KAJIAN TENTANG ALGORITMA DIJKSTRA, FLOYD-WARSHALL DAN A* DALAM PENCARIAN RUTE TERPENDEK

SKRIPSI

Diajukan untuk Memenuhi Sebagian Syarat Guna Memperoleh
Gelar Sarjana (Matematika)
dalam Ilmu Matematika

Oleh : **CHICHA AMALIA PUTRI**
NIM : 2008046014

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI WALISONGO
SEMARANG
2024

PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini :

Nama	:	Chicha Amalia Putri
NIM	:	2008046014
Jurusan/Program Studi	:	Matematika/ Matematika

menyatakan bahwa skripsi yang berjudul :

KAJIAN TENTANG ALGORITMA DIJKSTRA, FLOYD-WARSHALL DAN A* DALAM PENCARIAN RUTE TERPENDEK

secara keseluruhan adalah hasil penelitian/karya saya sendiri, kecuali bagian tertentu yang dirujuk sumbernya.

Semarang, 27 Mei 2024

Pernyataan,

Chicha Amalia Putri

NIM : 2008046014

KEMENTERIAN AGAMA R.I.
UNIVERSITAS ISLAM NEGERI WALISONGO
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Prof. Dr. Hamka (Kampus III) Ngaliyan Semarang
Telp. 024-7601295 Fax. 7615387

PENGESAHAN

Naskah skripsi berikut ini :

Judul : **Kajian tentang Algoritma Dijkstra, Floyd-Warshall dan A* dalam Pencarian Rute Terpendek**
Penulis : Chicha Amalia Putri
NIM : 2008046014
Jurusan : Matematika

Telah diujikan dalam sidang *tugas akhir* oleh Dewan Penguji Fakultas Sains dan Teknologi UIN Walisongo dan dapat diterima sebagai salah satu syarat memperoleh gelar sarjana dalam Ilmu Matematika.

Semarang, 24 Juni 2024

DEWAN PENGUJI

Penguji I,
Any Muanalifah, M.Si., Ph.D.
NIP : 198201132011012009

Penguji II,
Prihadi Kurniawan, M.Sc
NIP : 199012262019031012

Penguji III,
Seftina Diyah Miasary, M.Sc
NIP : 198709212019032010

Penguji IV,
Ariska Kurnia Rachmawati, M.Sc
NIP : 198908112019032019

Pembimbing I,
Prihadi Kurniawan, M.Sc.
NIP : 199012262019031012

Pembimbing II,
Agus Wayan Yulianto, M.Sc.
NIP : 198907162019031007

NOTA DINAS

Semarang, 28 Mei 2024

Yth. Ketua Program Studi Matematika
Fakultas Sains dan Teknologi
UIN Walisongo Semarang

Assalamu'alaikum warahmatullahi wabarakatuh

Dengan ini diberitahukan bahwa saya telah melakukan bimbingan, arahan dan koreksi naskah skripsi dengan:

Judul	:	Kajian tentang Algoritma Dijkstra, Floyd-Warshall dan A* dalam Pencarian Rute Terpendek
Nama	:	Chicha Amalia Putri
NIM	:	20108046014
Jurusan	:	Matematika

Saya memandang bahwa naskah skripsi tersebut sudah dapat diajukan kepada Fakultas Sains dan Teknologi UIN Walisongo untuk diujikan dalam Sidang Munaqasyah.

Wassalamu'alaikum warahmatullahi wabarakatuh

Pembimbing I,

Prihadi Kurniawan, M.Sc.
NIP : 199012262019031012

NOTA DINAS

Semarang, 29 Mei 2024

Yth. Ketua Program Studi Matematika
Fakultas Sains dan Teknologi
UIN Walisongo Semarang

Assalamu'alaikum warahmatullahi wabarakatuh

Dengan ini diberitahukan bahwa saya telah melakukan bimbingan, arahan dan koreksi naskah skripsi dengan:

Judul	:	Kajian tentang Algoritma Dijkstra, Floyd-Warshall dan A* dalam Pencarian Rute Terpendek
Nama	:	Chicha Amalia Putri
NIM	:	2008046014
Jurusan	:	Matematika

Saya memandang bahwa naskah skripsi tersebut sudah dapat diajukan kepada Fakultas Sains dan Teknologi UIN Walisongo untuk diujikan dalam Sidang Munaqasyah.

Wassalamu'alaikum warahmatullahi wabarakatuh

Pembimbing II,

Agus Wayan Yulianto, M.Sc.
NIP : 198907162019031007

ABSTRAK

Terdapat banyak algoritma yang telah dikembangkan dan dapat digunakan dalam pencarian rute terpendek. Penelitian ini bertujuan untuk mengkaji lebih lanjut Algoritma Dijkstra, Floyd-Warshall dan A* dalam sebuah graf. Pengujian dilakukan dengan mempertimbangkan total jarak, jumlah titik, dan rata-rata kompleksitas waktu yang dibutuhkan untuk menemukan rute terpendek antara dua titik dalam sebuah graf. Pengujian menggunakan simulasi rute graf pada skenario dengan titik awal yang berbeda tetapi titik tujuan yang sama. Hasil pengujian menunjukkan Algoritma Dijkstra cenderung memberikan solusi yang akurat tetapi memerlukan waktu yang sedikit lebih lama dibandingkan Algoritma A*, sedangkan Algoritma A* memiliki kinerja yang lebih cepat namun sensitif terhadap pemilihan fungsi heuristik, serta tidak dijamin menemukan solusi yang akurat. Di sisi lain, Algoritma Floyd-Warshall cocok untuk graf yang sangat padat atau sebagian besar titiknya memiliki sisi yang menghubungkannya tetapi memiliki kompleksitas waktu yang tinggi. Kesimpulan dalam penelitian ini adalah Algoritma Dijkstra dan Floyd-Warshall lebih baik dalam penentuan rute terpendek dalam total jarak serta jumlah titik, apabila kecepatan waktu tidak terlalu dipentingkan. Dalam segi kecepatan waktu Algoritma A* lebih cepat dibandingkan kedua algoritma lain tetapi tidak ada jaminan dapat menemukan rute terpendek. Namun, untuk menjamin bahwa rute yang ditemukan adalah jalur terpendek, heuristik yang digunakan haruslah *admissible* dan konsisten. Jika heuristik tidak memenuhi kriteria ini, A* dapat menghasilkan solusi yang lebih cepat tetapi tidak optimal.

Kata kunci : Graf, Algoritma, Algoritma A*, Algoritma Dijkstra, Algoritma Floyd-Warshall

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarokatuh

Alhamdulilah, segala puji dan syukur kehadirat Allah Swt. atas pertolongan, rahmat, dan kasih sayang-Nya, sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "Kajian tentang Algoritma Dijkstra, Floyd-Warshall dan A* dalam Pencarian Rute Terpendek". Selawat serta salam tidak lupa kita curahkan kepada junjungan besar Nabi Muhammad Saw. yang telah membimbing kita dari zaman jahiliyah menuju zaman yang dikaruniai oleh Allah Swt. Dalam proses penulisan tugas akhir ini, penulis menyadari tidak lepas dari bantuan banyak pihak. Oleh karena itu, sudah sudah sepantasnya penulis dengan hormat mengucapkan terimakasih dan mendoakan semoga Allah Swt. memberikan balasan terbaik kepada:

1. Bapak Prof. Dr. Nizar, M.Ag., selaku Rektor Universitas Islam Walisongo.
2. Bapak Prof. Dr. H. Musahadi, M.Ag., selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Walisongo.
3. Ibu Any Muanalifah, P.hD., selaku Ketua Prodi Jurusan Matematika Universitas Islam Negeri Walisongo.
4. Bapak Prihadi Kurniawan, M.Sc., selaku Dosen pembimbing I yang memberikan bimbingan, arahan, nasehat serta saran, sehingga penulis dapat menyelesaikan tugas akhir dengan baik. Bapak telah berperan sebagai sumber pengetahuan, mentor, dan pembimbing yang luar biasa dalam perjalanan

akademik saya. Setiap arahan, masukan, dan dorongan dari Bapak telah menjadi pendorong bagi saya untuk terus berkembang dan menyelesaikan tugas akhir ini. Semoga kebaikan Bapak terus membawa berkah bagi banyak orang di sekitar.

5. Bapak Agus Wayan Yulianto, M.Sc., selaku Dosen pembimbing II yang memberikan bimbingan, arahan, nasehat serta saran, sehingga penulis dapat menyelesaikan tugas akhir dengan baik. Bapak telah memberikan waktu, tenaga, dan perhatian yang berharga untuk membantu saya mencapai tujuan akademik saya. Semoga kebaikan Bapak terus membawa berkah bagi banyak orang di sekitar.

6. Ibu Dwi Winarni, S.Pd., selaku Ibu yang paling terbaik di dunia. Terima kasih tak terhingga atas segala kasih sayang, dukungan, dan pengorbanan yang telah Ibu berikan kepada saya sepanjang hidup ini. Kata-kata tidak cukup untuk mengungkapkan betapa berharganya kehadiran dan peran Ibu dalam hidup saya. Setiap langkah yang saya ambil, setiap pencapaian yang saya raih, dan setiap mimpi yang saya kejar, semuanya berawal dari cinta dan dukungan Ibu. Saya ingin mengucapkan terima kasih atas segala pengorbanan dan perhatian yang Ibu berikan kepada saya. Terima kasih telah menjadi teladan yang luar biasa, mengajari saya tentang nilai-nilai kehidupan, kejujuran, dan kerja keras. Saya berjanji untuk selalu menghargai setiap momen yang kita bagikan bersama, dan berusaha menjadi anak yang Ibu banggakan.

7. Terima kasih kepada sivitas akademik Program Studi

Matematika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Walisongo, terima kasih untuk seluruh ilmu serta bimbingan selama menempuh pendidikan di dunia perkuliahan.

8. Teman-teman satu perjuangan yang selalu memberikan dukungan dan menyemangati penulis dalam menyelesaikan tugas akhir ini.

9. Semua pihak yang tidak dapat penulis sebutkan satu persatu yang telah memberikan kontribusi hingga selesainya tugas akhir ini.

10. Saya ingin mengucapkan terima kasih kepada diri sendiri atas upaya, keberanian, dan komitmen yang telah Anda tunjukkan dalam mencapai impian dan tujuan Anda. Diri sendiri, terima kasih telah menjadi sahabat yang setia, pendukung yang tak tergoyahkan, dan terus menerus menjadi lebih baik. Terima kasih telah selalu berusaha melihat sisi positif dalam setiap situasi, tetap kuat dalam menghadapi kesulitan, dan tidak pernah menyerah pada impian dan tujuan. Saat ini, saya menghargai dan merayakan diri saya sendiri, lengkap dengan kelebihan dan kekurangan saya. Saya akan berusaha untuk terus mencintai dan menghormati diri saya sendiri, serta untuk selalu mendukung dan memotivasi diri saya sendiri dalam setiap langkah yang saya ambil di masa depan. Semoga saat ini menjadi pengingat bahwa Anda layak mendapatkan penghargaan dan penghormatan dari diri Anda sendiri. Terima kasih, diri sendiri, atas semua yang telah kita capai dan atas semua yang akan kita capai di masa depan.

Dengan kata-kata sederhana ini, saya ingin menutup kata pengantar ini dengan harapan bahwa pesan yang terkandung di dalamnya dapat menyentuh hati Anda sebagaimana hal ini menyentuh hati saya saat menulisnya. Terima kasih atas kesempatan untuk menyampaikan penghargaan dan rasa terima kasih kepada Anda, yang telah menjadi bagian penting dalam hidup saya.

Mari kita terus berjalan di sepanjang perjalanan hidup ini dengan penuh semangat, keberanian, dan kasih sayang. Mari kita saling mendukung, menghargai, dan merayakan satu sama lain setiap langkahnya. Semoga Allah Swt. selalu melimpahkan rahmat serta karunia-Nya kepada kita semua. Penulis berharap tugas akhir ini dapat bermanfaat kepada seluruh pembaca maupun bagi penulis sendiri. Amin.

Terima kasih sekali lagi, dan semoga hari-hari Anda penuh dengan kebahagiaan, cinta, dan berkah.

Wassalammu'alaikum Warahmatullahi Wabarakatuh.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

Halaman

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Tahun 1736, awal mula yang memperkenalkan teori graf adalah Leonhard Euler yang terkenal sebagai ahli Matematika dari Swiss melalui tulisannya yang membahas cara memecahkan permasalahan jembatan Konigsberg (Diestel, 2000). Teori graf mengambarkan suatu hubungan antar tujuan dua buah titik atau lebih, dengan setiap tujuan digambarkan oleh titik *(vertex)* dan penghubung antar tujuan digambarkan oleh sisi *(edge)* (West, 2001). Dalam teori graf salah satu persoalan yang kerap dikaitkan oleh teori graf adalah pencarian rute terpendek.

Dalam masa kini, permasalahan pencarian rute terpendek sering kali timbul dalam sebuah kajian yang banyak dibahas serta dipelajari sejak tahun 1950 (Distel, 2018). Luasnya kota serta banyaknya pilihan jalan kerap sekali menyulitkan dalam memilih rute yang optimal, baik dari biaya maupun jarak. Pada beberapa akhir ini, pencarian rute optimal menjadi masalah yang makin penting karena naiknya harga bahan bakar. Dalam perjalanan juga kerap sekali dihadapkan dengan berbagai masalah, seperti banyaknya pilihan jalan atau perbaikan jalan sehingga adanya perpindahan jalan yang dipilih atau terjadi demonstrasi, banjir, tanah retak, dan lainnya. Pilihan jalan yang dianggap optimal harus seketika berubah. Saat situasi seperti itu, pemilihan rute terpendek menjadi sukar karena terdapat beberapa rute jalan yang tidak dapat digunakan.

Pemilihan opsi jalan dengan waktu paling pendek sebenarnya

dapat dipilih menggunakan peta karena letak, lokasi serta jarak antar beberapa wilayah atau titik telah diperoleh. Akan tetapi, langkah yang digunakan hanya percobaan dan tanpa adanya algoritma yang digunakan dengan nyata dan tegas. Kumpulan tempat dan total jarak dari semua rute yang mungkin harus diperoleh terlebih dahulu, kemudian dibandingkan. Total jarak terkecil diakui sebagai rute yang paling efisien antara titik awal dan titik tujuan.

Rute terpendek telah banyak diterapkan dalam sistem perutean, sistem perutean jaringan, logistik dan *Global Positioning System* (GPS). Teknologi internet yang semakin berkembang pesat membuat informasi tersebar dan dapat diakses dimana saja sehingga waktu yang diperlukan untuk mencari informasi relatif cepat dan singkat (Meng dkk, 2023). Dalam permasalahan rute terpendek banyak algoritma yang biasa dipergunakan dalam menyelesaikan persoalan tersebut, antara lain Algoritma A*, Algoritma *Ant Colony Optimization* (ACO), Algoritma Bellman-Ford, Algoritma Dijkstra, Algoritma Floyd-Warshall, dan sebagainya (Buako, 2021). Penerapan sebuah algoritma dalam penentuan rute terpendek akan memudahkan dan meningkatkan efisiensi waktu tetapi meskipun begitu setiap algortima memiliki karakterisitiknya masing-masing (Munir, 2002).

Penelitian sebelumnya dilakukan oleh Aziz dkk (2022) yang berjudul *"A Comparative Analysis among Three Different Shortest Path-finding Algorithms"* menyatakan Algoritma A* lebih unggul apabila diperbandingkan oleh Algoritma *Ant Colony Optimization* (ACO) dan Dijkstra. Walaupun dalam penelitian tersebut Algoritma A* lebih baik, Algoritma A* masih mempunyai kekurangan pada bagian *runtime* yang mana berada di tingkat

terendah dibandingkan dengan Algoritma ACO dan Dijkstra.

Selain penelitian di atas, terdapat penelitian lain yang dilakukan oleh Sidhu dan Krishan (2022) yang berjudul *"Research On Dijkstra's, A*, Bellman-Ford And Floyd-Warshall Path Finding Algorithms"* yang menyatakan bahwa Algoritma A* lebih cepat dalam menemukan jalur optimal karena melintasi lebih sedikit titik dibandingkan dengan Algoritma Dijkstra yang melintasi lebih banyak titik dari pada Algoritma A*. Selain itu, dalam penelitian tersebut juga mengemukakan Algoritma Floyd-Warshall ialah algoritma yang dinamis dan lebih dulu memperkirakan rute yang terpendek antara setiap titik.

Selain literatur di atas terdapat banyak literatur lain yang penulis gunakan. Dalam penelitian masing-masing ahli terdapat banyak algoritma dalam penentuan sebuah rute yang paling pendek sehingga penulis menggunakan Algoritma A*, Dijkstra, dan Floyd-Warshall untuk memperkaya studi literatur dengan perbandingan antar metode dalam hal pencarian solusi rute terpendek. Penelitian ini menjadikan simulasi perbandingan antara tiga algoritma untuk mencari rute terpendek pada sebuah graf berbobot. Penulis tertarik untuk mengkaji tentang Algoritma Dijkstra, Floyd-Warshall, dan A* dalam Pencarian Rute Terpendek pada graf berbobot.

1.2 Rumusan Masalah

Pembahasan yang akan dikaji pada tugas akhir ini ialah bagaimana kinerja dari Algoritma Dijkstra, Floyd-Warshall, dan A* dalam sebuah pencarian rute terpendek pada graf?

1.3 Tujuan Penelitian

Penelitian ini bertujuan untuk mengetahui dan mengkaji kinerja dari Algoritma Dijkstra, Floyd-Warshall, dan A* dalam sebuah pencarian rute terpendek pada graf.

1.4 Manfaat Penelitian

Berdasarkan yang sudah dijabarkan sebelumnya, manfaat penelitian yang diperoleh ialah berikut ini.

1. Bagi penulis, media untuk mengembangkan dan mengaplikasikan wawasan tentang keilmuan Matematika di bidang aljabar, khususnya tentang graf.
2. Bagi Universitas, penelitian bisa menjadi media acuan dan menambah literatur perpustakaan yang berkaitan dengan graf untuk permasalahan pencarian rute terpendek.
3. Bagi pengembang ilmu pengetahuan, memberikan kemudahan untuk memahami dan menambah wawasan mendalam tentang teori graf untuk materi rujukan kajian selanjutnya, apalagi bagi yang akan melaksanakan penelitian serupa tentang pencarian rute terpendek menggunakan graf berbobot.

1.5 Metodologi Penelitian

Metode dalam penelitian yang digunakan untuk menulis tugas akhir ini adalah studi literatur. Studi literatur merupakan proses sistematik yang melibatkan identifikasi, evaluasi, dan sintesis

karya-karya ilmiah yang relevan untuk memahami pengetahuan yang ada tentang topik penelitian. Studi literatur bertujuan untuk memberikan landasan teori, mendefinisikan masalah penelitian, dan mengidentifikasi kekurangan atau area yang memerlukan penelitian lebih lanjut. Berikut proses penelitian yang dilakukan oleh penulis:

1. Mengkaji graf
   Mengkaji tentang definisi graf, jenis-jenis graf serta definisi masing-masing jenis graf, komplemen graf, dan definisi keterhubungan dalam graf.

2. Mengkaji rute terpendek
   Mengkaji tentang definisi rute terpendek dan definisi algoritma.

3. Mengkaji Algoritma Dijkstra
   Mengkaji tentang definisi Algoritma Dijkstra, cara kerja Algoritma Dijkstra, dan kelemahan serta kelebihan Algoritma Dijkstra.

4. Mengkaji Algoritma Floyd-Warshall
   Mengkaji tentang definisi Algoritma Floyd-warshall, cara kerja Algoritma Floyd-Warshall, dan kelemahan serta kelebihan Algoritma Floyd-Warshall.

5. Mengkaji Algoritma A*
   Mengkaji tentang definisi Algoritma A*, cara kerja Algoritma A*, dan kelemahan serta kelebihan Algoritma A*.

BAB II

LANDASAN TEORI

2.1 Graf

Teori graf pertama kali diperkenalkan oleh seorang matematikawan dari Swiss, bernama Leonhard Euler yang mengatakan rahasia teka-teki jembatan Konigsberg. Tahun 1824 sampai 1887, Gustav Robert Kirchhoff berhasil mengembangkan teori pohon yang digunakan untuk permasalahan jaringan listrik. Lalu pada 1821 sampai 1895 kemudian, Arthur Cayley menggunakan konsep pohon untuk menerangkan masalah kimia tentang hidrokarbon. Teori graf adalah topik tua yang terdapat banyak aplikasinya pada era modern sampai masa kini (West, 2001).

