	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	0	0.759	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	0	0.222	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	0	0.067
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

Langkah 2

Menghitung jarak antar titik dilakukan dengan memeriksa jalur alternatif yang lebih pendek pada setiap iterasi. Untuk setiap pasangan titik (i, j), dicari titik perantara yang bisa mempersingkat jarak antara i dan j. Jika ditemukan, matriks jarak diperbarui dengan nilai yang lebih kecil. Proses ini diulang sampai iterasi N selesai, sehingga matriks jarak akhirnya berisi jarak terpendek antar semua pasangan titik.

Tabel 4.4 Iterasi Perhitungan Algoritma Floyd Warshall

Iterasi pertama N = 0										
N	0	1	2	3	4	5	6	7	8	9
0	0	0.086	∞	∞	∞	∞	∞	∞	∞	∞
1	∞	0	0.894	∞	0.238	∞	1.419	∞	∞	∞
2	∞	∞	0	0.024	∞	∞	∞	∞	∞	∞
3	∞	∞	∞	0	∞	∞	∞	∞	∞	∞
4	∞	∞	∞	∞	0	1.39	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	0	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	0	0.759	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	0	0.222	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	0	0.067
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	0
Iterasi ke-2 N = 1										
N	0	1	2	3	4	5	6	7	8	9
0	0	0.086	∞ 0.98	∞	∞ 0.324	∞	∞ 1.505	∞	∞	∞

3	∞	∞	∞	0	∞	∞	∞	∞	∞	∞
4	∞	∞	∞	∞	0	1.39	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	0	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	0	0.759	0.981	1.048
7	∞	∞	∞	∞	∞	∞	∞	0	0.222	0.289
8	∞	∞	∞	∞	∞	∞	∞	∞	0	0.067
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

Dari hasil perhitungan yang telah dilakukan oleh algoritma Floyd Warshall, maka dari titik 0 (lokasi pengguna) didapat jarak 1.004 km menuju titik 3 (RS Imelda Pekerja Indonesia), 1.714 km menuju titik 5 (Klinik Mitra Sehat), dan 2.553 km menuju titik 9 (Praktik Dokter Dr. Nining Julie Astuty).

4.3. Hasil Pengujian

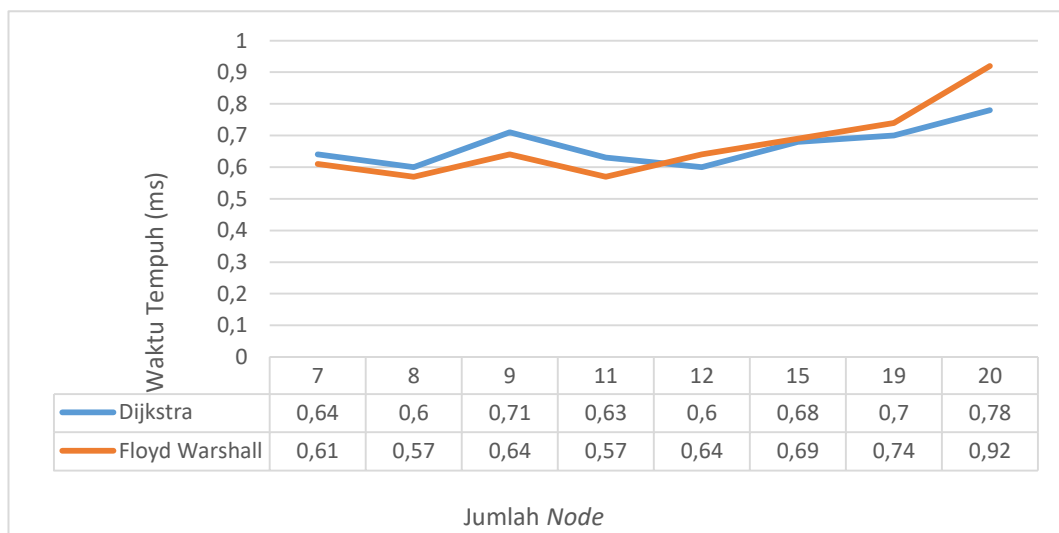
Hasil yang diperoleh dari sistem yang telah diuji menggunakan algoritma Dijkstra dan algoritma Floyd Warshall selanjutnya dibandingkan. Tujuan perbandingan ialah untuk mengevaluasi waktu eksekusi masing-masing algoritma dengan menggunakan titik awal yang sama. Dalam tahap ini, 8 lokasi acak sebagai titik tujuan digunakan untuk membandingkan hasilnya. Rincian hasil pengujian dapat ditemukan dalam tabel berikut.