Graf merupakan himpunan model yang dihubungkan oleh himpunan berhingga yang anggota tidak boleh kosong biasa disebut titik (*vertices*) dan himpunan berhingga yang anggota boleh kosong disebut sisi (*edges*). Graf dipergunakan dalam mengambarkan objek yang memiliki elemen berhingga dan memiliki sebuah hal yang berhubungan diantara satu objek dengan objek lainnya. Gambaran graf mengatakan bahwa objek itu digambarkan dengan titik dan penghubung diantara satu objek dengan objek lainnya digambarkan dengan sisi.

Definisi 2.1.1 ***(Graf, e.g Munir, 2005)***

Graf G *diinterpretasi dengan sebuah pasangan himpunan berhingga* $V(G)$ *dan* $E(G)$ *disimbolkan* $G = (V(G), E(G))$*, dengan* $V(G)$ *ialah himpunan berhingga dari titik-titik (vertices) pada graf* G *dan* $E(G)$ *adalah himpunan berhingga yang menghubungkan*

sepasang titik disebut dengan sisi (edge) pada graf G yang mana setiap sisi dalam himpunan $E(G)$ akan menyatukan beberapa titik pada $V(G)$. Dalam sebuah graf dapat diperbolehkan tidak memiliki sisi, akan tetapi harus tetap memiliki sebuah titik.

Titik dalam suatu graf sering diberi nama dengan sebuah huruf atau bilangan asli, seperti $A, B, C, ...$ atau $\{v_1, v_2, v_3, ..., v_n\}$ ataupun gabungan. Sedangkan sisi yang menghubungkan titik u dengan titik v dapat dituliskan (u, v) atau sisi dilambangkan dengan $e_1, e_2, e_3, ..., e_n$. Dapat dikatakan, e ialah sisi yang memautkan titik u dengan titik v, dinotasikan $e = (u, v)$. Jumlah titik-titik dalam graf $G = (V(G), E(G))$ disimbolkan oleh $|V(G)|$. Lalu, jumlah sisi-sisi dalam graf $G = (V(G), E(G))$ disimbolkan $|E(G)|$.

Definisi 2.1.2 ***(Bertetangga, e.g Siang, 2002)***

Dalam graf dua buah titik dapat disebut bertetangga (adjacent) apabila kedua titik dipautkan dengan sebuah sisi.

Definisi 2.1.3 ***(Bersisian, e.g Siang, 2002)***

Sisi dapat dikatakan bersisian (incident) jika sembarang sisi menghubungkan dua buah titik pada graf.

Definisi 2.1.4 ***(Loop, e.g Siang, 2002)***

Sisi yang memautkan titik dengan titik itu sendiri dikatakan dengan gelang (loop).

Definisi 2.1.5 ***(Sisi Ganda, e.g West, 2001)***

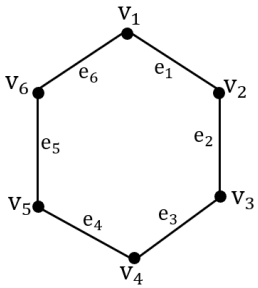
Dua sisi atau lebih yang menghubungkan dua titik yang sama disebut sisi ganda (multiple edges).

Definisi 2.1.6 ***(Titik Terpencil, e.g Siang, 2002)***

Titik yang tidak mempunyai sisi disebut titik terpencil/terisolasi.

Definisi 2.1.7 ***(Derajat, e.g Siang, 2002)***

Misalkan u adalah titik dari sebuah graf G. Derajat (degree) titik u disimbolkan dengan $d(u)$ merupakan banyak sisi yang menempel dengan titik u dan sisi dengan loop akan menyumbangkan dua derajat.



Gambar 2.1. Graf C_6

Contoh 2.1.1 *Berdasarkan Gambar 2.1, diperoleh himpunan titik dan sisi dalam graf C_6 ialah $V(G)$ = {$v_1, v_2, v_3, v_4, v_5, v_6$} dengan banyaknya titik dalam graf tersebut $|V(G)|$ = 6 dan $E(G)$ = {$e_1 = v_1v_2, e_2 = v_2v_3, e_3 = v_3v_4, e_4 = v_4v_5, e_5 = v_5v_6, e_6 = v_6v_1$} dengan banyaknya sisi dalam graf $|E(G)|$ = 6. Akan ditunjukkan contoh hubungan antara titik dan sisi.*

1. *Titik v_1 dikatakan bertetangga dengan titik v_2 karena terdapat sisi yang menghubungkannya.*

2. *Titik v_3 dikatakan bersisian dengan sisi {v_2, v_3} dan {v_3, v_4} karena titik v_3 berada satu ujung dari kedua sisi tersebut.*

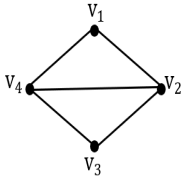
2.1.1 Jenis-jenis Graf

Graf dibagi menjadi beberapa jenis dan kelompok, tergantung dari sudut pandang pengelompokkannya. Pengelompokkan graf berdasarkan ada tidaknya sisi ganda atau *loop*, berdasarkan arah pada sisi dan berdasarkan strukturnya (Munir, 2005).

Secara umum, berdasarkan ada atau tidaknya sebuah sisi ganda atau loop suatu graf dapat dibedakan menjadi dua jenis, yaitu:

Definisi 2.1.8 ***(Graf Sederhana (Simple Graph), e.g Harju, 2012)***

Graf sederhana adalah graf yang tidak memiliki lebih dari satu sisi pada tiap titiknya dan tidak memiliki sisi yang menghubungkan titik dengan dirinya sendiri loop. Dalam sebuah graf sederhana, sisi ialah pasangan tidak berurutan. Pada penulisannya, sisi (u, v) = (v, u)*. Representasi graf sederhana pada Gambar 2.2 berikut ini.*

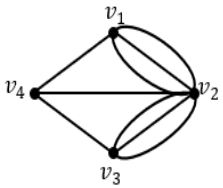


Gambar 2.2. Graf Sederhana

Definisi 2.1.9 ***(Graf Tak Sederhana (Unsimple Graph), e.g Harju, 2012)***

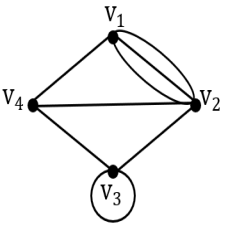
Graf tak sederhana adalah graf yang memiliki sebuah sisi ganda maupun loop. Secara umum, graf tak sederhana dibagi menjadi dua, yaitu:

1. *Graf Ganda (Multigraph)*
   Graf ganda merupakan graf yang mempunyai beberapa sisi yang dapat menggabungkan dua buah titik yang sama. Salah satu representasi graf ganda di bawah ini.



Gambar 2.3. Graf Ganda

2. *Graf Semu (Pseudograph)*
   Graf semu merupakan graf yang memiliki sisi ganda, loop, maupun keduanya. Salah satu representasi graf semu berikut ini.



Gambar 2.4. Graf Semu

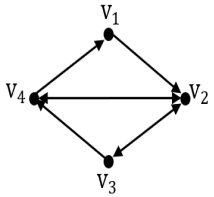
Berdasarkan pada orientasi jenis graf berdasarkan arah sisinya dapat dibagi menjadi dua jenis, antara lain:

Definisi 2.1.10 ***(Graf Tak Berarah (Undirected Graph), e.g Bondy, 1982)***

Graf tak berarah merupakan graf dimana sisinya tidak memiliki tujuan arah. Dalam graf ini, deretan pasangan titik yang digabungkan dengan sebuah sisi tidak diperhitungkan. Jadi, (u, v) = (v, u) adalah sisi yang sama. Contoh graf tak berarah terdapat di Gambar 2.1 hingga Gambar 2.4.

Definisi 2.1.11 ***(Graf Berarah (Directed Graph atau Digraph), e.g Bondy, 1982)***

Graf berarah merupakan graf yang setiap sisinya memiliki orientasi arah. Dalam graf berarah, deretan pasangan titik yang digabungkan oleh sebuah sisi diperhitungkan, dapat ditulis dengan $(u, v) \neq (v, u)$. Representasi contoh graf berarah pada Gambar 2.5 berikut ini.



Gambar 2.5. Graf Berarah

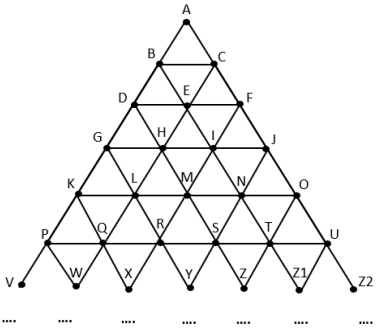
Secara umum, graf dapat ditinjau berdasarkan jumlah titik yang menyusun suatu graf tersebut dibagi menjadi dua jenis, yaitu:

Definisi 2.1.12 ***(Graf Berhingga (Limited Graph), e.g Diestel, 2000)***

Graf berhingga merupakan graf dengan banyak titiknya n terbatas. Representasi graf berhingga pada Gambar 2.1 hingga Gambar 2.5.

Definisi 2.1.13 *(**Graf Tak Berhingga (Unlimited Graph), e.g Diestel, 2000**)*

Graf tak berhingga merupakan graf dengan banyak titiknya n tak berhingga. Representasi graf tak berhingga digambarkan berikut ini.

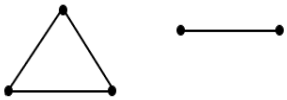


Gambar 2.6. Graf Tak Berhingga

Selain graf yang telah disebutkan sebelumnya, terdapat graf sederhana yang khusus. Terdapat graf khusus yang sering ditemukan, yaitu:

Definisi 2.1.14 *(**Graf Lengkap (Complete Graph), e.g Siang, 2002**)*

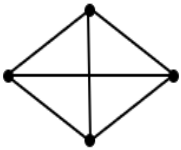
Graf lengkap merupakan graf sederhana yang setiap titiknya terhubung dengan semua titik yang lain dengan hanya satu sisi. Graf lengkap dengan n buah titik dapat dinotasikan dengan K_n. Setiap titik K_n memiliki derajat $n-1$ dengan jumlah sisi terdiri dari n titik adalah $\frac{n(n-1)}{2}$. Contoh representasi graf lengkap digambarkan di bawah ini.



Gambar 2.7. Graf Lengkap

Definisi 2.1.15 *(**Graf Teratur (Reguler Graph), e.g Siang, 2002**)*

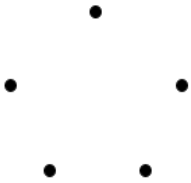
Graf teratur merupakan graf sederhana yang setiap titiknya memiliki derajat yang sama. Apabila dalam graf setiap titiknya adalah r, maka graf tersebut dikatakan sebagai graf teratur berderajat r. Representasi graf pada Gambar 2.8..



Gambar 2.8. Graf Teratur

Definisi 2.1.16 *(**Graf Kosong (Trivial Graph), e.g Siang, 2002**)*

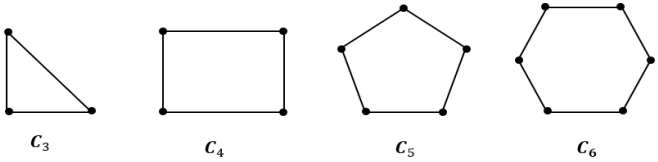
Graf kosong atau graf trivial merupakan graf yang hanya memiliki sebuah titik tanpa memiliki sisi. Contoh representasi graf digambarkan di bawah ini.



Gambar 2.9. Graf Kosong atau Graf Trivial

Definisi 2.1.17 ***(Graf Lingkaran (Cycle Graph), e.g Siang, 2002)***

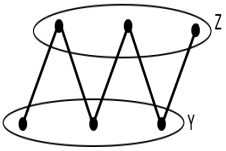
Graf lingkaran merupakan graf sederhana yang dimana setiap titiknya memiliki derajat 2. Graf lingkaran dengan n titik dilambangkan dengan C_n. Contoh representasi graf lingkaran digambarkan di bawah ini.



Gambar 2.10. Graf Lingkaran

Definisi 2.1.18 ***(Graf Bipartit (Bipartite Graph), e.g Siang, 2002)***

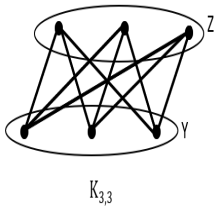
Graf G dapat dikatakan graf bipartit apabila $V(G)$ adalah gabungan dari dua buah himpunan bagian tak kosong, misal Y dan Z, sehingga setiap sisi dalam graf G menghubungkan sebuah titik didalam Y dengan titik didalam Z. Dengan maksud, setiap pasang titik di Y maupun titik di Z tidak bertetangga. Contoh representasi gambar graf bipartit dapat dilihat di bawah ini.



Gambar 2.11. Graf Bipartit

Apabila setiap titik di Y terhubung dengan seluruh titik di Z, maka dapat dikatakan graf bipartit lengkap (complate bipartite

graph). Jika Y terdiri dari m titik dan Z terdiri dari n titik, maka graf bipartit lengkap sering disimbolkan dengan $K_{m,n}$. Contoh representasi gambar graf bipartit lengkap dapat dilihat di bawah ini.

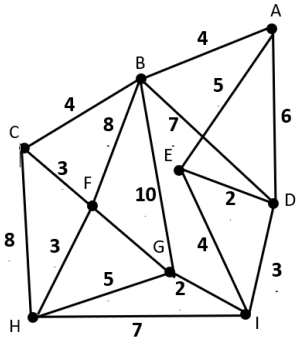


Gambar 2.12. Graf Bipartit Lengkap

2.1.2 Graf Berbobot

Definisi 2.1.19 ***(Graf Berbobot (Weighted Graph), e.g Munir, 2005)***

Graf berbobot (Weighted Graph) merupakan sebuah graf G yang setiap sisinya diberi sebuah bilangan real (bobot).



Gambar 2.13. Graf Berbobot

Dengan bobot tiap sisi dapat berbeda, tergantung dengan model masalah dalam graf. Bobot pada graf menyatakan jarak antara dua buah titik atau kota, biaya, ongkos produksi, dan lainnya.

2.1.3 Komplemen Graf

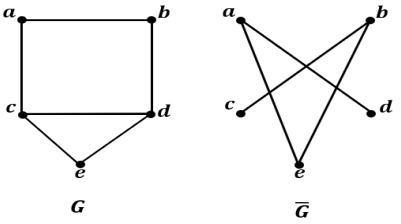
Definisi 2.1.20 ***(Komplemen Graf, e.g Munir, 2002)***

Komplemen suatu graf G disimbolkan dengan $\bar{G}$ dengan n titik adalah sebuah graf sederhana dengan:

1. *Titik-titik dalam $\bar{G}$ sama dengan titik-titik G, disombolkan dengan V ($\bar{G}$) = $V(G)$.*
2. *Kemudian sisi-sisi $\bar{G}$ adalah komplemen sisi-sisi G terhadap graf lengkapnya (K_n).*

$$E(\bar{G}) = E(K_n) - E(G) \tag{2.1}$$

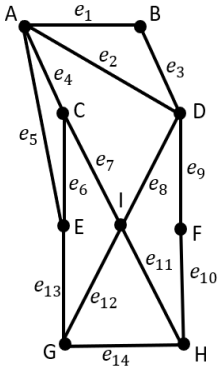
Titik yang dihubungkan dengan sisi dalam graf G tidak terhubung dalam $\bar{G}$. Sebaliknya, titik-titik yang terhubung dalam graf G menjadi tidak terhubung dalam $\bar{G}$. Dapat dilihat pada gambar di bawah, bahwa $\bar{G}$ merupakan komplemen bagian dari graf G.



Gambar 2.14. Graf G dan Komplemen Graf G

2.1.4 Keterhubungan

Sebuah graf terhubung jika memiliki sebuah sisi diantara beberapa titik. Terdapat beberapa terapan dalam teori graf pada bidang transportasi dan jaringan komunikasi. Model sistem rute perjalanan dan penentuan waktu tersingkat dalam pengiriman pesan dengan jaringan telekomunikasi merupakan bentuk sistem dalam graf yang biasa digunakan. Graf dengan V = $\{v_1, v_2, v_3, ..., v_k\}$ dan E = $\{e_1, e_2, e_3, ..., e_k\}$, membuat barisan tak kosong W = $\{v_1, e_1, v_2, e_2, ..., v_k, e_k\}$. Pada Gambar 2.15, digunakan dalam mempelajari beberapa istilah yang berhubungan dengan keterhubungan dalam sebuah graf.



Gambar 2.15. Contoh Graf C_9

Berikut beberapa keterhubungan graf yang dapat dijelaskan, yaitu jalan (*walk*), jejak (*trail*), lintasan (*path*), sirkuit (*circuit*), dan sikel (*cycle*).

1. Jalan (*Walk*)

   Jalan pada suatu graf G merupakan sebuah barisan tak kosong dimana setiap suku-sukunya ada titik dan sisi secara

berurutan bergantian (Mardiyono, 1996). Berdasarkan sebuah titik awal dan titik akhir pada jalan dibagi menjadi dua, yaitu jalan terbuka dan jalan tertutup. Sebuah jalan dikatakan terbuka apabila titik awal dan titik akhir berbeda atau tidak sama. Sedangkan sebuah jalan dikatakan jalan terbuka apabila titik awal dan titik akhirnya berbeda (Vasudev, 2006). Dalam Gambar 2.15, contoh suatu jalan berikut ini.

- Jalan = $\{A, e_2, D, e_8, I, e_{12}, I, e_8, D\}$
- Jalan terbuka = $\{A, e_2, D, e_8, I, e_{12}, I\}$
- Jalan tertutup = $\{A, e_2, D, e_8, I, e_{12}, I, e_8, D, e_2, A\}$

2. Jejak *(Trail)*

   Suatu barisan graf G dapat dikatakan jejak jika memiliki sisi yang berbeda (Mardiyono, 1996). Dengan kata lain, jejak merupakan jalan tanpa sisi berulang sehingga sebuah jejak dibagi menjadi dua. Pertama, sebuah jejak dikatakan jejak terbuka apabila titik awal dan titik akhir berbeda sedangkan dikatakan jejak tertutup apabila titik awal dan titik akhir sama (Vasudev, 2006). Dalam Gambar 2.15, contoh suatu jejak adalah berikut ini.

   - Jejak = $\{A, e_4, C, e_7, I, e_{11}, H\}$
   - Jejak terbuka = $\{A, e_2, D, e_3, B, e_1, A, e_4, C\}$
   - Jejak tertutup = $\{A, e_4, C, e_7, I, e_{11}, H, e_{10}, F, e_9, D, e_2, A\}$

3. Lintasan *(Path)*

   Suatu barisan graf G dapat dikatakan lintasan jika memiliki titik yang tidak berulang. (Mardiyono, 1996).

Dalam Gambar 2.15, contoh suatu lintasan adalah $W = \{A, e_4, C, e_7, I, e_{11}, H, e_{10}, F\}$.

4. Sirkuit *(Circuit)*

   Sirkuit adalah jalan tertutup dengan titik yang dapat berulang tetapi setiap sisi hanya dapat dilalui sekali (West, 2001). Dalam Gambar 2.15, contoh suatu sirkuit adalah $W = \{C, e_7, I, e_{11}, H, e_{10}, F, e_4, D, e_8, I, e_{12}, G, e_{13}, E, e_6, C\}$

5. Sikel *(Cycle)*

   Sikel adalah jalan tertutup dalam graf dimana tidak ada titik atau sisi yang berulang, kecuali titik awal dan akhir (West, 2001). Dalam Gambar 2.15, contoh suatu sikel adalah $W = \{C, e_7, I, e_{11}, H, e_{14}, G, e_{12}, E, e_6, C\}$

2.2 Optimasi

Optimasi merupakan salah satu ilmu pada matematika yang memfokuskan untuk mendapatkan nilai minimum atau maksimum secara sistematis dari suatu peluang, fungsi, maupun pencarian nilai lainnya dalam suatu masalah. Optimasi digunakan pada berbagai macam bidang untuk mencapai efektivitas dan efisiensi dari sasaran yang diinginkan. Dalam model matematika, tujuan dari optimasi ialah minimal dalam penelitian ini.

Dapat didefinisikan bahwa optimasi merupakan segala bentuk cara yang melibatkan banyak konsep untuk mencari nilai optimal, efektif, dan efisien. Pencarian optimasi pada penelitian ini adalah pencarian rute terpendek. Beberapa metode untuk menyelesaikan permasalahan pencarian rute terpendek, yaitu

metode konvensional dan metode heuristik (Mutakhiroh dkk, 2007).

1. Metode Konvensional

   Metode konvensional dalam optimasi biasanya melibatkan pendekatan matematis yang terstruktur dan sistematis, seperti pemrograman linear, pemrograman integer, pemrograman dinamis, dan metode simplex. Pendekatan ini digunakan untuk menemukan solusi optimal dengan menggunakan algoritma yang sudah terdefinisi dengan baik. Beberapa algoritma dalam metode konvensional yang dapat digunakan untuk pencarian rute terpendek, yaitu Algoritma Dijkstra, Algoritma Bellman-Ford, Algoritma Floyd-Warshall, Algoritma Kruskal dan lainnya (Wibowo, 2012).

2. Metode Heuristik

   Metode heuristik dalam optimasi digunakan untuk menyelesaikan masalah yang kompleks dan tidak dapat diselesaikan dengan mudah oleh metode konvensional. Metode heuristik berfokus pada menemukan solusi yang cukup baik dalam waktu yang wajar, meskipun mungkin bukan solusi optimal. Beberapa algoritma dalam metode heuristik yang dapat digunakan untuk permasalahan pencarian rute terpendek, yaitu *Simulated Annealing*, Algoritma Memetika, Algoritma Genetika, *Ant Colony Optimization* (ACO), Algoritma A*, dan lainnya (Suryani, 2010).

2.3 Rute Terpendek

Rute terpendek merupakan suatu persoalan untuk mencari rute antara dua buah titik pada sebuah graf berbobot yang memiliki gabungan jumlah bobot pada sisi yang di lalui dengan jumlah yang paling kecil atau minimum. Cara ini juga dapat digunakan apabila ingin mencari rute termurah. Dalam permasalahan rute terpendek dibantu oleh banyak algoritma.

Algoritma merupakan suatu upaya yang memiliki urutan atau prosedur operasi yang disusun dengan logis dan sistematis agar dapat menyelesaikan sebuah masalah untuk menghasilkan suatu *output* atau keluaran hasil tertentu. Algoritma yang digunakan dalam pencarian rute terpendek berikut ini.

2.3.1 Algoritma Dijkstra

Pada tahun 1959, Algoritma Dijkstra ditemukan oleh Edsger Wybe Dijkstra (Diestel, 2018). Algoritma Dijkstra adalah salah satu bentuk algoritma yang popular dalam persoalan yang terkait dengan optimasi dan bersifat sederhana. Algoritma Dijkstra merupakan sebuah algoritma yang dapat digunakan dalam pencarian rute terpendek dari suatu titik tertentu ke setiap titik lain pada suatu graf.

Algoritma Dijkstra merupakan algoritma untuk mencari panjang rute terpendek dari sebuah titik ke sebuah titik lain di graf berbobot, dengan bobot setiap sisi adalah bilangan positif (Munir, 2008). Cara kerja Algoritma Dijkstra adalah membuat jalur ke satu titik pada setiap langkah. Artinya, pada langkah ke n, setidaknya ada n titik yang sudah diketahui jalur terpendeknya (Sunardi dkk, 2019).

Dalam penggunaan algoritma Dijkstra dalam pencarian rute terpendek terdapat kelebihan dan kekurangan yang harus diperhatikan. Terdapat beberapa kelebihan algoritma Dijkstra, antara lain:

1. Keakuratan
   Dalam penggunaan algoritma Dijkstra untuk mecari rute terpendek akan menghasilkan rute dengan bobot yang optimal sehingga algoritma ini dapat digunakan dan diandalkan di berbagai kasus.

2. Efisiensi
   Algoritma ini dapat menemukan rute terpendek dengan tepat, cermat dan tidak membuang banyak waktu bahkan dapat digunakan pada graf yang besar ataupun kompleks.

3. Implementasi
   Algoritma ini dapat diimplementasikan dalam banyaknya kasus atau kondisi, seperti perencanaan logistik, rute menuju rumah sakit, jalur evakuasi darurat, jaringan komputer, dan lainnya.

Selain kelebihan yang telah disebutkan sebelumnya, Algoritma Djikstra juga memiliki beberapa kekurangan dalam penggunaannya, antara lain:

1. Bobot negatif
   Dalam penggunaannya Algoritma Dijkstra tidak dapat bekerja dengan efisien dan baik bila terdapat bobot negatif pada graf, namun jika terjadi demikian, maka hasil yang diberikan adalah infiniti atau jumlah tak terbatas.

2. Kompleksitas

   Dalam graf yang sangat besar atau memiliki kerumitan yang tinggi, algoritma ini dapat bekerja menjadi lambat dan memakan banyak sumber daya.

Didefinisikan sebuah graf G merupakan graf berbobot yang memuat titik dan lintasan terpendek yang dipecahkan ialah v_1 menuju v_n. Algoritma Dijkstra diawali dari titik v_1, dengan setiap iterasi akan menemukan sebuah titik dengan nilai bobot paling kecil dari titik awal. Titik tersebut disebut dengan titik tetap dan sudah tidak dipertimbangkan lagi pada iterasi selanjutnya.
Diberikan:

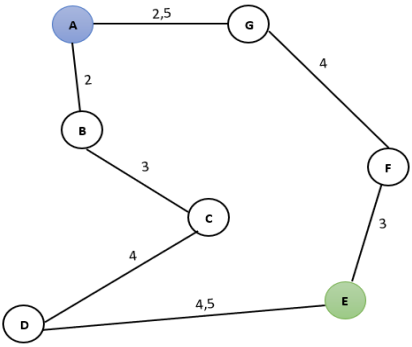
$V(G)$: $\{v_1, v_2, ..., v_n\}$
L : kumpulan beberapa titik $V(G)$ yang terpilih
$D(j)$: jumlah bobot jalur terkecil berawal V_1 ke V_j
$W(i, j)$: bobot garis dari titik V_i ke V_j
$W(1, j)$: jumlah bobot terkecil jalur berawal V_1 menuju V_j

Algorithm 1 Algoritma Dijkstra

1. Menentukan: L ={ , };
   $V(G)$ = $\{v_1, v_2, ..., v_n\}$.
2. Untuk $i = 2, ..., n$; lakukan $D(i)$ = $W(1, i)$
3. Apabila $V_n \notin L$ (v_n tidak menjadi titik tetap, maka)
   (a) Tentukan titik $v_k \in V - L$ dimana $D(k)$ terkecil, lalu $L = L \cup \{v_k\}$ (buat v_k sebagai sebuah titik tetap)
   (b) Untuk setiap $v_j \in V - L$ maka diterapkannya: Seandainya $D(j) > D(k) + W(k, j)$ maka dapat diubah *D(j)* menjadi $D(k) + W(k, j)$

Berdasarkan Algoritma Dijkstra, rute terpendek berawal v_1 menuju v_n terjadi dengan melewati titik dalam L dengan cara berurut-urut dan total bobotnya yang paling kecil, yaitu $D(n)$ (Siang, 2002).

Contoh 2.3.1 *Rute di bawah ini digunakan untuk simulasi mencari rute terpendek menggunakan Algoritma Dijkstra.*



Gambar 2.16. Simulasi Rute

Tabel 2.1. Perhitungan dengan Algoritma Dijkstra

Titik	A	B	C	D	F	G	E
A	**0-A**	2-A	-	-	-	2,5-A	-
B		**2-A**	5-B	-	-	2,5-A	-
G			5-B	-	6,5-G	**2,5**-A	-
C			**5-B**	9-C	6,5-G		-
F				9-C	**6,5-G**		9,5-F
D				**9-C**			9,5-F
E							**9,5**-F

Ditunjukkan perhitungan pada Tabel 2.1 di bawah ini dengan menggunakan Algoritma Dijkstra, diperoleh jumlah jarak jalur rute terpendeknya adalah 9,5. Rute yang dilewati dari titik $A \rightarrow G \rightarrow F \rightarrow E$*, dengan banyaknya titik ialah empat buah.*

2.3.2 Algoritma Floyd-Warshall

Tahun 1962, Algoritma Floyd-Warshall diperkenalkan oleh Robert Floyd. Algoritma ini adalah sebuah prosedur dalam memecahkan kendala dengan mempertimbangkan solusi yang didapat menjadi suatu pertimbangan dimana solusi tersebut berkaitan satu sama lain. Solusi tersebut didapatkan dari solusi tahap sebelumnya dan terdapat kemungkinan bahwa solusi yang dihasilkan bertambah dan tidak hanya satu. Algoritma ini menjalankan pemprograman dinamis yang akan membuktikan sebuah keberhasilan dalam menemukan solusi paling baik dalam persoalan pencarian rute terpendek dari seluruh titik.

Menurut Siang (2011), algoritma ini ialah sebuah algoritma yang sangat tepat dalam pencarian jarak yang paling pendek dari semua titik ke semua titik. Algoritma ini lebih mahir dalam mengambil keputusan yang tidak memfokuskan menuju titik solusi. Prinsip dalam algoritma ini merupakan prinsip optimalisasi yang memiliki arti bahwa seandainya penyelesaian setiap langkahnya ialah penyelesaian yang terbaik, maka bagian penyelesaian hingga suatu langkah (misalkan langkah ke-j) juga terbaik (Siang, 2006).

Algoritma ini digunakan untuk memperkirakan bobot yang paling kecil dari semua rute yang menyambungkan sebuah pasangan titik dan menjalankan langsung pada seluruh pasangan titik (Nawagusti, 2018). Dalam sebuah graf G = (V, E) dengan jumlah titik *n*. Untuk mewujudkan sebuah rute dengan jarak yang paling pendek, maka harus ditambahkan matriks persegi yang mana matriks berordo *n* atau W = $[W_{i,j}]$, dengan kemungkinan nilai, yaitu:

1. $W_{i,j} = 0$, jika titik i = j
2. $W_{i,j} =$ bobot sisi, jika i $\neq$ j dan titik v_i terhubung dengan titik v_j
3. $W_{i,j} = \infty$, jika i $\neq$ j dan titik v_i tidak terhubung dengan titik v_j

Menurut Siang (2011), langkah kerja Algoritma Floyd-Warshall dalam pencarian rute yang terpendek berikut ini.

Algorithm 2 Algoritma Floyd-Warshall

1. Inisialisasi: matriks bobot awal $W = W_0$
2. Iterasi:
   Untuk k = 1 hingga n, lakukan;
   Untuk i = 1 hingga n, lakukan;
   Untuk j = 1 hingga n, lakukan;
   (a) Jika $W_{i,j} \geq W_{i,k}$ + $W_{k,j}$, maka tukar $W_{i,j}$ dengan $W_{i,k}$ + $W_{k,j}$
   (b) Jika $W_{i,j} \leq W_{i,k}$ + $W_{k,j}$, maka $W_{i,j}$ tidak perlu ditukar dengan $W_{i,k}$ + $W_{k,j}$
3. Jika jarak yang ditemukan melalui titik k lebih kecil, perbarui jarak dari i ke j.
4. Jika tidak, jarak tetap seperti semula.
5. Output: Matriks yang berisi seluruh jarak terpendek W* = W.

Keterangan:
W_0 : matriks ketetanggan graf berlabel awal
*W** : matriks ketetanggan minimal

$Z^{(0)}$: matriks ketetanggan jalur awal

Z^* : matriks ketetanggan jalur minimal

$W_{i,j}$: jalur terpendek dari titik v_i ke v_j.

Contoh 2.3.2 *Untuk setiap matriks W dicek apakah* $W[i,j] \geq W[i,k] + W[k,j]$. *Jika ya, maka* $W[i,j]$ *akan diganti dengan* $W[i,k] + W[k,j]$. *Pada simulasi Gambar 2.16, akan dibuat menjadi matriks dibawah ini, dengan:*

k = 0, 1, 2, 3, 4, 5, 6, 7

i = 1, 2, 3, 4, 5, 6, 7

j = 1, 2, 3, 4, 5, 6, 7

Iterasi k = 0

$$W_0 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ F \\ G \\ E \end{array} \begin{array}{c} \begin{array}{ccccccc} A & B & C & D & F & G & E \end{array} \\ \begin{pmatrix} 0 & 2 & \infty & \infty & \infty & 2,5 & \infty \\ 2 & 0 & 3 & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 4 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & \infty & 4,5 \\ \infty & \infty & \infty & \infty & 0 & 4 & 3 \\ 2,5 & \infty & \infty & \infty & 4 & 0 & \infty \\ \infty & \infty & \infty & 4,5 & 3 & \infty & 0 \end{pmatrix} \end{array}$$

Dalam iterasi k = 1

Pilih bobot yang terkecil dari titik yang berhubungan $W_{i,j} \geq W_{i,k}$ *+* $W_{k,j}$ *dimana untuk setiap* $i, j = 1, 2, ..., n$. *Kemudian tentukan titik lintasan yang paling kecil berikut ini.*

- *Untuk* (i, j) *= (2, 6):* $W_{2,6} \geq W_{2,1} + W_{1,6}$, *yaitu* $\infty \geq 2 + 2,5$. *Jadi,* $W_{i,j} = 4,5$

- *Untuk* $(i, j) = (4, 3)$*:* $W_{4,3} \geq W_{4,1} + W_{1,3}$*, yaitu* $\infty \geq \infty + \infty$*. Jadi,* $W_{i,j} = \infty$

- *Untuk* $(i, j) = (5, 6)$*:* $W_{5,6} \geq W_{5,1} + W_{1,6}$*, yaitu* $4 \leq 2,5 + \infty$*. Jadi,* $W_{i,j} = 4$

Iterasi k = 1

$$W_1 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ F \\ G \\ E \end{array} \begin{array}{c} \begin{array}{ccccccc} A & B & C & D & F & G & E \end{array} \\ \begin{pmatrix} 0 & 2 & \infty & \infty & \infty & 2,5 & \infty \\ 2 & 0 & 3 & \infty & \infty & 4,5 & \infty \\ \infty & \infty & 0 & 4 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & \infty & 4,5 \\ \infty & \infty & \infty & \infty & 0 & 4 & 3 \\ 2,5 & 4,5 & \infty & \infty & 4 & 0 & \infty \\ \infty & \infty & \infty & 4,5 & 3 & \infty & 0 \end{pmatrix} \end{array}$$

Dalam iterasi k = 2

Pilih bobot yang terkecil dari titik yang berhubungan $W_{i,j} \geq W_{i,k}$ *+* $W_{k,j}$*, kemudian tentukan titik lintasan yang paling kecil berikut ini.*

- *Untuk* $(i, j) = (1, 3)$*:* $W_{1,3} \geq W_{1,2} + W_{2,3}$*, yaitu* $\infty \geq 2 + 3$*. Jadi,* $W_{i,j} = 5$

- *Untuk* $(i, j) = (6, 3)$*:* $W_{6,3} \geq W_{6,2} + W_{2,3}$*, yaitu* $\infty \geq 4,5 + 3$*. Jadi,* $W_{i,j} = 7,5$

Dengan cara yang sama untuk menghitung bobot dan titik pada matriks setiap i *dan* j *untuk iterasi-iterasi berikutnya. Diperoleh matriks dalam graf untuk iterasi* k *= 2*

Iterasi k = 2

$$W_2 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ F \\ G \\ E \end{array} \begin{array}{c} \begin{array}{ccccccc} A & B & C & D & F & G & E \end{array} \\ \begin{pmatrix} 0 & 2 & 5 & \infty & \infty & 2,5 & \infty \\ 2 & 0 & 3 & \infty & \infty & 4,5 & \infty \\ \infty & \infty & 0 & 4 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & \infty & 4,5 \\ \infty & \infty & \infty & \infty & 0 & 4 & 3 \\ 2,5 & 4,5 & 7,5 & \infty & 4 & 0 & \infty \\ \infty & \infty & \infty & 4,5 & 3 & \infty & 0 \end{pmatrix} \end{array}$$

Iterasi k = 3

$$W_3 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ F \\ G \\ E \end{array} \begin{array}{c} \begin{array}{ccccccc} A & B & C & D & F & G & E \end{array} \\ \begin{pmatrix} 0 & 2 & 5 & 9 & \infty & 2,5 & \infty \\ 2 & 0 & 3 & 7 & \infty & 4,5 & \infty \\ \infty & \infty & 0 & 4 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & \infty & 4,5 \\ \infty & \infty & \infty & \infty & 0 & 4 & 3 \\ 2,5 & 4,5 & 7,5 & 11,5 & 4 & 0 & \infty \\ \infty & \infty & \infty & 4,5 & 3 & \infty & 0 \end{pmatrix} \end{array}$$

Iterasi k = 4

$$W_4 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ F \\ G \\ E \end{array} \begin{array}{c} \begin{array}{ccccccc} A & B & C & D & F & G & E \end{array} \\ \begin{pmatrix} 0 & 2 & 5 & 9 & \infty & 2,5 & 13,5 \\ 2 & 0 & 3 & 7 & \infty & 4,5 & 11,5 \\ \infty & \infty & 0 & 4 & \infty & \infty & 8,5 \\ \infty & \infty & \infty & 0 & \infty & \infty & 4,5 \\ \infty & \infty & \infty & \infty & 0 & 4 & 3 \\ 2,5 & 4,5 & 7,5 & 11,5 & 4 & 0 & 16 \\ \infty & \infty & \infty & 4,5 & 3 & \infty & 0 \end{pmatrix} \end{array}$$

Iterasi k = 5

$$W_5 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ F \\ G \\ E \end{array} \begin{array}{c} \begin{array}{ccccccc} A & B & C & D & F & G & E \end{array} \\ \begin{pmatrix} 0 & 2 & 5 & 9 & \infty & 2,5 & 13,5 \\ 2 & 0 & 3 & 7 & \infty & 4,5 & 11,5 \\ \infty & \infty & 0 & 4 & \infty & \infty & 8,5 \\ \infty & \infty & \infty & 0 & \infty & \infty & 4,5 \\ \infty & \infty & \infty & \infty & 0 & 4 & 3 \\ 2,5 & 4,5 & 7,5 & 11,5 & 4 & 0 & 7 \\ \infty & \infty & \infty & 4,5 & 3 & 7 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 6

$$W_6 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ F \\ G \\ E \end{array} \begin{array}{c} \begin{array}{ccccccc} A & B & C & D & F & G & E \end{array} \\ \begin{pmatrix} 0 & 2 & 5 & 9 & 6,5 & 2,5 & 9,5 \\ 2 & 0 & 3 & 7 & 8,5 & 4,5 & 11,5 \\ \infty & \infty & 0 & 4 & \infty & \infty & 8,5 \\ \infty & \infty & \infty & 0 & \infty & \infty & 4,5 \\ 6,5 & 8,5 & 11,5 & 15,5 & 0 & 4 & 3 \\ 2,5 & 4,5 & 7,5 & 11,5 & 4 & 0 & 7 \\ 9,5 & 11,5 & 14,5 & 4,5 & 3 & 7 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 7

$$W = W_7 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ F \\ G \\ E \end{array} \begin{array}{c} \begin{array}{ccccccc} A & B & C & D & F & G & E \end{array} \\ \begin{pmatrix} 0 & 2 & 5 & 9 & 6,5 & 2,5 & 9,5 \\ 2 & 0 & 3 & 7 & 8,5 & 4,5 & 11,5 \\ 18 & 20 & 0 & 4 & 11,5 & 15,5 & 8,5 \\ 14 & 16 & 19 & 0 & 7,5 & 11,5 & 4,5 \\ 6,5 & 8,5 & 11,5 & 7,5 & 0 & 4 & 3 \\ 2,5 & 4,5 & 7,5 & 11,5 & 4 & 0 & 7 \\ 9,5 & 11,5 & 14,5 & 4,5 & 3 & 7 & 0 \end{pmatrix} \end{array}$$

Dari perhitungan dengan k hingga 7, dapat disimpulkan bahwa pemecahan jalur terpendek menggunakan perhitungan Algoritma

Floyd-Warshall adalah 9,5 km ($A \rightarrow G \rightarrow F \rightarrow E$).

2.3.3 Algoritma A*

Pada tahun 1964, Nils Nilsson pertama kali mengembangkan algoritma A* berdasarkan Algoritma Dijkstra. Pertama kali algoritma ini diberi nama dengan Algoritma $A1$. Lalu pada tahun 1967, ilmuan bernama Bertram R. mengoptimalkan lebih lanjut algoritma terdahulu dan diberi nama dengan $A2$. Kemudian Peter E. Hant menunjukan bukti bahwa kurang optimalnya Algoritma $A1$ dibandingkan dengan Algoritma $A2$. Akhirnya, Algoritma $A2$ dikemukakan sebagai algoritma terbaik dan ideal pada permasalahan tersebut dan diberi nama dengan A* atau A-Star. Dalam pengujian menurut waktu, Algoritma A* ideal dan terbaik dibandingkan Algoritma Dijkstra (Munir, 2008).

Algoritma A* adalah karakter algoritma pencarian yang paling baik dan terkenal, algoritma tersebut biasa dipergunakan dalam penemuan jalur terpendek dari titik asal menuju ke titik yang akan dituju. Algoritma A* dapat memperediksi jumlah heuristik $h(x)$ rute terbaik yang akan dilewati oleh tiap titiknya (Mansuri dkk, 2018). Algoritma A* berfungsi mirip dengan algoritma Dijkstra kecuali algoritmanya menggunakan kontrol heuristik dalam memilih titik untuk setiap iterasi.