Tabel 4.5 Hasil Pengujian Acak Kedua Algoritma

No.	Lokasi Tujuan	Jumlah Node	Running Time (ms)	
			Dijkstra	Floyd Warshall
1	Klinik Adinda (3.5526507608177407, 98.67739111007002)	20	0,78	0,92
2	Puskesmas Polonia (3.569910242828818, 98.66766100558456)	15	0,68	0,69
3	RS USU (3.56804486625784, 98.65738826781505)	12	0,60	0,64
4	Puskesmas Medan Johor (3.544625213556921, 98.67967643898007)	19	0,70	0,74

No.	Lokasi Tujuan	Jumlah Node	Running Time (ms)	
			Dijkstra	Floyd Warshall
5	Puskesmas Sentosa Baru (3.601351119322902, 98.70151566977528)	8	0,60	0,57
6	Praktik Dokter Dr. Sutanto AH (3.580219328718221, 98.6763989922032)	9	0,71	0,64
7	Klinik Millenium (3.5961445797857428, 98.64498440706387)	7	0,64	0,61
8	Puskesmas Padang Bulan (3.563857515973795, 98.6616058157139)	11	0,63	0,57
Rata-rata			0,6675	0,6725

Running time atau waktu tempuh dari kedua algoritma sangat bergantung pada kemampuan mesin dalam melakukan kalkulasi. Grafik perbandingan waktu tempuh kedua algoritma dimuat dalam gambar 4.9 dibawah ini.

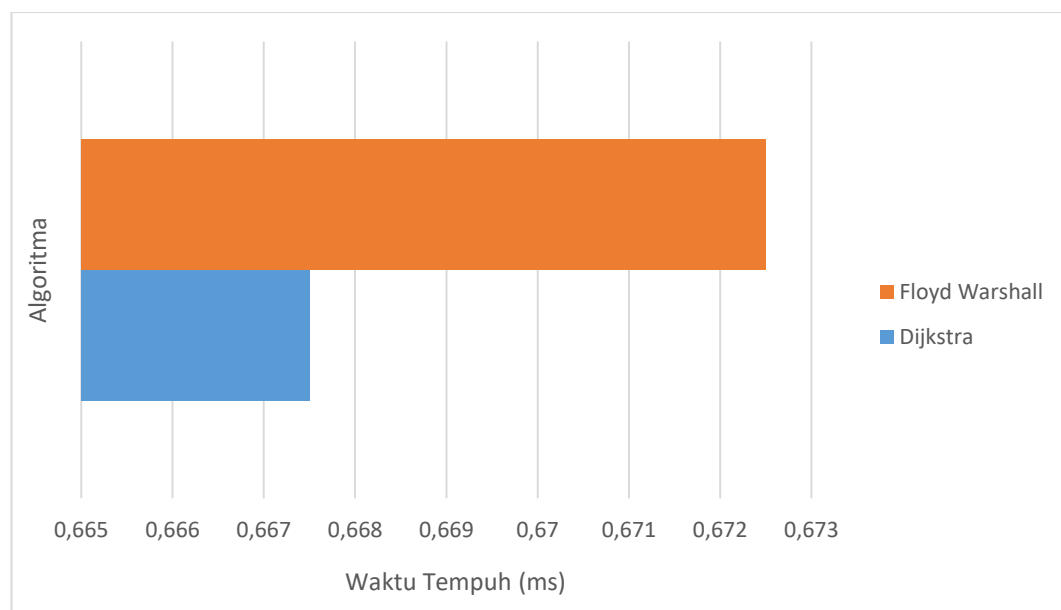


Gambar 4.9 Grafik Perbandingan Waktu Tempuh Kedua Algoritma

Grafik diatas menunjukkan peningkatan *running time* kedua algoritma seiring dengan peningkatan jumlah *node* yang diproses. Penting untuk dicatat bahwa bobot jarak antar *node* tidak berpengaruh pada *running time*, karena waktu tempuh algoritma ditentukan oleh jumlah *node* dalam matriks parameter. Semakin banyak

node yang diambil dari respon API, matriks parameter akan lebih panjang, dan kedua algoritma akan melakukan lebih banyak iterasi. Karena graf pada pengujian ini memiliki struktur yang lebih jarang terhubung atau banyak *node* yang tidak dapat dijangkau dari satu titik awal, algoritma Floyd Warshall mungkin melakukan banyak perhitungan yang tidak perlu. Sehingga untuk jumlah *node* yang banyak, algoritma Floyd Warshall membutuhkan waktu tempuh yang lebih lama daripada algoritma Dijkstra.