Dalam pemilihan jalur terpendek, Algoritma A* akan memilih titik berdasarkan jaraknya jalur dari titik awal dengan penambahan heuristik estimasi kedekatannya dengan titik tujuan. Heuristik estimasi dievaluasi oleh salah satu dari dua utama fungsi evaluasi, yaitu Jarak Euclidean dan jarak Manhattan (Chartrand dkk, 2005). Jarak Euclidean dihitung dengan panjang garis lurus (diagonal) antara titik yang dievaluasi dan titik tujuan,

sedangkan jarak Manhattan hanya diperhitungkan perbedaan sumbu koordinat (horizontal dan vertikal). Melalui penggunaan ini heuristik, Algoritma A* telah mengurangi ruang pencarian diperlukan untuk mencapai titik tujuan dibandingkan dengan Algoritma Dijkstra. Hal ini menunjukkan bahwa Algoritma A* akan memiliki performa atau kemampuan yang baik dan unggul dibandingkan dengan Algoritma Dijkstra kecuali heuristiknya kurang tepat (Munir, 2008). Algoritma ini akan ditunjukkan sebagai berikut.

$$f(n) = g(n) + h(n) \tag{2.2}$$

Keterangan:

$f(n)$: jarak estimasi terendah
$g(n)$: jarak titik awal menuju titik *n*
$h(n)$: jarak perkiraan titik n menuju titik tujuan.

Dalam memperkirakan heuristik atau jarak perkiraan $h(n)$ oleh salah satu dari dua utama fungsi evaluasi, yaitu Euclidean jarak dan jarak Manhattan. Akan ditunjukkan rumus untuk mengitung jarak Euclidean antara dua titik (x_1, y_1) dan (x_2, y_2) dalam ruang kartesian adalah sebagai berikut (Diestel, 2000):

$$h(n) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2.3}$$

Untuk dua titik (x_1, y_1) dan (x_2, y_2), jarak Manhattan dihitung dengan rumus berikut ini

$$h(n) = |x_2 - x_1| + |y_2 - y_1| \tag{2.4}$$

dengan:

x_1 dan y_1 : koordinat titik pertama

x_2 dan y_2 : koordinat titik kedua atau titik tujuan

Algorithm 3 Algoritma A*

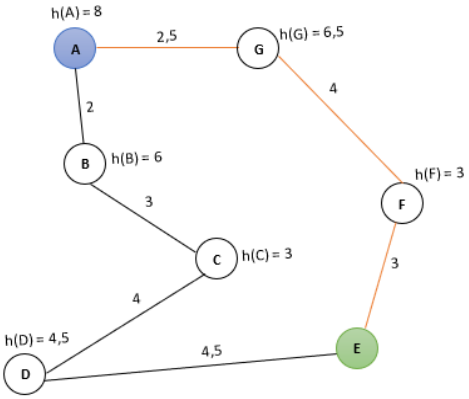
1. **Inisialisasi**: Mulai dengan titik awal, tambahkan ke *open list*. Setel $f(n)$, $g(n)$, dan $h(n)$ ke tak terhingga, kecuali untuk titik awal ($g(n) = 0$ dan $h(n)$ adalah perkiraan biaya ke tujuan).
2. **Iterasi**: Selama *open list* tidak kosong:
   (a) Pilih titik dengan $f(n)$ terendah dari *open list*.
   (b) Pindahkan titik ke *close list*.
   (c) Jika titik tujuan ditemukan, pencarian selesai.
   (d) Jika tidak, ekspansi titik dengan mengeksplorasi tetangga-tetangganya.
   (e) Untuk setiap tetangga:
      i. Hitung $g(n)$ dan $h(n)$.
      ii. Abaikan jika tetangga ada di *close list* dengan biaya lebih rendah.
      iii. Jika tidak ada di *open list* atau memiliki biaya lebih rendah, perbarui $f(n)$, $g(n)$, dan $h(n)$, dan tambahkan ke *open list*.
3. **Penyimpulan**: Jika *open list* kosong dan titik tujuan tidak ditemukan, jalur tidak ada dan pencarian gagal.

Keterangan:

Open List : Titik-titik yang belum dievaluasi, dimana titik dengan nilai $f(n)$ terendah akan dievaluasi berikutnya

Close List : Titik yang sudah dievaluasi dan tidak akan dievaluasi kembali.

Contoh 2.3.3 *Pada Gambar 2.17 di bawah ini, terdapat simulasi rute yang akan dihitung menggunakan Algoritma A* dan telah memiliki heuristik atau jarak dari titik n menuju titik tujuan. Dalam gambar terdapat titik berwarna biru yang disimbolkan dengan A diandaikan sebuah titik mulai dan titik berwarna hijau E sebagai titik tujuan. Selanjutnya, akan dihitung jarak yang dibutuhkan dimulai dari titik mulai A yang mengarah pada titik tujuan E dengan menggunakan perhitungan Algoritma A*. Dalam percobaan ini penggunaan jarak heuristik adalah dengan estimasi yang akan diuji coba dalam algoritma A*.*



Gambar 2.17. Simulasi Rute dengan Heuristik

Berikut ini merupakan perhitungan secara manual pada simulasi graf di atas.

1. *Inisialisasi*

   - *Titik awal :* A
   - *Titik tujuan :* E
   - $g(A) = 0$
   - $h(A) = 8$
   - $f(A) = g(A) + h(A) = 0 + 8 = 8$

2. *Iterasi 1 (Evaluasi titik* A*)*
   Titik A *bertetangga dengan titik* B *dan titik* G *sehingga dimasukkan ke dalam open list.*

   [label=)]Titik B

   (a)
   - $g(B) = g(A) + 2 = 0 + 2 = 2$
   - $h(B) = 6$
   - $f(B) = g(B) + h(B) = 2 + 6 = 8$

   (b) Titik G

   - $g(G) = g(A) + 2.5 = 0 + 2.5 = 2.5$
   - $h(G) = 6.5$
   - $f(G) = g(G) + h(G) = 2.5 + 6.5 = 9$

3. *Iterasi 2 (Evaluasi titik* B*)*
   Titik B *bertetangga dengan titik* C *sehingga dimasukkan ke dalam open list.*

   [label=)]Titik C

   (a)
   - $g(C) = g(B) + 3 = 2 + 3 = 5$
   - $h(C) = 3$

- $f(C) = g(C) + h(C) = 5 + 3 = 8$

4. *Iterasi 3 (Evaluasi titik G)*
   Titik G bertetangga dengan titik F sehingga dimasukkan ke dalam open list.

   [label=)]Titik F

   (a)
   - $g(F) = g(G) + 4 = 2.5 + 4 = 6.5$
   - $h(F) = 3$
   - $f(F) = g(F) + h(F) = 6.5 + 3 = 9.5$

   Evaluasi untuk titik G terhenti karena titik B dan titik setelah B memiliki nilai $f(n)$ lebih kecil dibandingkan titik G dan titik setelahnya. Maka, iterasi akan dilanjutkan melewati titik B ke titik selanjutnya.

5. *Iterasi 4 (Evaluasi titik C)*
   Titik C bertetangga dengan titik D sehingga dimasukkan ke dalam open list.

   [label=)]Titik D

   (a)
   - $g(D) = g(C) + 4 = 5 + 4 = 9$
   - $h(D) = 4.5$
   - $f(D) = g(D) + h(D) = 9 + 4.5 = 13.5$

6. *Iterasi 5 (Evaluasi titik D)*
   Titik D bertetangga dengan titik I sehingga dimasukkan ke dalam open list.

   [label=)]Titik I

   (a)
   - $g(E) = g(D) + 4.5 = 9 + 4.5 = 13.5$
   - $h(E) = 0$
   - $f(E) = g(E) + h(E) = 13.5 + 0 = 13.5$

Tabel 2.2. *Closed List* Titik $A \rightarrow E$

Via	**Tujuan**	***f(n) = g(n) + h(n)***	**Jarak**
A	A	8	0
A	B	8	2
B	C	8	3
C	D	13.5	4
D	E	13.5	4.5

Berdasarkan perhitungan di atas, ternyata jarak titik mulai A menuju titik tujuan E. Perhitungan jarak titik A menuju titik G melalui titik B sebesar 13,5 km dari titik $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$.

BAB III

HASIL DAN PEMBAHASAN PENELITIAN

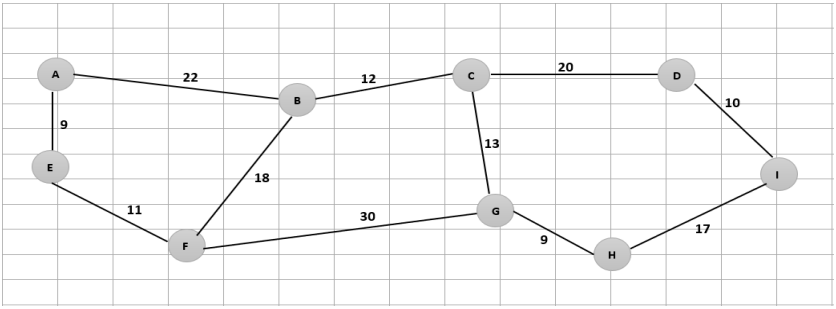
3.1 Spesifikasi Sistem

Dalam penelitian ini, proses penelitian dilakukan dengan perhitungan manual dan dibantu oleh program yang dikembangkan menggunakan Bahasa C sebagai pengecekan perhitungan manual serta tampilan untuk menghasilkan sebuah rute terpendek. Penelitian tersebut di uji coba dengan beberapa tes, yaitu total jarak terpendek, banyak titik, dan kecepatan dalam ketiga algoritma yang digunakan dalam penelitian. Adapun beberapa pendukung pengujian penelitian sebagai berikut:

1. *Hardware*
   (a) Lenovo IdeaPad S145-14IWL
   (b) Prosessor Intel(R) Core(TM) i5-1035G4 CPU 1.10GHz 1.50GHz
   (c) RAM 12.0 GB
2. *Software*
   (a) Sistem Operasi Windows 11 64Bit
   (b) CodeBlocks 20.03
   (c) Bahasa pemprograman $C/C++$
   (d) Microsoft Word Office 16

3.2 Kasus Rute Terpendek

Dalam penelitian ini dibutuhkan kasus untuk diterapkan dalam metode graf dan algoritma yang akan diuji pada pencarian rute terpendek. Kasus yang akan digunakan dalam penelitian ini adalah sebagai berikut.



Gambar 3.1. Graf

Dalam pengujian penelitian, terdapat 9 titik dan 11 sisi yang digambarkan dalam graf di atas. Untuk mencapai hasil yang akurat, setiap algoritma akan dilakukan percobaan sebanyak 3 (tiga) kali untuk setiap kasus percobaan. Masing-masing algoritma akan diuji dengan titik awal yang berbeda dan titik tujuan yang sama sehingga dapat menunjukkan kinerja masing-masing algoritma untuk menyelesaikan masalah rute terpendek dalam hal waktu berjalannya algoritma, banyaknya titik yang dilewati dan total jarak. Hasil dari percobaan akan dibagi menjadi tiga, yaitu data waktu berjalan, data banyak titik, dan data hasil jarak.

3.3 Analisis Algoritma

Dalam penelitian kali ini, proses analisis dibagi menjadi dua bagian. Pertama, perhitungan pada masing-masing algoritma dengan tiga titik awal yang berbeda dan titik tujuan yang sama. Lalu, akan dilanjutkan dengan perbandingan antara hasil dari perhitungan tiga buah algoritma tersebut dalam tiga titik yang berbeda. Tiga buah titik uji percobaan antara lain:

1. Percobaan pada titik A menuju titik I ($A \rightarrow I$)
2. Percobaan Pada titik B menuju titik I ($B \rightarrow I$)
3. Percobaan pada titik E menuju titik I ($E \rightarrow I$)

3.3.1 Perhitungan Algoritma Dijkstra

Algortima Dijkstra adalah salah satu algoritma yang digunakan dalam penentuan rute terpendek dalam graf. Cara kerja algoritma ini, dimana setiap langkah akan menemukan sebuah titik dengan bobot paling kecil dari titik awal, serta titik tersebut akan ditentukan sebagai titik tetap dan sudah tidak dipertimbangkan lagi dalam langkah selanjutnya. Perhitungan menggunakan algoritma ini sebagai berikut.

1. Dalam analisis menggunakan Algoritma Dijkstra, akan dihitung graf dengan titik awalnya A dan titik tujuannya I. Perhitungan akan dijabarkan dan ditunjukkan di bawah ini.

Tabel 3.1. Perhitungan titik A menuju titik I

V	A	B	C	D	E	F	G	H	I
A	**0-A**	22-A			9-A				
E		22-A			**9-A**	20-E			
F		22-A	34-B			**20-E**	50-F		
B		**22-A**	34-B				50-F		
C			**34-B**	54-C			47-C		
G				54-C			**47-C**	56-G	
D				**54-C**				56-G	64-D
H								**56-G**	64-D
I									**64-D**

Selain menggunakan perhitungan dengan cara manual, diberikan bantuan dengan menggunakan program C untuk membantu dalam proses pengolahan Algoritma Dijkstra dari titik awal A menuju titik tujuan I ditunjukkan berikut ini.

```
Jarak terpendek titik 0 ke titik 1 = 22
Jalur terpendek: 1 <- 0

Jarak terpendek titik 0 ke titik 2 = 34
Jalur terpendek: 2 <- 1 <- 0

Jarak terpendek titik 0 ke titik 3 = 54
Jalur terpendek: 3 <- 2 <- 1 <- 0

Jarak terpendek titik 0 ke titik 4 = 9
Jalur terpendek: 4 <- 0

Jarak terpendek titik 0 ke titik 5 = 20
Jalur terpendek: 5 <- 4 <- 0

Jarak terpendek titik 0 ke titik 6 = 47
Jalur terpendek: 6 <- 2 <- 1 <- 0

Jarak terpendek titik 0 ke titik 7 = 56
Jalur terpendek: 7 <- 6 <- 2 <- 1 <- 0

Jarak terpendek titik 0 ke titik 8 = 64
Jalur terpendek: 8 <- 3 <- 2 <- 1 <- 0

Total Jaraknya adalah 64
```

Gambar 3.2. Hasil Program C/C++ Algoritma Dijkstra Titik Awal A

Dalam perhitungan menggunakan Algoritma Dijkstra dapat di lihat pada Tabel 3.1 dan Gambar 3.2, bahwa total jarak terpendek dalam graf dengan titik awal A dan titik tujuan I adalah 64. Dengan rute titik yang digunakan adalah dari titik

$A \rightarrow B \rightarrow C \rightarrow D \rightarrow I$, serta banyak titik yang dilewati adalah lima buah titik.

2. Dalam analisis menggunakan Algoritma Dijkstra, akan dihitung graf dengan titik awalnya B dan titik tujuannya I. Perhitungan akan dijabarkan dan ditunjukkan di bawah ini.

Tabel 3.2. Perhitungan titik B menuju titik I

V	B	A	C	D	E	F	G	H	I
B	**0-B**	22-B	12-B			18-B			
C		22-B	**12-B**	32-C		18-B	25-C		
F		22-B		32-C	29-F	**18-B**	25-C		
A		**22-B**		32-C	29-F		25-C		
G				32-C	29-F		**25-C**	34-G	
E				32-C	**29-F**			34-G	
D				**32-C**				34-G	42-D
H								**34-G**	42-D
I									**42-D**

Selain menggunakan perhitungan dengan cara manual, diberikan bantuan dengan menggunakan program C ditunjukkan berikut ini.

```
Jarak terpendek titik 1 ke titik 0 = 22
Jalur terpendek: 0 <- 1

Jarak terpendek titik 1 ke titik 2 = 12
Jalur terpendek: 2 <- 1

Jarak terpendek titik 1 ke titik 3 = 32
Jalur terpendek: 3 <- 2 <- 1

Jarak terpendek titik 1 ke titik 4 = 29
Jalur terpendek: 4 <- 5 <- 1

Jarak terpendek titik 1 ke titik 5 = 18
Jalur terpendek: 5 <- 1

Jarak terpendek titik 1 ke titik 6 = 25
Jalur terpendek: 6 <- 2 <- 1

Jarak terpendek titik 1 ke titik 7 = 34
Jalur terpendek: 7 <- 6 <- 2 <- 1

Jarak terpendek titik 1 ke titik 8 = 42
Jalur terpendek: 8 <- 3 <- 2 <- 1

Total Jaraknya adalah 42
```

Gambar 3.3. Hasil Program C/C++ Algoritma Dijkstra Titik Awal B

Dalam perhitungan menggunakan Algoritma Dijkstra dapat di lihat pada Tabel 3.2 dan Gambar 3.3, bahwa total jarak terpendek dalam graf dengan titik awal B dan titik tujuan I adalah 42. Dengan rute titik yang digunakan adalah dari titik $B \rightarrow C \rightarrow D \rightarrow I$, serta banyak titik yang dilewati adalah empat buah titik.

3. Dalam analisis menggunakan Algoritma Dijkstra, akan dihitung graf dengan titik awalnya E dan titik tujuannya I. Perhitungan akan dijabarkan dan ditunjukkan di bawah ini.

Tabel 3.3. Perhitungan titik E menuju titik I

V	E	A	B	C	D	F	G	H	I
E	**0-E**	9-E				11-E			
A		**9-E**	31-A			11-E			
F			29-F			**11-E**	41-F		
B			**29-F**	41-B			41-F		
C				**41-B**	61-C		41-F		
G					61-C		**41-F**	50-G	
H					61-C			**50-G**	67-H
D					**61-C**				67-H
I									**67-H**

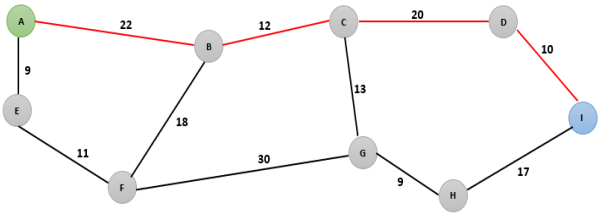
Selain menggunakan perhitungan dengan cara manual, diberikan bantuan dengan menggunakan program C untuk membantu dalam proses pengolahan Algoritma Dijkstra dari titik awal E menuju titik tujuan I ditunjukkan berikut ini.

```
Jarak terpendek titik 4 ke titik 0 = 9
Jalur terpendek: 0 <- 4

Jarak terpendek titik 4 ke titik 1 = 29
Jalur terpendek: 1 <- 5 <- 4

Jarak terpendek titik 4 ke titik 2 = 41
Jalur terpendek: 2 <- 1 <- 5 <- 4

Jarak terpendek titik 4 ke titik 3 = 61
Jalur terpendek: 3 <- 2 <- 1 <- 5 <- 4

Jarak terpendek titik 4 ke titik 5 = 11
Jalur terpendek: 5 <- 4

Jarak terpendek titik 4 ke titik 6 = 41
Jalur terpendek: 6 <- 5 <- 4

Jarak terpendek titik 4 ke titik 7 = 50
Jalur terpendek: 7 <- 6 <- 5 <- 4

Jarak terpendek titik 4 ke titik 8 = 67
Jalur terpendek: 8 <- 7 <- 6 <- 5 <- 4

Total Jaraknya adalah 67
```

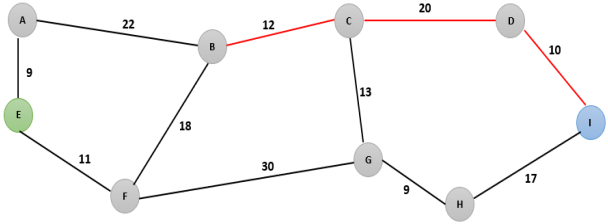
Gambar 3.4. Hasil Program C/C++ Algoritma Dijkstra Titik Awal E

Dalam perhitungan menggunakan Algoritma Dijkstra diketahui pada Tabel 3.3 dan Gambar 3.4, total jarak terpendek dalam graf dengan titik awal E dan titik tujuan I adalah 67. Dengan rute titik yang digunakan adalah titik $E \rightarrow F \rightarrow G \rightarrow H \rightarrow I$, serta banyak titik ialah lima titik.

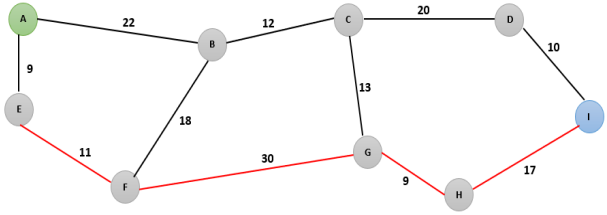
Berikut ini akan ditunjukan hasil rute terpendek yang terpilih menggunakan Algoritma Dijkstra dari titik awal A, B dan E menuju titik tujuan I.



Gambar 3.5. Rute Terpilih Titik $A \rightarrow I$



Gambar 3.6. Rute Terpilih Titik $B \rightarrow I$



Gambar 3.7. Rute Terpilih Titik $E \rightarrow I$

3.3.2 Perhitungan Algoritma Floyd-Warshall

Selain, perhitungan graf menggunakan Algoritma Dijkstra penulis juga akan menggunakan Algoritma Floyd-Warshall sebagai salah satu algoritma yang akan diteliti. Algoritma Floyd-Warshall merupakan algoritma yang digunakan untuk memperkirakan bobot terkecil dari semua pasangan titik yang menyambung dan menjalankan langsung pada seluruh pasangan titik.

1. Pada perhitungan graf menggunakan Algoritma Floyd-Warshall, graf akan dihitung dengan titik awalnya A dan titik tujuannya I. Sebelum menghitung rute terpendek, perlu dibuat gambaran graf dalam bentuk matriks. Representasi matriks titik awal A menuju titik tujuan I dan perhitungan rute terpendek menggunakan

Algoritma Floyd-Warshall akan dijabarkan berikut ini.

Iterasi k = 0

$$W = W_0 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & \infty & \infty & 9 & \infty & \infty & \infty & \infty \\ 22 & 0 & 12 & \infty & \infty & 18 & \infty & \infty & \infty \\ \infty & 12 & 0 & 20 & \infty & \infty & 13 & \infty & \infty \\ \infty & \infty & 20 & 0 & \infty & \infty & \infty & \infty & 10 \\ 9 & \infty & \infty & \infty & 0 & 11 & \infty & \infty & \infty \\ \infty & 18 & \infty & \infty & 11 & 0 & 30 & \infty & \infty \\ \infty & \infty & 13 & \infty & \infty & 30 & 0 & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ \infty & \infty & \infty & 10 & \infty & \infty & \infty & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 1

$$W_1 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & \infty & \infty & 9 & \infty & \infty & \infty & \infty \\ 22 & 0 & 12 & \infty & 31 & 18 & \infty & \infty & \infty \\ \infty & 12 & 0 & 20 & \infty & \infty & 13 & \infty & \infty \\ \infty & \infty & 20 & 0 & \infty & \infty & \infty & \infty & 10 \\ 9 & 31 & \infty & \infty & 0 & 11 & \infty & \infty & \infty \\ \infty & 18 & \infty & \infty & 11 & 0 & 30 & \infty & \infty \\ \infty & \infty & 13 & \infty & \infty & 30 & 0 & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ \infty & \infty & \infty & 10 & \infty & \infty & \infty & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 2

$$W_2 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & 34 & \infty & 9 & 40 & \infty & \infty & \infty \\ 22 & 0 & 12 & \infty & 31 & 18 & \infty & \infty & \infty \\ 34 & 12 & 0 & 20 & 43 & 30 & 13 & \infty & \infty \\ \infty & \infty & 20 & 0 & \infty & \infty & \infty & \infty & 10 \\ 9 & 31 & 43 & \infty & 0 & 11 & \infty & \infty & \infty \\ 40 & 18 & 30 & \infty & 11 & 0 & 30 & \infty & \infty \\ \infty & \infty & 13 & \infty & \infty & 30 & 0 & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ \infty & \infty & \infty & 10 & \infty & \infty & \infty & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 3

$$W_3 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & 34 & 54 & 9 & 40 & 47 & \infty & \infty \\ 22 & 0 & 12 & 32 & 31 & 18 & 25 & \infty & \infty \\ 34 & 12 & 0 & 20 & 43 & 30 & 13 & \infty & \infty \\ 54 & 32 & 20 & 0 & 63 & 50 & 33 & \infty & 10 \\ 9 & 31 & 43 & 63 & 0 & 11 & 56 & \infty & \infty \\ 40 & 18 & 30 & 50 & 11 & 0 & 30 & \infty & \infty \\ 47 & 25 & 13 & 33 & 56 & 30 & 0 & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ \infty & \infty & \infty & 10 & \infty & \infty & \infty & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 4

$$W_4 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & 34 & 54 & 9 & 40 & 47 & \infty & 64 \\ 22 & 0 & 12 & 32 & 31 & 18 & 25 & \infty & 42 \\ 34 & 12 & 0 & 20 & 43 & 30 & 13 & \infty & 30 \\ 54 & 32 & 20 & 0 & 63 & 50 & 33 & \infty & 10 \\ 9 & 31 & 43 & 63 & 0 & 11 & 56 & \infty & 73 \\ 40 & 18 & 30 & 50 & 11 & 0 & 30 & \infty & 60 \\ 47 & 25 & 13 & 33 & 56 & 30 & 0 & 9 & 43 \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ 64 & 42 & 30 & 10 & 73 & 60 & 43 & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 5

$$W_5 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & 34 & 54 & 9 & 20 & 47 & \infty & 64 \\ 22 & 0 & 12 & 32 & 31 & 18 & 25 & \infty & 42 \\ 34 & 12 & 0 & 20 & 43 & 30 & 13 & \infty & 30 \\ 54 & 32 & 20 & 0 & 63 & 50 & 33 & \infty & 10 \\ 9 & 31 & 43 & 63 & 0 & 11 & 56 & \infty & 73 \\ 20 & 18 & 30 & 50 & 11 & 0 & 30 & \infty & 60 \\ 47 & 25 & 13 & 33 & 56 & 30 & 0 & 9 & 43 \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ 64 & 42 & 30 & 10 & 73 & 60 & 43 & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 6

$$W_6 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & 34 & 54 & 9 & 20 & 47 & \infty & 64 \\ 22 & 0 & 12 & 32 & 29 & 18 & 25 & \infty & 42 \\ 34 & 12 & 0 & 20 & 41 & 30 & 13 & \infty & 30 \\ 54 & 32 & 20 & 0 & 61 & 50 & 33 & \infty & 10 \\ 9 & 29 & 41 & 61 & 0 & 11 & 41 & \infty & 71 \\ 20 & 18 & 30 & 50 & 11 & 0 & 30 & \infty & 60 \\ 47 & 25 & 13 & 33 & 41 & 30 & 0 & 9 & 43 \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ 64 & 42 & 30 & 10 & 71 & 60 & 43 & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 7

$$W_7 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & 34 & 54 & 9 & 20 & 47 & 56 & 64 \\ 22 & 0 & 12 & 32 & 29 & 18 & 25 & 34 & 42 \\ 34 & 12 & 0 & 20 & 41 & 30 & 13 & 22 & 30 \\ 54 & 32 & 20 & 0 & 61 & 50 & 33 & 42 & 10 \\ 9 & 29 & 41 & 61 & 0 & 11 & 41 & 50 & 71 \\ 20 & 18 & 30 & 50 & 11 & 0 & 30 & 39 & 60 \\ 47 & 25 & 13 & 33 & 41 & 30 & 0 & 9 & 43 \\ 56 & 34 & 22 & 42 & 50 & 39 & 9 & 0 & 17 \\ 64 & 42 & 30 & 10 & 71 & 60 & 43 & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 8

$$W_8 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & 34 & 54 & 9 & 20 & 47 & 56 & 64 \\ 22 & 0 & 12 & 32 & 29 & 18 & 25 & 34 & 42 \\ 34 & 12 & 0 & 20 & 41 & 30 & 13 & 22 & 30 \\ 54 & 32 & 20 & 0 & 61 & 50 & 33 & 42 & 10 \\ 9 & 29 & 41 & 61 & 0 & 11 & 41 & 50 & 67 \\ 20 & 18 & 30 & 50 & 11 & 0 & 30 & 39 & 56 \\ 47 & 25 & 13 & 33 & 41 & 30 & 0 & 9 & 26 \\ 56 & 34 & 22 & 42 & 50 & 39 & 9 & 0 & 17 \\ 64 & 42 & 30 & 10 & 67 & 56 & 26 & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 9

$$W_9 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & 34 & 54 & 9 & 20 & 47 & 56 & \mathbf{64} \\ 22 & 0 & 12 & 32 & 29 & 18 & 25 & 34 & 42 \\ 34 & 12 & 0 & 20 & 41 & 30 & 13 & 22 & 30 \\ 54 & 32 & 20 & 0 & 61 & 50 & 33 & 27 & 10 \\ 9 & 29 & 41 & 61 & 0 & 11 & 41 & 50 & 67 \\ 20 & 18 & 30 & 50 & 11 & 0 & 30 & 39 & 56 \\ 47 & 25 & 13 & 33 & 41 & 30 & 0 & 9 & 26 \\ 56 & 34 & 22 & 27 & 50 & 39 & 9 & 0 & 17 \\ 64 & 42 & 30 & 10 & 67 & 56 & 26 & 17 & 0 \end{pmatrix} \end{array}$$

Dalam perhitungan W_9 = $W*$ merupakan matriks jarak terpendek dari seluruh titik ke seluruh pasangan titik. Pada iterasi terakhir (iterasi k = 9), Algoritma Floyd-Warshall menelusuri rute dari titik awal A menuju titik tujuan I adalah melalui titik $A \rightarrow B \rightarrow C \rightarrow D \rightarrow I$. Dalam perhitungan tersebut ditunjukkan hasil total jaraknya adalah 64.

2. Pada perhitungan graf menggunakan Algoritma Floyd-Warshall, graf akan dihitung dengan titik awalnya B dan titik tujuannya I. Sebelum menghitung rute terpendek, perlu dibuat gambaran graf dalam bentuk matriks. Perhitungan rute terpendek menggunakan Algoritma Floyd-Warshall akan dijabarkan berikut ini.

Iterasi k = 0

$$W = W_0 = \begin{array}{c} \\ B \\ A \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & 12 & \infty & \infty & 18 & \infty & \infty & \infty \\ 22 & 0 & \infty & \infty & 9 & \infty & \infty & \infty & \infty \\ 12 & \infty & 0 & 20 & \infty & \infty & 13 & \infty & \infty \\ \infty & \infty & 20 & 0 & \infty & \infty & \infty & \infty & 10 \\ \infty & 9 & \infty & \infty & 0 & 11 & \infty & \infty & \infty \\ 18 & \infty & \infty & \infty & 11 & 0 & 30 & \infty & \infty \\ \infty & \infty & 13 & \infty & \infty & 30 & 0 & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ \infty & \infty & \infty & 10 & \infty & \infty & \infty & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 1

$$W_1 = \begin{array}{c} \\ B \\ A \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & 12 & \infty & \infty & 18 & \infty & \infty & \infty \\ 22 & 0 & 34 & \infty & 9 & 40 & \infty & \infty & \infty \\ 12 & 34 & 0 & 20 & \infty & 30 & 13 & \infty & \infty \\ \infty & \infty & 20 & 0 & \infty & \infty & \infty & \infty & 10 \\ \infty & 9 & \infty & \infty & 0 & 11 & \infty & \infty & \infty \\ 18 & 40 & 30 & \infty & 11 & 0 & 30 & \infty & \infty \\ \infty & \infty & 13 & \infty & \infty & 30 & 0 & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ \infty & \infty & \infty & 10 & \infty & \infty & \infty & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 2

$$W_2 = \begin{array}{c} \\ B \\ A \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & 12 & \infty & 31 & 18 & \infty & \infty & \infty \\ 22 & 0 & 34 & \infty & 9 & 40 & \infty & \infty & \infty \\ 12 & 34 & 0 & 20 & 43 & 30 & 13 & \infty & \infty \\ \infty & \infty & 20 & 0 & \infty & \infty & \infty & \infty & 10 \\ 31 & 9 & 43 & \infty & 0 & 11 & \infty & \infty & \infty \\ 18 & 40 & 30 & \infty & 11 & 0 & 30 & \infty & \infty \\ \infty & \infty & 13 & \infty & \infty & 30 & 0 & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ \infty & \infty & \infty & 10 & \infty & \infty & \infty & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 3

$$W_3 = \begin{array}{c} \\ B \\ A \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & 12 & 32 & 31 & 18 & 25 & \infty & \infty \\ 22 & 0 & 34 & 54 & 9 & 40 & 47 & \infty & \infty \\ 12 & 34 & 0 & 20 & 43 & 30 & 13 & \infty & \infty \\ 32 & 54 & 20 & 0 & 63 & 50 & 33 & \infty & 10 \\ 31 & 9 & 43 & 63 & 0 & 11 & 56 & \infty & \infty \\ 18 & 40 & 30 & 50 & 11 & 0 & 30 & \infty & \infty \\ 25 & 47 & 13 & 33 & 56 & 30 & 0 & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ \infty & \infty & \infty & 10 & \infty & \infty & \infty & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 4

$$W_4 = \begin{array}{c} \\ B \\ A \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & 12 & 32 & 31 & 18 & 25 & \infty & 42 \\ 22 & 0 & 34 & 54 & 9 & 40 & 47 & \infty & 64 \\ 12 & 34 & 0 & 20 & 43 & 30 & 13 & \infty & 30 \\ 32 & 54 & 20 & 0 & 63 & 50 & 33 & \infty & 10 \\ 31 & 9 & 43 & 63 & 0 & 11 & 56 & \infty & 73 \\ 18 & 40 & 30 & 50 & 11 & 0 & 30 & \infty & 60 \\ 25 & 47 & 13 & 33 & 56 & 30 & 0 & 9 & 43 \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ 42 & 64 & 30 & 10 & 73 & 60 & 43 & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 5

$$W_5 = \begin{array}{c} \\ B \\ A \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & 12 & 32 & 31 & 18 & 25 & \infty & 42 \\ 22 & 0 & 34 & 54 & 9 & 20 & 47 & \infty & 64 \\ 12 & 34 & 0 & 20 & 43 & 30 & 13 & \infty & 30 \\ 32 & 54 & 20 & 0 & 63 & 50 & 33 & \infty & 10 \\ 31 & 9 & 43 & 63 & 0 & 11 & 56 & \infty & 73 \\ 18 & 20 & 30 & 50 & 11 & 0 & 30 & \infty & 60 \\ 25 & 47 & 13 & 33 & 56 & 30 & 0 & 9 & 43 \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ 42 & 64 & 30 & 10 & 73 & 60 & 43 & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 6

$$W_6 = \begin{array}{c} \\ B \\ A \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & 12 & 32 & 29 & 18 & 25 & \infty & 42 \\ 22 & 0 & 34 & 54 & 9 & 20 & 47 & \infty & 64 \\ 12 & 34 & 0 & 20 & 41 & 30 & 13 & \infty & 30 \\ 32 & 54 & 20 & 0 & 61 & 50 & 33 & \infty & 10 \\ 29 & 9 & 41 & 61 & 0 & 11 & 41 & \infty & 71 \\ 18 & 20 & 30 & 50 & 11 & 0 & 30 & \infty & 60 \\ 25 & 47 & 13 & 33 & 41 & 30 & 0 & 9 & 43 \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ 42 & 64 & 30 & 10 & 71 & 60 & 43 & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 7

$$W_7 = \begin{array}{c} \\ B \\ A \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & 12 & 32 & 29 & 18 & 25 & 34 & 42 \\ 22 & 0 & 34 & 54 & 9 & 20 & 47 & 56 & 64 \\ 12 & 34 & 0 & 20 & 41 & 30 & 13 & 22 & 30 \\ 32 & 54 & 20 & 0 & 61 & 50 & 33 & 42 & 10 \\ 29 & 9 & 41 & 61 & 0 & 11 & 41 & 50 & 71 \\ 18 & 20 & 30 & 50 & 11 & 0 & 30 & 39 & 60 \\ 25 & 47 & 13 & 33 & 41 & 30 & 0 & 9 & 43 \\ 34 & 56 & 22 & 42 & 50 & 39 & 9 & 0 & 17 \\ 42 & 64 & 30 & 10 & 71 & 60 & 43 & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 8

$$W_8 = \begin{array}{c} \\ B \\ A \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & 12 & 32 & 29 & 18 & 25 & 34 & 42 \\ 22 & 0 & 34 & 54 & 9 & 20 & 47 & 56 & 64 \\ 12 & 34 & 0 & 20 & 41 & 30 & 13 & 22 & 30 \\ 32 & 54 & 20 & 0 & 61 & 50 & 33 & 42 & 10 \\ 29 & 9 & 41 & 61 & 0 & 11 & 41 & 50 & 67 \\ 18 & 20 & 30 & 50 & 11 & 0 & 30 & 39 & 56 \\ 25 & 47 & 13 & 33 & 41 & 30 & 0 & 9 & 26 \\ 34 & 56 & 22 & 42 & 50 & 39 & 9 & 0 & 17 \\ 42 & 64 & 30 & 10 & 67 & 56 & 26 & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 9

$$W_9 = \begin{array}{c} \\ B \\ A \\ C \\ D \\ E \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} A & B & C & D & E & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 22 & 12 & 32 & 29 & 18 & 25 & 34 & \mathbf{42} \\ 22 & 0 & 34 & 54 & 9 & 20 & 47 & 56 & 64 \\ 12 & 34 & 0 & 20 & 41 & 30 & 13 & 22 & 30 \\ 32 & 54 & 20 & 0 & 61 & 50 & 33 & 27 & 10 \\ 29 & 9 & 41 & 61 & 0 & 11 & 41 & 50 & 67 \\ 18 & 20 & 30 & 50 & 11 & 0 & 30 & 39 & 56 \\ 25 & 47 & 13 & 33 & 41 & 30 & 0 & 9 & 26 \\ 34 & 56 & 22 & 27 & 50 & 39 & 9 & 0 & 17 \\ 42 & 64 & 30 & 10 & 67 & 56 & 26 & 17 & 0 \end{pmatrix} \end{array}$$

Dalam perhitungan W_9 = $W*$ merupakan matriks jarak terpendek dari seluruh titik ke seluruh pasangan titik. Pada iterasi terakhir (iterasi k = 9), Algoritma Floyd-Warshall menelusuri rute dari titik awal B menuju titik tujuan I adalah melalui titik $B \rightarrow C \rightarrow D \rightarrow I$. Dalam perhitungan tersebut ditunjukkan hasil total jaraknya adalah 42.

3. Pada perhitungan graf menggunakan Algoritma Floyd-Warshall, graf akan dihitung dengan titik awalnya E dan titik tujuannya I. Sebelum menghitung rute terpendek, perlu dibuat gambaran graf dalam bentuk matriks. Representasi matriks dan perhitungan rute terpendek menggunakan Algoritma Floyd-Warshall dijabarkan berikut.