Jika dilihat tabel 4.5 di atas diperoleh waktu rata-rata proses algoritma Dijkstra sebesar 0.6675 ms. Sedangkan waktu rata-rata proses algoritma Floyd Warshall sebesar 0.6725 ms. Ini terlihat pada gambar 4.10 berikut. Dengan demikian, dapat disimpulkan bahwa algoritma Dijkstra memiliki kinerja waktu yang lebih efisien dibandingkan dengan algoritma Floyd Warshall.



Gambar 4.10 Grafik Rata-rata Waktu Tempuh Kedua Algoritma

4.4. Kompleksitas Algoritma

Kompleksitas algoritma adalah perkiraan dari durasi yang diperlukan sebuah algoritma untuk menyelesaikan proses perhitungan. Dalam konteks ini, perhatian utama difokuskan pada kompleksitas waktu. Menilai karena pengukuran yang dilakukan oleh mesin bersifat relatif bergantung pada spesifikasi mesin, alokasi memori pada suatu waktu, dan faktor-faktor lainnya, maka kinerja berjalannya

algoritma tidak konsisten dan tergantung oleh faktor-faktor tersebut. Oleh karena itu, menghitung kompleksitas waktu dengan mengukur jumlah operasi untuk setiap perintah dalam algoritma memberikan ukuran yang lebih akurat sebagai acuan tingkat efisiensi algoritma. Untuk mengukur kompleksitas waktu pada kedua algoritma ini, notasi yang digunakan adalah *big Theta* (Θ).

4.4.1. Kompleksitas Algoritma Dijkstra

Tabel di bawah ini menampilkan kompleksitas waktu dari algoritma Dijkstra.

Tabel 4.6 Kompleksitas Algoritma Dijkstra

No	Kode Program	C	#	C*#
1.	Map<String, double> distances = {};	C ₁	1	C ₁
2.	Map<String, String> predecessors = {};	C ₁	1	C ₁
3.	currentGraph.forEach((key, value) {	C ₂	n ²	C ₂ n ²
4.	distances[key] = double.infinity;});	C ₂	n ²	C ₂ n ²
5.	distances[start] = 0;	C ₃	1	C ₃
6.	currentGraph.forEach((u, edges) {	C ₄	n	C ₄ n
7.	edges.forEach((v, weight) {	C ₄	m	C ₄ m
8.	double distance = distances[u]! + weight;	C ₄	1	C ₄
9.	if (distance < distances[v]!) {	C ₄	1	C ₄
10.	distances[v] = distance;	C ₄	1	C ₄
11.	predecessors[v] = u;}}});});	C ₄	1	C ₄
12.	currentGraph.forEach((u, edges) {	C ₅	n	C ₅ n
13.	edges.forEach((v, weight) {	C ₅	m	C ₅ m
14.	double distance = distances[u]! + weight;	C ₅	1	C ₅
15.	if (distance < distances[v]!) {	C ₅	1	C ₅
16.	throw Exception("Graph contains a negative-weight cycle");}}});});	C ₅	1	C ₅
17.	return distances;	C ₆	1	C ₆

Berdasarkan tabel di atas, diperoleh hasil sebagai berikut:

$$T(n) = 2C_1 + 2C_2n^2 + C_3 + C_4n + C_4m + 4C_4 + C_5n + C_5m + 3C_5 + C_6$$

$$T(n) = ((2C_2)*n^2) + ((C_4 + C_5)*n) + ((C_4 + C_5)*m) + ((2C_1 + C_3 + 4C_4 + 3C_5 + C_6)*n^0)$$

$$T(n) = \Theta(n^2)$$

Berdasarkan perhitungan di atas, dihasilkan kompleksitas waktu dari algoritma Dijkstra sebesar $\Theta(n^2)$, dimana n adalah jumlah *node* dan m adalah jumlah *edge* pada graf .

4.4.2. Kompleksitas Algoritma Floyd Warshall

Kompleksitas waktu dari algoritma Floyd Warshall ditampilkan pada tabel dibawah ini.