Iterasi k = 0

$$W = W_0 = \begin{array}{c} \\ E \\ A \\ B \\ C \\ D \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} E & A & B & C & D & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 9 & \infty & \infty & \infty & 11 & \infty & \infty & \infty \\ 9 & 0 & 22 & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 22 & 0 & 12 & \infty & 18 & \infty & \infty & \infty \\ \infty & \infty & 12 & 0 & 20 & \infty & 13 & \infty & \infty \\ \infty & \infty & \infty & 20 & 0 & \infty & \infty & \infty & 10 \\ 30 & \infty & 18 & \infty & \infty & 0 & 30 & \infty & \infty \\ \infty & \infty & \infty & 13 & \infty & 30 & 0 & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ \infty & \infty & \infty & \infty & 10 & \infty & \infty & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 1

$$W_1 = \begin{array}{c} \\ E \\ A \\ B \\ C \\ D \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} E & A & B & C & D & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 9 & \infty & \infty & \infty & 11 & \infty & \infty & \infty \\ 9 & 0 & 22 & \infty & \infty & 20 & \infty & \infty & \infty \\ \infty & 22 & 0 & 12 & \infty & 18 & \infty & \infty & \infty \\ \infty & \infty & 12 & 0 & 20 & \infty & 13 & \infty & \infty \\ \infty & \infty & \infty & 20 & 0 & \infty & \infty & \infty & 10 \\ 11 & 20 & 18 & \infty & \infty & 0 & 30 & \infty & \infty \\ \infty & \infty & \infty & 13 & \infty & 30 & 0 & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ \infty & \infty & \infty & \infty & 10 & \infty & \infty & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 2

$$W_2 = \begin{array}{c} \\ E \\ A \\ B \\ C \\ D \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} E & A & B & C & D & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 9 & 31 & \infty & \infty & 11 & \infty & \infty & \infty \\ 9 & 0 & 22 & \infty & \infty & 20 & \infty & \infty & \infty \\ 31 & 22 & 0 & 12 & \infty & 18 & \infty & \infty & \infty \\ \infty & \infty & 12 & 0 & 20 & \infty & 13 & \infty & \infty \\ \infty & \infty & \infty & 20 & 0 & \infty & \infty & \infty & 10 \\ 11 & 20 & 18 & \infty & \infty & 0 & 30 & \infty & \infty \\ \infty & \infty & \infty & 13 & \infty & 30 & 0 & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ \infty & \infty & \infty & \infty & 10 & \infty & \infty & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 3

$$W_3 = \begin{array}{c} \\ E \\ A \\ B \\ C \\ D \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} E & A & B & C & D & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 9 & 31 & 43 & \infty & 11 & \infty & \infty & \infty \\ 9 & 0 & 22 & 34 & \infty & 20 & \infty & \infty & \infty \\ 31 & 22 & 0 & 12 & \infty & 18 & \infty & \infty & \infty \\ 43 & 34 & 12 & 0 & 20 & 30 & 13 & \infty & \infty \\ \infty & \infty & \infty & 20 & 0 & \infty & \infty & \infty & 10 \\ 11 & 20 & 18 & 30 & \infty & 0 & 30 & \infty & \infty \\ \infty & \infty & \infty & 13 & \infty & 30 & 0 & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ \infty & \infty & \infty & \infty & 10 & \infty & \infty & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 4

$$W_4 = \begin{array}{c} \\ E \\ A \\ B \\ C \\ D \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} E & A & B & C & D & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 9 & 31 & 43 & 63 & 11 & 56 & \infty & \infty \\ 9 & 0 & 22 & 34 & 54 & 20 & 47 & \infty & \infty \\ 31 & 22 & 0 & 12 & 32 & 18 & 25 & \infty & \infty \\ 43 & 34 & 12 & 0 & 20 & 30 & 13 & \infty & \infty \\ 63 & 54 & 32 & 20 & 0 & 50 & 33 & \infty & 10 \\ 11 & 20 & 18 & 30 & 50 & 0 & 30 & \infty & \infty \\ 56 & 47 & 25 & 13 & 33 & 30 & 0 & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ \infty & \infty & \infty & \infty & 10 & \infty & \infty & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 5

$$W_5 = \begin{array}{c} \\ E \\ A \\ B \\ C \\ D \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} E & A & B & C & D & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 9 & 31 & 43 & 63 & 11 & 56 & \infty & 73 \\ 9 & 0 & 22 & 34 & 54 & 20 & 47 & \infty & 64 \\ 31 & 22 & 0 & 12 & 32 & 18 & 25 & \infty & 42 \\ 43 & 34 & 12 & 0 & 20 & 30 & 13 & \infty & 30 \\ 63 & 54 & 32 & 20 & 0 & 50 & 33 & \infty & 10 \\ 11 & 20 & 18 & 30 & 50 & 0 & 30 & \infty & 60 \\ 56 & 47 & 25 & 13 & 33 & 30 & 0 & 9 & 43 \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ 73 & 64 & 42 & 30 & 10 & 60 & 43 & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 6

$$W_6 = \begin{array}{c} \\ E \\ A \\ B \\ C \\ D \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} E & A & B & C & D & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 9 & 29 & 41 & 61 & 11 & 41 & \infty & 71 \\ 9 & 0 & 22 & 34 & 54 & 20 & 47 & \infty & 64 \\ 29 & 22 & 0 & 12 & 32 & 18 & 25 & \infty & 42 \\ 41 & 34 & 12 & 0 & 20 & 30 & 13 & \infty & 30 \\ 61 & 54 & 32 & 20 & 0 & 50 & 33 & \infty & 10 \\ 11 & 20 & 18 & 30 & 50 & 0 & 30 & \infty & 60 \\ 41 & 47 & 25 & 13 & 33 & 30 & 0 & 9 & 43 \\ \infty & \infty & \infty & \infty & \infty & \infty & 9 & 0 & 17 \\ 71 & 64 & 42 & 30 & 10 & 60 & 43 & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 7

$$W_7 = \begin{array}{c} \\ E \\ A \\ B \\ C \\ D \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} E & A & B & C & D & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 9 & 29 & 41 & 61 & 11 & 41 & 50 & 71 \\ 9 & 0 & 22 & 34 & 54 & 20 & 47 & 56 & 64 \\ 29 & 22 & 0 & 12 & 32 & 18 & 25 & 34 & 42 \\ 41 & 34 & 12 & 0 & 20 & 30 & 13 & 22 & 30 \\ 61 & 54 & 32 & 20 & 0 & 50 & 33 & 42 & 10 \\ 11 & 20 & 18 & 30 & 50 & 0 & 30 & 39 & 60 \\ 41 & 47 & 25 & 13 & 33 & 30 & 0 & 9 & 43 \\ 50 & 56 & 34 & 22 & 42 & 39 & 9 & 0 & 17 \\ 71 & 64 & 42 & 30 & 10 & 60 & 43 & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 8

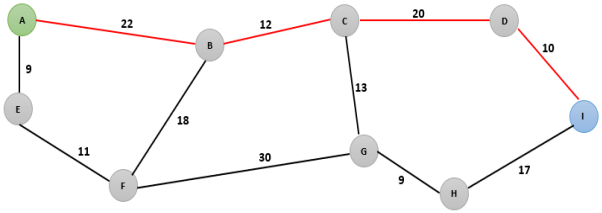
$$W_8 = \begin{array}{c} \\ E \\ A \\ B \\ C \\ D \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} E & A & B & C & D & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 9 & 29 & 41 & 61 & 11 & 41 & 50 & 67 \\ 9 & 0 & 22 & 34 & 54 & 20 & 47 & 56 & 64 \\ 29 & 22 & 0 & 12 & 32 & 18 & 25 & 34 & 42 \\ 41 & 34 & 12 & 0 & 20 & 30 & 13 & 22 & 30 \\ 61 & 54 & 32 & 20 & 0 & 50 & 33 & 42 & 10 \\ 11 & 20 & 18 & 30 & 50 & 0 & 30 & 39 & 60 \\ 41 & 47 & 25 & 13 & 33 & 30 & 0 & 9 & 26 \\ 50 & 56 & 34 & 22 & 42 & 39 & 9 & 0 & 17 \\ 67 & 64 & 42 & 30 & 10 & 60 & 26 & 17 & 0 \end{pmatrix} \end{array}$$

Iterasi k = 9

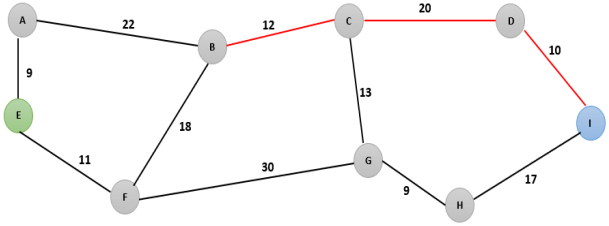
$$W_9 = \begin{array}{c} \\ E \\ A \\ B \\ C \\ D \\ F \\ G \\ H \\ I \end{array} \begin{array}{c} \begin{array}{ccccccccc} E & A & B & C & D & F & G & H & I \end{array} \\ \begin{pmatrix} 0 & 9 & 29 & 41 & 61 & 11 & 41 & 50 & \mathbf{67} \\ 9 & 0 & 22 & 34 & 54 & 20 & 47 & 56 & 64 \\ 29 & 22 & 0 & 12 & 32 & 18 & 25 & 34 & 42 \\ 41 & 34 & 12 & 0 & 20 & 30 & 13 & 22 & 30 \\ 61 & 54 & 32 & 20 & 0 & 50 & 33 & 27 & 10 \\ 11 & 20 & 18 & 30 & 50 & 0 & 30 & 39 & 56 \\ 41 & 47 & 25 & 13 & 33 & 30 & 0 & 9 & 26 \\ 50 & 56 & 34 & 22 & 27 & 39 & 9 & 0 & 17 \\ 67 & 64 & 42 & 30 & 10 & 56 & 26 & 17 & 0 \end{pmatrix} \end{array}$$

Dalam perhitungan W_9 = $W*$ merupakan matriks jarak terpendek dari seluruh titik ke seluruh pasangan titik. Pada iterasi terakhir (iterasi k = 9), Algoritma Floyd-Warshall menelusuri rute dari titik awal E menuju titik tujuan I adalah melalui titik $E \rightarrow F \rightarrow G \rightarrow H \rightarrow I$. Dalam perhitungan ditunjukkan hasil total jaraknya adalah 67.

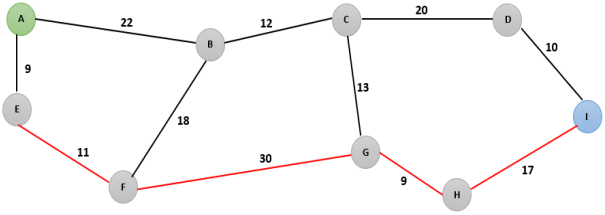
Berikut ini ditunjukkan hasil rute terpendek yang terpilih menggunakan Algoritma Floyd-Warshall dari titik awal A, B dan E menuju titik tujuan I.



Gambar 3.8. Rute Terpilih Titik $A \rightarrow I$



Gambar 3.9. Rute Terpilih Titik $B \rightarrow I$



Gambar 3.10. Rute Terpilih Titik $E \rightarrow I$

3.3.3 Perhitungan Algoritma A*

Dalam analisis untuk pengujian rute terpendek pada graf, penulis juga menggunakan Algoritma A* atau biasa disebut A-Star selain dua algoritma yang telah dijelaskan sebelumnya. Algoritma ini digunakan dalam pencarian rute terpendek dengan memilih titik berdasarkan jarak rute dari titik awal dengan menambah heuristik estimasi kedekatan dengan titik tujuan. Dalam perhitungan menggunakan algoritma ini, heuristik estimasi dievaluasi akan dihitung dengan jarak Euclidean yang mana akan diperkirakan atau diprediksi dengan panjang garis lurus antara titik yang dievaluasi dengan titik tujuan. Sebelum perhitungan rute terpendek, akan ditunjukkan di bawah ini jarak garis lurus dari suatu titik menuju titik tujuan dan perhitungan menggunakan Algoritma A*. Rumus untuk mengitung jarak Euclidean antara dua titik (x_1, y_1) dan (x_2, y_2) dalam graf berikut ini.

$$h(n) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{3.1}$$

A	(-15, 5)	D	(-3, 5)	G	(-7, -3)
B	(-10, 4)	E	(-15, 0)	H	(-4, -4)
C	(-7, 5)	F	(-12, -4)	I	(0, 0)

Tabel 3.4. Koordinat Titik dalam Graf

1. Dalam analisis menggunakan Algoritma A*, akan dihitung graf dengan titik awalnya A dan titik tujuannya I. Perhitungan akan dijabarkan dan ditunjukkan di bawah ini.

   (a) Inisialisasi

- Titik awal : A
- Titik tujuan : I
- $g(A)$ = 0
- $h(A) = \sqrt{(0-(-15))^2 + (0-(-5))^2} \approx 15,8$
- $f(A) = g(A) + h(A) = 0 + 15,8 = 15,8$

(b) Iterasi 1 (Evaluasi Titik A)

Titik A bertetangga dengan titik B dan titik E sehingga dimasukkan ke dalam *open list*.

i. Titik B

- $g(B) = g(A) + 22 = 0 + 22 = 22$
- $h(B) = \sqrt{(0-(-10))^2 + (0-(-4))^2} \approx 10,7$
- $f(B) = g(B) + h(B)$ = 22 + 10,7 = 32,7

ii. Titik E

- $g(E) = g(A) + 9 = 0 + 9 = 9$
- $h(E) = \sqrt{(0-(-15))^2 + (0-0)^2} \approx 15$
- $f(E) = g(E) + h(E) = 9 + 15 = 24$

Titik E dimasukan kedalam *closed list*.

(c) Iterasi 2 (Evaluasi titik B)

Titik B bertetangga dengan titik $A, C,$ dan titik F sehingga dimasukkan ke dalam *open list*.

i. Titik C

- $g(C) = g(B) + 12 = 22 + 12 = 34$
- $h(C) = \sqrt{(0-(-7))^2 + (0-5)^2} \approx 8,6$
- $f(C) = g(C) + h(C) = 34 + 8,6 = 42,6$

ii. Titik F

- $g(F) = g(B) + 18 = 22 + 18 = 40$

- $h(F) = \sqrt{(0-(-12))^2 + (0-(-4))^2} \approx 12,6$
- $f(F) = g(F) + h(F) = 40 + 12,6 = 52,6$

(d) Iterasi 3 (Evaluasi titik E)

Titik E bertetangga dengan titik A dan titik F sehingga dimasukkan ke dalam *open list*.

i. Titik F

- $g(F) = g(E) + 11 = 9 + 11 = 20$
- $h(F) = \sqrt{(0-(-12))^2 + (0-(-4))^2} \approx 12,6$
- $f(F) = g(F) + h(F) = 20 + 12,6 = 32,6$

Algoritma A* memilih titik dengan nilai $f(n)$ terendah dari titik yang ada dalam *open list*, sehingga selanjutnya titik B akan dievaluasi sebelum titik G. Namun, dalam perhitungan titik yang bertetangga, titik B mengevaluasi titik C dan F yang memiliki nilai $f(n)$ lebih tinggi dibandingkan dengan evaluasi tetangga titik E. Sehingga akhirnya titik E dan F akan tetap terpilih ke dalam *closed list*.

(e) Iterasi 4 (Evaluasi titik F)

Titik F bertetangga dengan titik B, E dan titik G sehingga dimasukkan ke dalam *open list*.

i. Titik G

- $g(G) = g(F) + 30 = 20 + 30 = 50$
- $h(G) = \sqrt{(0-(-7))^2 + (0-(-3))^2} \approx 7,6$
- $f(G) = g(G) + h(G) = 50 + 7,6 = 57,6$

(f) Iterasi 5 (Evaluasi titik G)

Titik G bertetangga dengan titik C, F dan titik H sehingga dimasukkan ke dalam *open list*.

i. Titik C

- $g(C) = g(G) + 13 = 50 + 13 = 63$
- $h(C) = \sqrt{(0-(-7))^2 + (0-5)^2} \approx 8,6$
- $f(C) = g(C) + h(C) = 63 + 8,6 = 71,6$

ii. Titik H

- $g(H) = g(G) + 9 = 50 + 9 = 59$
- $h(H) = \sqrt{(0-(-4))^2 + (0-(-4))^2} \approx 5,6$
- $f(H) = g(H) + h(H) = 59 + 5,6 = 64,6$

Titik H dipilih untuk dimasukkan ke dalam *closed list*.

(g) Iterasi 6 (Evaluasi titik H)

Titik H bertetangga dengan titik G dan I sehingga dimasukkan ke dalam *open list*.

i. Titik I

- $g(I) = g(H) + 17 = 59 + 17 = 76$
- $h(I) = \sqrt{(0-0)^2 + (0-0)^2} = 0$
- $f(I) + g(I) + h(I) = 76 + 0 = 76$

Iterasi telah selesai dilakukan karena telah sampai di titik tujuan I dan iterasi selesai pada iterasi ke-6.

Tabel 3.5. *Closed List* Titik $A \rightarrow I$

Via	**Tujuan**	***f(n) = g(n) + h(n)***	**Jarak**
A	A	15,8	0
A	E	24	9
E	F	32,6	11
F	G	57,6	30
G	H	64,6	9
H	I	76	17

Pada Tabel 3.5, perhitungan rute yang dilalui dari titik A menuju titik tujuan I. Algoritma A* menelusuri suatu rute dengan menghitung jarak terpendek yang jaraknya garis lurus ke titik I sehingga rute yang dilalui dari titik awal menuju titik tujuan adalah $A \rightarrow E \rightarrow F \rightarrow G \rightarrow H \rightarrow I$, dengan total jarak ialah 76.

2. Dalam analisis menggunakan Algoritma A*, akan dihitung graf dengan titik awalnya B dan titik tujuannya I. Perhitungan akan dijabarkan dan ditunjukkan di bawah ini.

   (a) Inisialisasi

   - Titik awal : B
   - Titik tujuan : I
   - $g(B) = 0$
   - $h(B) = \sqrt{(0-(-10))^2 + (0-4)^2} \approx 10,7$
   - $f(E) = g(E) + h(E) = 0 + 10,7 = 10,7$

   (b) Iterasi 1 (Evaluasi Titik B)

   Titik B bertetangga dengan titik A, C dan titik F.

   i. Titik A

   - $g(A) = g(B) + 22 = 0 + 22 = 22$
   - $h(A) = \sqrt{(0-(-15)^2 + (0-5)^2} \approx 15,8$
   - $f(A) = g(A) + h(A) = 22 + 15,8 = 37,8$

   ii. Titik C

   - $g(C) = g(B) + 12 = 0 + 12 = 12$
   - $h(C) = \sqrt{(0-(-7)^2 + (0-5)^2} \approx 8,6$
   - $f(C) = g(C) + h(C) = 12 + 8,6 = 20,6$

iii. Titik F

- $g(F) = g(B) + 18 = 0 + 18 = 18$
- $h(F) = \sqrt{(0-(-12)^2 + (0-(-4))^2} \approx 12,6$
- $f(F) = g(F) + h(F) = 18 + 12,6 = 30,6$

Ditunjukkan titik C memiliki nilai $f(n)$ terendah sehingga titik C akan dipilih dan dimasukkan ke dalam *closed list*.

(c) Iterasi 2 (Evaluasi titik A)

Titik A bertetangga dengan titik B dan titik E sehingga dapat dimasukkan ke dalam *open list*.

i. Titik E

- $g(E) = g(A) + 9 = 22 + 9 = 31$
- $h(E) = \sqrt{(0-(-15)^2 + (0-0)^2} \approx 15$
- $f(D) = g(D) + h(D) = 31 + 15 = 46$

(d) Iterasi 3 (Evaluasi titik C)

Titik C bertetangga dengan titik B, D, dan titik G.

i. Titik D

- $g(D) = g(C) + 20 = 12 + 20 = 32$
- $h(D) = \sqrt{(0-(-3)^2 + (0-5)^2} \approx 5,8$
- $f(D) = g(D) + h(D) = 32 + 5,8 = 37,8$

ii. Titik G

- $g(G) = g(C) + 13 = 12 + 13 = 25$
- $h(A) = \sqrt{(0-(-7)^2 + (0-(-3))^2} \approx 7,6$
- $f(A) = g(A) + h(A) = 25 + 7,6 = 32,6$

(e) Iterasi 4 (Evaluasi titik F)

Titik F bertetangga dengan titik B, E, dan titik G.

i. Titik E

- $g(E) = g(F) + 11 = 18 + 11 = 29$
- $h(E) = \sqrt{(0-(-15)^2 + (0-0)^2} \approx 15$
- $f(E) = g(E) + h(E) = 29 + 15 = 44$

ii. Titik G

- $g(G) = g(F) + 30 = 18 + 30 = 48$
- $h(G) = \sqrt{(0-(-7)^2 + (0-(-3))^2} \approx 7,6$
- $f(G) = g(G) + h(G) = 48 + 7,6 = 55,6$

Ditunjukkan pada iterasi ke 3 dimana nilai $f(n)$ dari titik C dan tetangganya memiliki nilai $f(n)$ yang lebih rendah dari pada melewati titik A dan F sehingga titik C dan tetangganya yaitu titik G akan dipilih selanjutnya dan dimasukkan ke dalam *closed list*.

(f) Iterasi 5 (Evaluasi titik G)

Titik G bertetangga dengan titik C, F dan titik H

i. Titik F

- $g(F) = g(G) + 30 = 25 + 30 = 55$
- $h(F) = \sqrt{(0-(-12)^2 + (0-(-4))^2} \approx 12,6$
- $f(F) = g(F) + h(F) = 55 + 12,6 = 67,6$

ii. Titik H

- $g(H) = g(G) + 9 = 25 + 9 = 34$
- $h(H) = \sqrt{(0-(-4)^2 + (0-(-4))^2} \approx 5,6$
- $f(H) = g(H) + h(H) = 34 + 5,6 = 39,6$

Ditunjukkan bahwa titik H memiliki nilai $f(n)$ lebih kecil sehingga akan dipilih selanjutnya dan dimasukkan ke dalam *closed list*.

(g) Iterasi 6 (Evaluasi titik H)

Titik H bertetangga dengan titik G dan titik I.

i. Titik I

- $g(I) = g(H) + 17 = 34 + 17 = 51$
- $h(I) = \sqrt{(0-0)^2 + (0-0)^2} \approx 0$
- $f(I) = g(I) + h(I) = 51 + 0 = 51$

Perhitungan telah sampai di iterasi dimana hasilnya mencapai titik tujuan I, maka iterasi dianggap telah selesai.

Tabel 3.6. *Closed List* Titik $A \rightarrow I$

Via	**Tujuan**	***f(n) = g(n) + h(n)***	**Jarak**
B	B	10,7	0
B	C	20,6	12
C	G	32,6	13
G	H	39,6	9
H	I	51	17

Pada Tabel 3.6, perhitungan rute yang dilalui dari titik B menuju titik tujuan I. Algoritma A* menelusuri suatu rute dengan menghitung jarak terpendek yang jaraknya garis lurus ke titik I sehingga rute yang dilalui dari titik awal menuju titik tujuan adalah $B \rightarrow C \rightarrow G \rightarrow H \rightarrow I$, dengan total jarak ialah 51.

3. Dalam analisis menggunakan Algoritma A*, akan dihitung graf dengan titik awalnya E dan titik tujuannya I. Perhitungan akan dijabarkan dan ditunjukkan di bawah ini.

(a) Inisialisasi

- Titik awal : E

- Titik tujuan : I
- $g(E)$ = 0
- $h(E) = \sqrt{(0-0)^2 + (0-(-15))^2} \approx 15$
- $f(E) = g(E) + h(E) = 0 + 15 = 15$

(b) Iterasi 1 (Evaluasi Titik E)

Titik E bertetangga dengan titik A dan titik F sehingga dimasukkan ke dalam *open list*.

i. Titik A

- $g(A) = g(E) + 9 = 0 + 9 = 9$
- $h(A) = \sqrt{(0-(-15)^2 + (0-5)^2} \approx 15,8$
- $f(A) = g(A) + h(A) = 9 + 15,8 = 24,8$

ii. Titik F

- $g(F) = g(E) + 11 = 0 + 11 = 11$
- $h(F) = \sqrt{(0-(-12)^2 + (0-(-4)^2} \approx 12,6$
- $f(F) = g(F) + h(F) = 11 + 12,6 = 23,65$

Ditunjukkan bahwa titik F memiliki nilai $f(n)$ lebih kecil sehingga akan dipilih selanjutnya dan dimasukkan ke dalam *closed list*.

(c) Iterasi 2 (Evaluasi titik F)

Titik F bertetangga dengan titik B, E dan titik G yang akan dimasukkan ke dalam *open list*.

i. Titik B

- $g(B) = g(F) + 18 = 11 + 18 = 29$
- $h(B) = \sqrt{(0-(-10)^2 + (0-4)^2} \approx 10,7$
- $f(B) = g(B) + h(B) = 29 + 10,7 = 39,7$

ii. Titik G

- $g(G) = g(F) + 30 = 11 + 30 = 41$

- $h(G) = \sqrt{(0-(-7))^2 + (0-(-3))^2} \approx 7,6$
- $f(G) = g(G) + h(G) = 41 + 7,6 = 48,6$

Ditunjukkan bahwa titik B memiliki nilai $f(n)$ lebih kecil sehingga akan tetapi belum tentu nilai B terpilih dan di masukkan ke dalam *closed list* karena akan dihitung di iterasi selanjutnya bahwa tetangga dari titik B lebih kecil dari pada titik G.

(d) Iterasi 3 (Evaluasi titik B)

Titik B bertetangga dengan titik $A, C,$ dan titik F sehingga dimasukkan ke dalam *open list*.

i. Titik C

- $g(C) = g(B) + 12 = 29 + 12 = 41$
- $h(C) = \sqrt{(0-(-7))^2 + (0-5)^2} \approx 8,6$
- $f(C) = g(C) + h(C) = 41 + 8,6 = 49,6$

(e) Iterasi 4 (Evaluasi titik C)

Titik C bertetangga dengan titik $A, D,$ dan titik G.

i. Titik D

- $g(D) = g(C) + 20 = 41 + 20 = 61$
- $h(D) = \sqrt{(0-(-3))^2 + (0-5)^2} \approx 5,8$
- $f(D) = g(D) + h(D) = 61 + 5,8 = 66,6$

(f) Iterasi 5 (Evaluasi titik G)

Titik G bertetangga dengan titik C, F dan titik H sehingga dimasukkan ke dalam *open list*.

i. Titik C

- $g(C) = g(G) + 13 = 41 + 13 = 54$
- $h(C) = \sqrt{(0-(-7))^2 + (0-5)^2} \approx 7,6$
- $f(C) = g(C) + h(C) = 54 + 7,6 = 71,6$

ii. Titik H

- $g(H) = g(G) + 9 = 41 + 9 = 50$
- $h(H) = \sqrt{(0-(-4))^2 + (0-(-4))^2} \approx 5,6$
- $f(H) = g(H) + h(H) = 50 + 5,6 = 55,6$

Ditunjukkan bahwa titik H melalui titik G memiliki nilai $f(n)$ lebih kecil dibandingkan melalui titik B sehingga akan dipilih selanjutnya dan dimasukkan ke dalam *closed list* untuk titik G dan titik H.

(g) Iterasi 6 (Evaluasi titik H)
Titik H bertetangga dengan titik G dan I sehingga dapat dimasukkan ke dalam *open list*.

i. Titik I

- $g(I) = g(H) + 17 = 50 + 17 = 67$
- $h(I) = \sqrt{(0-0)^2 + (0-0)^2} \approx 0$
- $f(I) = g(I) + h(I) = 67 + 0 = 67$

Ditunjukkan bahwa iterasi telah sampai di titik tujuan I, maka iterasi telah selesai dilakukan.

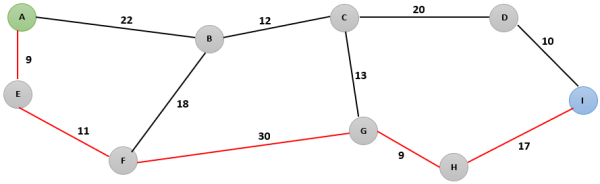
Tabel 3.7. *Closed List* Titik $E \rightarrow I$

Via	**Tujuan**	***f(n) = g(n) + h(n)***	**Jarak**
E	E	15	0
E	F	23,6	11
F	G	48,6	30
G	H	55,6	9
H	I	67	17

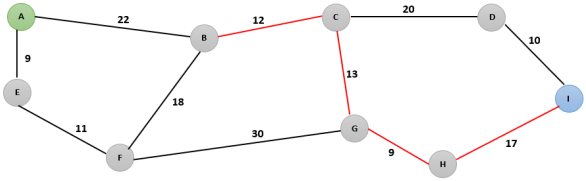
Pada Tabel 3.7, perhitungan rute yang dilalui dari titik E menuju titik tujuan I. Algoritma A* menelusuri suatu rute dengan menghitung jarak terpendek yang jaraknya garis

lurus ke titik L sehingga rute yang dilalui adalah titik $E \rightarrow F \rightarrow G \rightarrow H \rightarrow I$, dengan total jarak ialah 67.

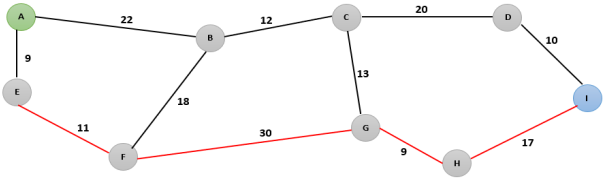
Berikut ini akan ditunjukan hasil rute terpendek yang terpilih menggunakan Algoritma A* dari titik awal A dan E menuju titik tujuan I.



Gambar 3.11. Rute Terpilih Titik $A \rightarrow I$



Gambar 3.12. Rute Terpilih Titik $B \rightarrow I$



Gambar 3.13. Rute Terpilih Titik $E \rightarrow I$

3.4 Perbandingan Algoritma

Dalam tahapan analisis, telah diuji masing-masing algoritma dengan tiga kali percobaan dengan tiga titik awal berbeda dan titik tujuan yang sama. Selanjutnya akan dilakukan perbandingan untuk menyelesaikan masalah rute terpendek dalam hasil total jarak, banyaknya titik yang dilewati, dan waktu berjalannya algoritma dalam tiga titik pengujian yang berbeda.

Tabel 3.8. Perbandingan Kinerja Algoritma pada Graf untuk Data Hasil Jarak

No.	**Titik Awal $\rightarrow$ Titik Tujuan**	**Jarak**		
		Dijkstra	**Floyd-Warshall**	**A***
1.	$A \rightarrow I$	64	64	76
2.	$B \rightarrow I$	42	42	51
3	$E \rightarrow I$	67	67	67

Pada Tabel 3.8, menunjukan data untuk membandingkan masing-masing algoritma dalam menyelesaikan permasalahan rute terpendek dalam hal total jarak untuk tiga kali pengujian dengan titik awal yang berbeda dan titik tujuan yang sama. Pada tabel tersebut menunjukkan bahwa masing-masing algoritma menghasilkan total jarak terpendek menggunakan Algoritma Dijkstra dan Algoritma Floyd-Warshall memiliki total jarak yang sama sedangkan Algoritma A* memiliki total jarak yang berbeda serta lebih besar dibandingkan dua algoritma lain yang digunakan pada penelitian ini. Akan tetapi pada percobaan dengan titik awal E dan titik tujuan I, hasil ketiga algoritma sama.

Dalam tabel terlihat bahwa algoritma A* pada titik awal A menuju titik tujuan I memiliki jumlah jarak ialah 76, memiliki selisih 12 jarak lebih besar dibandingkan dua algoritma lainnya.

Begitupun untuk titik awal B menuju titik tujuan I yang memiliki jumlah jarak ialah 51, dimana selisih 9 jarak lebih besar dibandingkan dua algoritma yang lainnya. Hal tersebut dikarenakan Algoritma A* merupakan sebuah fungsi yang membutuhkan bantuan heuristik atau estimasi jarak n menuju titik tujuan, dibandingkan dua algoritma lain yang tidak membutuhkan bantuan lain. Kebergantungan Algoritma A* pada heuristik dalam memperkirakan jarak ke titik n ke titik tujuan bisa membuat hasil algoritma ini menemukan rute yang tidak optimal.

Tabel 3.9. Perbandingan Kinerja Algoritma pada Graf untuk Jumlah Titik

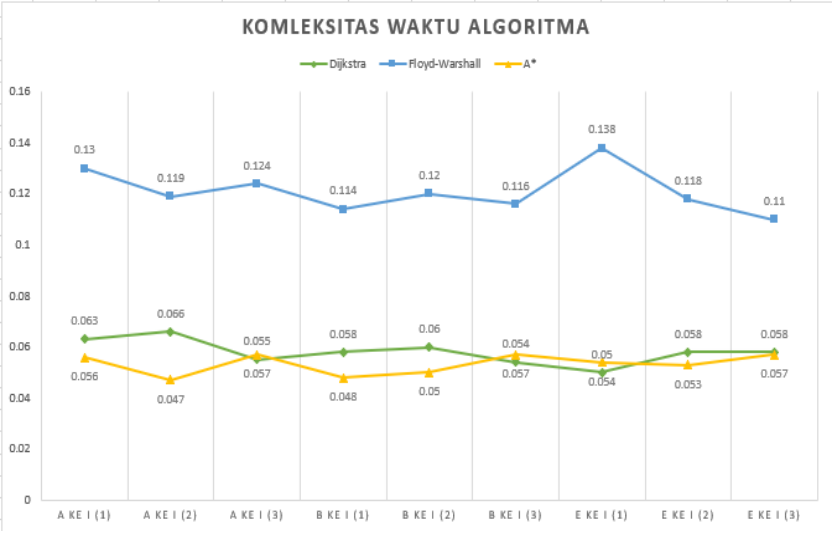
No.	**Titik Awal → Titik Tujuan**	**Jumlah Titik**		
		Dijkstra	**Floyd-Warshall**	**A***
1.	$A \rightarrow I$	5	5	6
2.	$B \rightarrow I$	4	4	5
3.	$E \rightarrow I$	5	5	5

Pada Tabel 3.9, menunjukan data untuk membandingkan masing-masing algoritma dalam menyelesaikan permasalahan rute terpendek dalam kasus banyaknya jumlah titik dalam rute terpendek yang ditemukan untuk tiga kali pengujian dengan titik awal yang berbeda dan titik tujuan yang sama. Pada tabel tersebut menunjukkan bahwa masing-masing algoritma menghasilkan jumlah titik yang hampir sama. Dalam perbandingan jumlah titik, Algortima Dijkstra dan Floyd-Warshall lebih unggul dibandingkan Algoritma A* dalam percobaan titik $A \rightarrow I$ dan $B \rightarrow I$. Akan tetapi, dalam percobaan titik $E \rightarrow I$ menghasilkan banyak titik yang sama untuk ketiga algoritma tersebut.

Dalam tabel terlihat bahwa Algoritma A* pada titik awal A

dan B menuju titik tujuan I, lebih banyak titik dibandingkan Algoritma lainnya yang hanya lima buah titik. Titik yang dipertimbangkan oleh Algoritma A* lebih banyak karena algoritma ini mengeksplorasi jalur berdasarkan fungsi heuristiknya atau jarak garis lurus ke titik tujuan, yang mengarah pada eksplorasi lebih banyak titik dari pada kedua algoritma lainnya yang hanya mempertimbangkan jarak sebenarnya dari titik awal. Meskipun algoritma ini lebih efisien dalam menemukan jalur optimal, penggunaan heuristik juga dapat memunculkan lebih banyak titik yang dieksplorasi dalam proses pencarian rute terpendek.

Berikut ini akan ditunjukkan kompleksitas waktu yang dibutuhkan masing-masing algoritma dalam 3 (tiga) kali memulai program pada setiap titik awal A, B dan E.



Gambar 3.14. Kompleksitas Waktu Algoritma

Setelah dilakukan perhitungan kompleksitas waktu sebanyak 3 (tiga) kali dalam tiap kasus titik awal yang berbeda. Hasil perhitungan rata-rata kompleksitas waktu dari tiap algoritma akan ditunjukkan berikut ini.

Tabel 3.10. Perbandingan Kinerja Algoritma pada Graf untuk Waktu Berjalan

No.	**Titik Awal $\rightarrow$ Titik Tujuan**	**Waktu**		
		Dijkstra	**Floyd-Warshall**	**A***
1.	$A \rightarrow I$	0,061 s	0,124 s	0,053 s
2.	$B \rightarrow I$	0,057 s	0,116 s	0,051 s
3.	$E \rightarrow I$	0,055 s	0,112 s	0,054 s

Pada Tabel 3.10, menunjukan data untuk membandingkan masing-masing algoritma dalam menyelesaikan permasalahan rute terpendek dalam kasus waktu komputasi dalam rute terpendek yang ditemukan untuk tiga kali pengujian dengan titik tujuan yang berbeda dan titik tujuan yang sama. Pada tabel tersebut menunjukkan bahwa masing-masing algoritma menghasilkan waktu yang berbeda-beda. Dalam perbandingan penggunaan waktu, Algortima A* dan Dijkstra lebih unggul dalam kecepatan rata-rata penggunaan waktu komputasi dibandingkan Algoritma Floyd-Warshall.

Dalam kasus ini, Algoritma Floyd-Warshall lebih lambat dibandingkan dua algoritma lain. Hal ini disebabkan karena algoritma ini mengharuskan memeriksa seluruh kemungkinan pasangan titik dan mencoba memperbarui jalur terpendek di antara mereka, bahkan jika graf tidak terhubung secara langsung. Walaupun begitu, keunggulan algoritma ini dapat menangani graf dengan berbobot negatif, serta kemampuan untuk mencari

rute terpendek antara seluruh pasangan titik dalam satu kali penggunaan algoritma. Dalam kasus ini, Algoritma Floyd-Warshall lebih butuh banyak waktu tetapi algoritma dapat menemukan rute terpendek dalam pengujian titik awal A dan B dibandingkan Algoritma A* karena tidak dibutuhkannya bantuan lain, seperti heurisik.

Algoritma A* lebih cepat daripada Algoritma Dijkstra dan Floyd-Warshall dalam pencarian rute terpendek dari satu titik ke titik lainnya dalam kasus graf ini. Namun, meskipun Algoritma A* cenderung lebih cepat, tidak ada jaminan bahwa algoritma ini dapat menemukan rute terpendek. Hal ini bergantung dengan fungsi heuristik yang digunakan dan apakah jalur yang diekspolasi oleh algoritma tersebut mencapai tujuan secara optimal. Algoritma ini mengunakan heuristik untuk memilih rute yang memiliki potensi untuk menjadi rute terpendek. Namun, jika heuristik yang digunakan tidak akurat, maka algoritma ini mungkin tidak menemukan rute terpendek. Sebaliknya, Algoritma Dijkstra dan Floyd-Warshall secara tegas dan jelas mencari rute terpendek antara setiap titik dalam graf, sehingga menjamin menemukan rute terpendek, tetapi dengan kompleksitas waktu yang lebih tinggi. Jadi, meskipun Algoritma A* lebih cepat terutama dalam graf besar, ada kemungkinan bahwa algoritma ini tidak menemukan rute terpendek.

3.5 Penerapan Algoritma Di Kehidupan Sehari-hari

Perencanaan rute sering kali menjadi hal penting dalam kehidupan sehari-hari. Baik itu perjalanan sehari-hari, transportasi kargo, atau bahkan perjalanan antarplanet untuk

eksplorasi ruang angkasa, menemukan rute terpendek atau solusi optimal seringkali menjadi sebuah tantangan. Algoritma Dijkstra, Floyd-Warshall, dan A* adalah alat penting yang mengubah cara penyelesaian masalah tersebut.

Algoritma Dijkstra dikenal luas dalam menemukan rute terpendek dalam jaringan lalu lintas, yang membantu kita menghindari kemacetan lalu lintas atau menemukan rute yang efisien di kota-kota yang padat. Di sisi lain, Algoritma Floyd-Warshall berguna untuk menemukan jalur terpendek antara semua pasangan titik dalam sebuah graf, yang memberikan pemahaman lebih dalam tentang hubungan antar titik. Sementara itu, A* adalah algoritma pencarian jalan yang terkenal digunakan dalam aplikasi mulai dari video game hingga robotika. Dengan menggunakan teknik heuristik, A* menggabungkan kecepatan pencarian algoritma pencarian berbasis graf dengan kemampuan untuk membuat keputusan cerdas tentang jalur yang paling mungkin, sehingga memungkinkan kita bernavigasi secara efisien dan efektif.

Penerapan algoritma tidak hanya memengaruhi teknologi dan industri, namun juga kehidupan kita sehari-hari, membantu kita mengambil keputusan yang lebih baik dan meningkatkan efisiensi dalam banyak konteks, mulai dari perjalanan sehari-hari hingga perencanaan proyek besar. Berikut ini adalah beberapa kasus di kehidupan nyata yang efektif digunakan pada masing-masing algoritma.

1. Algoritma Dijkstra

    (a) Jaringan Telekomunikasi

    (b) Penjelasan: Di industri telekomunikasi, algoritma

Dijkstra digunakan untuk merencanakan dan mengoptimalkan rute jaringan, misalnya, dalam jaringan telepon, jaringan internet, atau jaringan komunikasi lainnya. Ini membantu operator jaringan untuk mengelola lalu lintas dengan lebih efisien dan meminimalkan biaya.

2. Algoritma Floyd-Warshall

   (a) Perencanaan Rute Pengiriman Gas Bumi di Jaringan Pipa

   (b) Penjelasan: Dalam hal ini, setiap titik akan mewakili stasiun pengiriman gas atau titik jaringan pipa, dan sisi akan mewakili jalur pengiriman gas antara titik-titik tersebut. Algoritma ini akan membantu dalam menentukan jalur pengiriman gas terpendek antara semua pasangan stasiun pengiriman, memungkinkan untuk merencanakan pengiriman dengan efisien

3. Algoritma A*

   (a) Sistem Pengiriman Paket Pintar Berbasis Drone

   (b) Penjelasan: Algoritma ini diterapkan dalam sistem pengiriman paket pintar berbasis drone dimana drone harus menavigasi dari titik awal ke tujuan dengan efisien dan cepat. Dalam kasus ini, informasi heuristik seperti jarak dan rintangan di udara digunakan untuk membantu drone merencanakan rute yang optimal. Dengan menggunakan algoritma ini, drone dapat menemukan jalur terpendek ke tujuan dengan memperhitungkan informasi heuristik dan

meminimalkan waktu tempuh serta menghindari rute yang mengalami hambatan.

BAB IV
PENUTUP

4.1 Kesimpulan

Dalam teori graf, terdapat banyak algoritma yang dapat digunakan dalam pencarian rute terpendek. Penelitian ini menggunakan tiga buah algoritma yang memiliki perbedaan dalam proses pencarian rute terpendek. Algoritma yang digunakan dalam penelitian ini ialah Algoritma Dijkstra, Floyd-Warshall, dan A*. Penelitian ini menggunakan tiga buah algoritma untuk menuji kinerja ketiga algoritma. Perbandingan pengujian dilakukan dengan pengujian dalam konsteks total jarak, banyaknya jumlah titik dan waktu komputasi untuk menjalankan program algoritma.