Tabel 4.7 Kompleksitas Algoritma Floyd Warshall

No.	Kode Program	C	#	C*#
1.	Map<String, Map<String, double>> distances = {};	C ₁	1	C ₁
2.	Set<String> allVertices = Set<String>();	C ₁	1	C ₁
3.	currentGraph.forEach((u, edges) {	C ₂	n	C ₂ n
4.	allVertices.add(u);	C ₂	1	C ₂
5.	edges.forEach((v, weight) {	C ₃	m	C ₃ m
6.	allVertices.add(v);});});	C ₃	1	C ₃
7.	allVertices.forEach((u) {	C ₄	n	C ₄ n
8.	distances[u] = {};	C ₄	1	C ₄
9.	allVertices.forEach((v) {	C ₅	n ²	C ₅ n ²
10.	distances[u]![v] = double.infinity; });	C ₅	n ²	C ₅ n ²
11.	distances[u]![u] = 0; });	C ₅	1	C ₅
12.	currentGraph.forEach((u, edges) {	C ₆	n	C ₆ n
13.	edges.forEach((v, weight) {	C ₆	m	C ₆ m
14.	distances[u]![v] = weight; });});	C ₆	1	C ₆
15.	allVertices.forEach((k) {	C ₇	n ³	C ₇ n ³
16.	allVertices.forEach((i) {	C ₇	n ³	C ₇ n ³
17.	allVertices.forEach((j) {	C ₇	n ³	C ₇ n ³

No.	Kode Program	C	#	C*#
18.	if (distances[i][k] + distances[k][j] < distances[i][j]) {	C_7	1	C_7
19.	distances[i][j] = distances[i][k] + distances[k][j]; } } } } };	C_7	1	C_7
20.	allVertices.forEach((u) {	C_8	n	C_8n
21.	if (distances[u][u] < 0) {	C_8	1	C_8
22.	throw Exception("Graph contains a negative-weight cycle"); } } };	C_8	1	C_8
23.	return distances[start];	C_9	1	C_9

Hasil yang diperoleh dari tabel diatas adalah:

$$T(n) = 2C_1 + C_2n + C_2 + C_3m + C_3 + C_4n + C_4 + 2C_5n^2 + C_5 + C_6n + C_6m + C_6 + 3C_7n^3 + 2C_7 + C_8n + 2C_8 + C_9$$

$$T(n) = ((3C_7)n^3) + ((2C_5)n^2) + ((C_2 + C_4 + C_6 + C_8)n) + ((C_3 + C_6)m) + ((2C_1 + C_2 + C_3 + C_4 + C_5 + C_6 + 2C_7 + 2C_8 + C_9)n^0)$$

$$T(n) = \Theta(n^3)$$

Berdasarkan perhitungan di atas, diperoleh kompleksitas waktu dari algoritma Floyd Warshall sebesar $\Theta(n^3)$. Dengan demikian, algoritma Floyd Warshall menghasilkan kompleksitas waktu lebih tinggi daripada algoritma Dijkstra, yang memiliki kompleksitas waktu sekitar $\Theta(n^2)$.

BAB V

PENUTUP

5.1. KESIMPULAN

Ini adalah hasil akhir yang dipetik melalui evaluasi, perancangan, implementasi, dan pengujian algoritma Dijkstra serta Floyd Warshall pada pencarian jarak terpendek ke fasilitas kesehatan di kota Medan.

1. Algoritma Dijkstra dan algoritma Floyd Warshall terbukti efektif dan bisa diterapkan untuk mencari lokasi fasilitas kesehatan terdekat di kota Medan.
2. Waktu tempuh masing-masing algoritma dipengaruhi oleh jumlah *node*.
3. Algoritma Dijkstra memiliki kompleksitas waktu $\Theta(n^2)$ dan waktu tempuh 04.75 ms, sedangkan algoritma Floyd Warshall memiliki kompleksitas waktu $\Theta(n^3)$ dan waktu tempuh 08.32 ms.
4. Algoritma Dijkstra bersifat "*single-source shortest path*" yang mana fokus pada pencarian jarak terpendek dari satu titik ke semua titik lainnya. Sementara algoritma Floyd Warshall bersifat "*all-pairs shortest path*" yaitu mencari jarak terpendek antara semua pasangan simpul. Dikarenakan jenis graf yang digunakan pada penelitian ini memiliki hubungan terpusat dari titik awal ke titik-titik lainnya, maka penggunaan algoritma Dijkstra lebih efisien daripada algoritma Floyd Warshall.

5.2. SARAN

Berikut beberapa saran yang bisa dipertimbangkan dalam studi di masa mendatang.

1. Sistem dapat menampilkan rute untuk daftar fasilitas kesehatan yang lainnya, bukan hanya yang paling terdekat saja.
2. Melibatkan opsi rute alternatif dan variasi rute sesuai dengan jenis transportasi, dengan memanfaatkan fitur API yang tersedia di Mapbox.
3. Aplikasi dapat dikembangkan untuk wilayah yang lebih luas dan kasus berbeda.

DAFTAR PUSTAKA

- Adresman, F. J., Triyanto, D., & Hidayati, R. Penerapan Teknik Location Based Service dan Algoritma Floyd-Warshall pada Aplikasi Pencarian Fasilitas Kesehatan di Kota Pontianak. *Coding Jurnal Komputer dan Aplikasi*, 9(01), 142-151.
- Azizah, U. N. (2013). Perbandingan Detektor Tepi Prewitt dan Detektor Tepi Laplacian berdasarkan Kompleksitas Waktu dan Citra Hasil (Doctoral dissertation, Universitas Pendidikan Indonesia).
- Behún, M., Knežo, D., Cehlár, M., Knapčíková, L., & Behúnová, A. (2022). Recent Application of Dijkstra's Algorithm in the Process of Production Planning. *Applied Sciences*, 12(14), 7088.
- Dermawan, T. S. (2019). Comparison of Dijkstra dan Floyd-Warshall Algorithm to Determine the Best Route of Train. *IJID (International Journal on Informatics for Development)*, 7(2), 54-58.
- Furen, W., & Yufang, W. (2021, June). Research on new energy electric shared bus route optimization based on floyd algorithm. In *IOP Conference Series: Earth and Environmental Science* (Vol. 766, No. 1, p. 012069). IOP Publishing.
- Kamayudi, A. (2006). Studi dan Implementasi Algoritma Djikstra, Bellman-Ford dan Floyd-Warshall dalam menangani masalah lintasan terpendek dalam Graf. Program Studi Teknik Informatika, Institut Teknologi Bandung.
- Mukhlif, F., & Saif, A. (2020, February). Comparative study on Bellman-Ford and Dijkstra algorithms. In *Int. Conf. Comm. Electric Comp. Net*.
- Mukti, M. R. (2018). Menentukan Rute Terpendek dengan Menggunakan Algoritma Floyd-Warshall dalam Pendistribusian Barang pada PT. Rapy Ray Putratama (Doctoral Dissertation, Unimed).
- Munir, R. (2009). *Matematika Diskrit Edisi Ketiga*. Informatika, Bandung.

- Muzdalifah, L., Oktafianto, K., & Mustika, E. D. (2018). Model Jaringan Distribusi Beras Optimal Menggunakan Algoritma Floyd Warshall. *Jurnal Riset dan Aplikasi Matematika (JRAM)*, 2(2), 101-111.
- Nawagusti, V. A. (2018). Penerapan Algoritma Floyd Warshall dalam Aplikasi Penentuan Rute Terpendek Mencari Lokasi BTS (Base Tower Station) pada PT. GCI Palembang. *Jurnal Nasional Teknologi dan Sistem Informasi*, 4(2), 81-88.
- Pazil, N. S. M., Mahmud, N., & Jamaluddin, S. H. (2020). Shortest Path from Bandar Tun Razak to Berjaya Times Square using Dijkstra Algorithm. *Journal of Computing Research and Innovation*, 5(4), 61-67.
- Ray, A., Sharma, H., & Sharma, D. (2022). Analysis and Design of Public Transport Route Planner: Dijkstras Algorithm. *Analysis and Design of Public Transport Route Planner: Dijkstras Algorithm*, 10(VI), 4571-75.
- Ridwan, F., & Agustin, R. D. (2020). Penggunaan Algoritma Floyd-Warshall untuk Menentukan Rute Terpendek Menuju Air Terjun Waimarang. *LAPLACE: Jurnal Pendidikan Matematika*, 3(2), 87-94.
- Salem, I. E., Mijwil, M. M., Abdulqader, A. W., & Ismaeel, M. M. (2022). Flight-schedule using Dijkstra's algorithm with comparison of routes findings. *International Journal of Electrical and Computer Engineering*, 12(2), 1675.
- Sitompul, A., Charles, J., & Udjulawa, D. Implementasi Algoritma Floyd Warshall dalam Menentukan Jalur Terbaik Driver Pastifresh. Id.