Tiga buah algoritma tersebut berbeda dalam pendekatan dan kinerja mereka dalam mencari rute terpendek di graf. Algoritma Dijkstra biasanya digunakan untuk mencari rute terpendek dari satu titik ke seluruh titik lainnya dalam graf dengan bobot non-negatif. Meskipun algoritma ini menjamin rute terpendek, namun kinerjanya akan menurun secara signifikan saat jumlah titik dalam sebuah graf meningkat atau terlalu banyak. Sementara itu, Algoritma Floyd-Warshall lebih cocok digunakan untuk mencari rute terpendek untuk seluruh pasangan titik dalam graf berbobot negatif ataupun non-negatif. Meskipun algoritma ini juga menjamin menemukan rute terpendek, tetapi membutuhkan waktu komputasi yang lebih besar terutama pada graf yang besar. Hal tersebut dikarenakan Algoritma Floyd-Warshall menggunakan matriks untuk menyimpan informasi rute terpendek antar seluruh pasangan titik. Disisi lain, Algoritma A* menggunakan heuristik

untuk melakukan pencarian sehingga dapat lebih efisien. Namun, kinerja Algoritma A* sangat bergantung pada kualitas heuristik yang digunakan. Apabila heuristik yang digunakan buruk, Algoritma mungkin menghasilkan solusi yang tidak optimal. Selain itu, Algoritma A* tidak efisien dalam beberapa kasus ketika graf memiliki struktur yang kompleks atau faktor cabang yang sangat banyak serta tidak dapat menangani secara langsung sebuah graf berbobot negatif.

Berdasarkan analisis perhitungan kinerja masing-masing algoritma dari tiga konteks, ditunjukkan bahwa Algoritma Dijkstra dan Floyd-Warshall lebih baik dibandingkan Algoritma A* dalam hal penemuan total jarak dan banyaknya titik. Untuk rata-rata penggunaan waktu komputasi Algoritma A* lebih unggul dibandingkan kedua Algoritma lain. Dalam kasus penelitian ini, Algoritma Dijkstra dan Floyd-warshall melakukan penelusuran menyeluruh untuk pencarian rute terpendek yang lebih efisien serta optimal dibandingkan Algoritma A* karena dalam dua konteks pengujian, yaitu dengan total jarak dan banyaknya titik, Algoritma Dijkstra dan Floyd-Warshall menghasilkan total jarak dan banyak titik lebih kecil dibandingkan Algoritma A*. Jadi, pemilihan algoritma yang akan digunakan tergantung pada kebutuhan spesifik dari masalah yang dihadapi. Jika akurasi adalah prioritas utama dan waktu bukan merupakan faktor kritis, maka Algoritma Dijkstra atau Floyd-Warshall lebih disarankan. Jika kecepatan adalah prioritas utama dan toleransi terhadap sedikit ketidakakuratan dapat diterima, maka Algoritma A* adalah pilihan yang lebih baik.

4.2 Kritik dan Saran

Beberapa algoritma mungkin tidak dapat menangani perubahan dinamis yang terjadi dalam struktur graf yang sangat besar atau kompleks, seperti penambahan atau penghapusan sisi atau titik. Hal ini dapat menjadi tantangan saat diaplikasikan dalam analisis jaringan sosial atau pemodelan jaringan transportasi kota. Dibutuhkannya pengembangan algoritma yang mampu beradaptasi dengan perubahan dinamis dalam struktur graf yang dapat meningkatkan fleksibilitas dan efektivitas solusi. Hal ini dapat mencakup pengembangan algoritma inkremental atau algoritma yang dapat mengubah strategi saat graf mengalami perubahan.

DAFTAR PUSTAKA

Abdussakir, Azizah, N.N., Nilna Novandika, F.F. 2009. *Teori Graf*. Malang: UIN Malang Press.

Andiyani, Sri., dan Endah Wulan Perwitasari. 2014. Penentuan Rute Terpendek Pengambilan Sampah di Kota Merauke Menggunakan Algoritma Dijkstra. SEMINAR NASIONAL TEKNOLOGI INFORMASI DAN KOMUNIKASI TERAPAN, 4(1).

Anusha, Aziz., Sheikh Tasfia, dan M Akhtaruzzan. 2022. A Comparative Analysis among Three Different Shortest Path-finding Algorithms. *Journal of Physics Conference Series*, 1087(2).

Ardyan, S., Mulyono, dan Suyitno, A. 2017. Implementasi Algoritma Dijkstra Dalam Pencarian Rute Terpendek Tempat Wisata Di Kabupaten. *UNNES Journal of Mathematics*, 6(2): 108-116.

Bondy, J. A., dan Murty U.S.R. 1982. *Graph Theory with Applications*. New York: Elsevier Science Publishing.

Buako, Zulmagfir., Lailany Yahya, dan Novianita Achmad. 2021. *Aplikasi Algoritma Floyd-Warshall Dengan Pendekatan Madm Dalam Menentukan Rute Terpendek Pengangkutan Sampah*. EULER, 9(2): 62-70.

Chartrand, Gary., dan Zhang, Ping. 2005 *Introduction to Graph Theory*. New York: McGraw-Hill Internasional Editions.

Diestel, R. 2000. *Graph Theory*. Edisi Kedua. Springer.

Diestel, R. 2018. *Graph Theory*. Edisi Kelima. Springer.

Harju, Tero. 2012. *Graph Theory*. Finland: Department of Mathematics University of Turku.

Hidayat, Alfian., Purnamasari, Ika., Siringoringo, Meiliyani. 2020. Penentuan Jalur Terpendek dengan Metode Heuristik Menggunakan Algoritma Sarang Semut (Ant Colony). EKSPONENSIAL, 11(1): 93-98.

Krisnamurti, Cyrenia N., dan Geong, Efrem Alfrando Pascal. 2021. Implementasi Algoritma Floyd-Warshall Untuk Menentukan Rute Terpendek Destinasi Wisata Lahuan Bajo. *UNNES Journal of Mathematics*, 10(1): 75-84.

Mansuri, Shabina B., dan Shiv Kumar. 2018. Comparative Analysis of Path Finding Algorithms. *IOSR Journal of Computer Engineering*, 20(5): 38-45.

Mardiyono, S. 1996. *Matematika Diskret*. Yogyakarta : FMIPA IKIP Yogyakarta.

Meng, S. C. Y., Adnan, N., Sukri, S. S., dan Zainon, W. M. N. W. 2023. An experiment on the performance of shortest path algorithm. *Journal of Network and Systems Management*, 31(2): 123-140.

Mukti, M Ridwan., dan Mulyono. 2018. Menentukan Rute Terpendek Dengan Menggunakan Algoritma Floyd-Warshall Dalam Pendistribusian Barang Pada Pt. Rapy Ray Putratama. KARISMATIKA, 4(1): 39-53.

Munir, Rinaldi. 2005; 2007. *Matematika Diskrit*. Edisi Ketiga. Bandung: Informatika.

Munir, Rinaldi. 2016. *Matematika Diskrit*. Edisi Revisi Keenam. Bandung: Informatika.

Munir, Rinaldi. 2008. *Struktur Diskrit*. Bandung: Penerbit Institut Teknologi Bandung.

Mutakhiroh, Indrato, dan Taufiq Hidayati. 2007. Pencarian Jalur Terpendek Menggunakan Algoritma Semut. Seminar Nasional Aplikasi Teknologi Informasi 2007. Yogyakarta 16 Juni 2007.

Nawagusti, V. A. 2018. Penerapan Algoritma Floyd Warshall Dalam Aplikasi Penentuan Rute Terpendek Mencari Lokasi BTS (Base Towe Station) Pada PT. GCI Palembang. Jurnal Nasional Teknologi dan Sistem Informasi, 4(2): 81-88.

Ridwan, Fatrmawati, dan Ririn D. 2020. Penggunaan Algoritma Floyd-Warshall untuk Menentukan Rute Menuju Air Terjun Waimarang. LAPLACE, 3(2):87-94.

Saputrama, Rendi., dan Hartatiana. 2021. Aplikasi Algoritma Dijkstra Untuk Menentukan Rute Terpendek Dari Kampus A ke B UIN Raden Fatah. E-Jurnal Matematika Volume 10(3): 173-178.

Siang, Jong Jek. 2002. *Matematika Diskrit dan Aplikasinya*. Yogyakarta: Andi.

Siang, Jong Jek. 2006. *Matematika Diskrit dan Aplikasinya pada Ilmu Komputer*. Yogyakarta: Andi.

Siang, Jong Jek. 2011. *Riset Operasi Dalam Pendekatan Algoritmis*. Edisi Pertama. Yogyakarta: Andi.

Sidabutar, Juni A., dan Ujian Sinulinga. (2022). The Implementation of Dijkstra Algorithm in Searching the Shortest Path for Fire Engines in Medan City. *Journal of Mathematics Technology and Education*, 1(2): 160-172.

Sidhu, Harseerat S., dan Gopal Krishan. 2022. Research On Dijkstra's, A*, Bellman-Ford And Floyd-Warshall Path Finding Algorithms. *International Research Journal of Modernization in Engineering Technology and Science*, 4(1): 886-893.

Sitorus, Lamhot. 2015. *Algoritma dan Pemrograman*. Yogyakarta: Penerbit Andi.

Suarga. 2006. *Algoritma dan Pemrograman*. Yogyakarta: Andi.

Sunardi S., Anton Yudhana, dan Ahmad Azhar Kahim. 2019. Implementasi Algoritma Dijkstra dan Algoritma Semut Untuk Analisis Rute Transjogja Berbasis Android. *IT Journal Research and Development*, 4(1): 1-9.

Suryani, E. (2010). Metode Heuristik untuk Optimasi: Teori dan Implementasi. Surabaya: ITS Press.

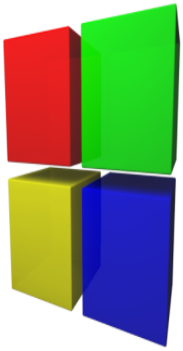
Umar, Rusdi., Anton Yudhana, dan Andi Prayudi. 2021. Analisis Perbandingan Algoritma Djikstra, A-Star, Dan Floyd Warshall Dalam Pencarian Rute Terdekat Pada Objek Wisata Kabupaten Dompu. Jurnal Teknologi Informasi dan Ilmu Komputer, 8(2): 227-234.

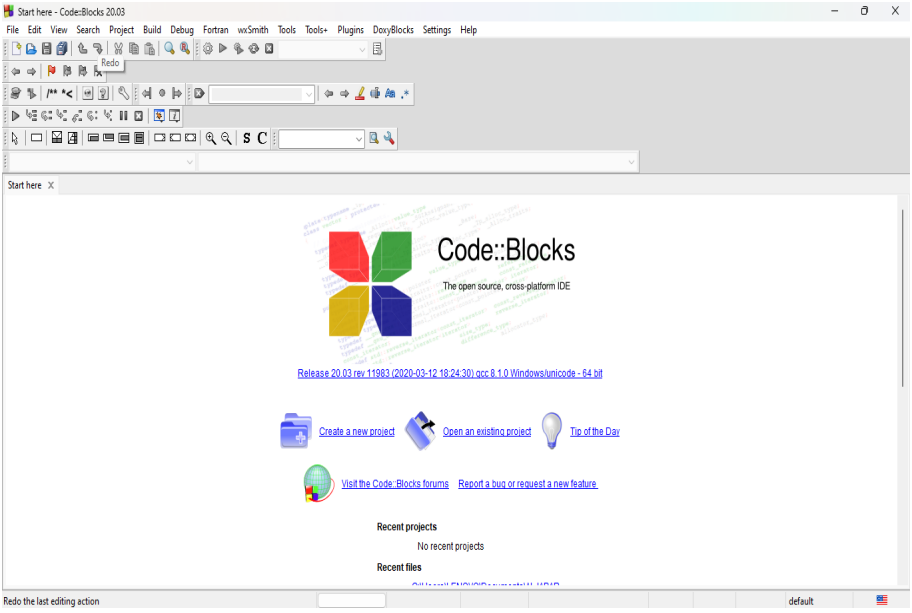
Vasudev, C. (2006). *Graph Theory with Applications*. New Delhi: New Age International Publisher.

West, Douglas B. (2001). *Introduction To Graph Theory*. Singapore: Pearson.

Wibowo, T. (2012). Optimasi Matematika: Teori dan Aplikasinya. Jakarta: Penerbit Universitas Terbuka.

Lampiran 1. Tampilan CodeBlocks





Lampiran 2. Kode Program C Algoritma Dijkstra

```c
#include <stdio.h>
 #include <stdlib.h>
 #define INF 999

 int main() {
     int n = 9; // Jumlah vertex

     // Matriks graf
     int G[12][12] = {
         {0, 22, 999, 999, 9, 999, 999, 999, 999},
         {22, 0, 12, 999, 999, 18, 999, 999, 999},
         {999, 12, 0, 20, 999, 999, 13, 999, 999},
         {999, 999, 20, 0, 999, 999, 999, 999, 10},
         {9, 999, 999, 999, 0, 11, 999, 999, 999},
         {999, 18, 999, 999, 11, 0, 30, 999, 999},
         {999, 999, 13, 999, 999, 30, 0, 9, 999},
         {999, 999, 999, 999, 999, 999, 9, 0, 17},
         {999, 999, 999, 10, 999, 999, 999, 17, 0}

     };

     int tempGraf[n][n], jarak[n], visit[n], temp[n], count;

     // Vertex asal
     int start = 0; // titik awal diganti

     // Inisialisasi matriks temporary dan variabel lainnya
     for (int i = 0; i < n; i++) {
         for (int j = 0; j < n; j++) {
             if (G[i][j] == 0) {
                 tempGraf[i][j] = INF;
             } else {
                 tempGraf[i][j] = G[i][j];
             }
         }
     }
     for (int i = 0; i < n; i++) {
         jarak[i] = tempGraf[start][i];
         temp[i] = start;
         visit[i] = 0;
```

```
41      }
42      jarak[start] = 0;
43      visit[start] = 1;
44
45      count = 1; // dimulai dari 1 karena kita tidak menghitung titik asal
46
47      // proses untuk menghitung vertex yang dikunjungi
48      int jarakmin, nextvertex;
49      while (count < n) {
50          jarakmin = INF;
51          for (int i = 0; i < n; i++) {
52              if (jarak[i] < jarakmin && visit[i] != 1) {
53                  jarakmin = jarak[i];
54                  nextvertex = i; // untuk memberikan titik pada jarak minimum
55              }
56          }
57          // untuk mengecek titik selanjutnya yang terhubung dengan
58          // titik lain yang memiliki jarak minimum
59          visit[nextvertex] = 1;
60          for (int i = 0; i < n; i++) {
61              if (visit[i] != 1) {
62                  if (jarakmin + tempGraf[nextvertex][i] < jarak[i]) {
63                      jarak[i] = jarakmin + tempGraf[nextvertex][i];
64                      temp[i] = nextvertex;
65                  }
66              }
67          }
68          count++;
69      }
70      // menampilkan jalur dan jarak untuk setiap titik
71      int a[n+1];
72      for (int i = 0; i < n; i++) {
73          if (i!= start) {
74              printf("\nJarak terpendek titik %d ke titik %d = %d\n", start, i, jarak[i]);
75              printf("Jalur terpendek: ");
76              int j = i;
77              printf("%d", i);
78              while (j!= start) {
79                  j = temp[j];
80                  printf(" <- %d", j);
```

```
81              }
82              printf("\n");
83          }
84      }
85      printf("\nTotal Jaraknya adalah %d\n", jarak[n-1]);
86      // Total jarak dari titik awal ke semua titik
87      return 0;
88  }
```

Lampiran 3. Kode Program C Algoritma Floyd-Warshall

```c
#include <stdio.h>
#include <limits.h>
#define V 9 // Jumlah Vertex dalam graf

//Fungsi untuk mencetak matriks
void printMatrix(int dist[][V]) {
  printf("Matriks jarak terpendek: %d \n");
  for(int i=0; i<V; i++) {
    for(int j=0; j<V; j++) {
      if(dist[i][j] == INT_MAX)
        printf("INF \t");
      else
        printf("%d \t", dist[i][j]);
    }
    printf("\n");
  }
}

// Fungsi untuk menyelesaikan masalah rute terpendek
void shortestPath(int graph[][V]) {
  int dist[V][V];

  //Menginisiasi matriks jarak awal
  for(int i=0; i<V; i++) {
    for(int j=0; j<V; j++) {
      dist[i][j] = graph[i][j];
    }
  }

  // Algoritma Floyd Warshall
  for(int k=0; k<V; k++) {
    for(int i=0; i<V; i++) {
      for(int j=0; j<V; j++) {
        if(dist[i][k] != INT_MAX && dist[k][j] != INT_MAX
           && dist[i][k] + dist[k][j] < dist[i][j]) {
            dist[i][j] = dist[i][k] + dist[k][j];
        }
      }
    }
    printMatrix(dist);
```

```
41    }
42    //printMatrix(dist);
43  }
44
45  int main() {
46    // Graf berbobot yang digunakan
47    int graph[V][V] = {
48          {0, 22, INT_MAX, INT_MAX, 9, INT_MAX, INT_MAX, INT_MAX, INT_MAX},
49          {22, 0, 12, INT_MAX, INT_MAX, 18, INT_MAX, INT_MAX, INT_MAX},
50          {INT_MAX, 12, 0, 20, INT_MAX, INT_MAX, 13, INT_MAX, INT_MAX},
51          {INT_MAX, INT_MAX, 20, 0, INT_MAX, INT_MAX, INT_MAX, INT_MAX, 10},
52          {9, INT_MAX, INT_MAX, INT_MAX, 0, 11, INT_MAX, INT_MAX, INT_MAX},
53          {INT_MAX, 18, INT_MAX, INT_MAX, 11, 0, 30, INT_MAX, INT_MAX},
54          {INT_MAX, INT_MAX, 13, INT_MAX, INT_MAX, 30, 0, 9, INT_MAX},
55          {INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, 9, 0, 17},
56          {INT_MAX, INT_MAX, INT_MAX, 10, INT_MAX, INT_MAX, INT_MAX, 17, 0}
57
58    };
59    shortestPath(graph);
60
61    return 0;
62  }
```

Lampiran 4. Kode Program C Algoritma A*

```c
#include <stdio.h>
#include <stdbool.h>
#include <limits.h>
#include <math.h>
#define NUM_NODES 9

// Struktur untuk merepresentasikan koordinat titik
typedef struct {
    int x, y;
} Point;

// Fungsi untuk menghitung jarak Euclidean antara dua titik
double euclideanDistance(Point p1, Point p2) {
    return sqrt(pow(p2.x - p1.x, 2) + pow(p2.y - p1.y, 2));
}

// Fungsi untuk menemukan indeks titik dengan nilai f(n) terendah yang belum dievaluasi
int minFValue(bool openSet[], double fValues[]) {
    double min = INT_MAX;
    int min_index;
    for (int i = 0; i < NUM_NODES; i++) {
        if (openSet[i] && fValues[i] < min) {
            min = fValues[i];
            min_index = i;
        }
    }
    return min_index;
}

// Fungsi untuk mencetak jalur yang ditemukan
void printPath(int cameFrom[], int current) {
    if (cameFrom[current] != -1) {
        printPath(cameFrom, cameFrom[current]);
        printf(" -> ");
    }
    printf("%d", current);
}

// Fungsi untuk menemukan jalur terpendek menggunakan algoritma A*
void AStar(int graph[NUM_NODES][NUM_NODES], Point coordinates[], int start, int goal) {
```

```
41      int gValues[NUM_NODES]; // Nilai g(n)
42      double hValues[NUM_NODES]; // Nilai heuristik (jarak Euclidean)
43      double fValues[NUM_NODES]; // Nilai f(n)
44      int cameFrom[NUM_NODES]; // Simpul sebelumnya dalam jalur terpendek
45      bool openSet[NUM_NODES]; // Menandai apakah titik telah dievaluasi
46
47      // Menginisialisasi nilai-nilai
48      for (int i = 0; i < NUM_NODES; i++) {
49          gValues[i] = INT_MAX;
50          fValues[i] = INT_MAX;
51          cameFrom[i] = -1;
52          openSet[i] = false;
53      }
54
55      // Menginisialisasi nilai untuk titik awal
56      gValues[start] = 0;
57      hValues[start] = 0; // Heuristik untuk titik awal bisa diatur menjadi 0 atau nilai yang sesuai
58      fValues[start] = gValues[start] + hValues[start]; // g(n) + h(n)
59      openSet[start] = true;
60
61      while (true) {
62          int current = minFValue(openSet, fValues); // Pilih titik dengan nilai f(n) terendah
63
64          if (current == goal)
65              break;
66
67          openSet[current] = false;
68
69          // Periksa tetangga-tetangga dari titik saat ini
70          for (int i = 0; i < NUM_NODES; i++) {
71              if (graph[current][i] != 0) {
72                  int tentative_g = gValues[current] + graph[current][i];
73
74                  if (tentative_g < gValues[i]) {
75                      cameFrom[i] = current;
76                      gValues[i] = tentative_g;
77                      hValues[i] = euclideanDistance(coordinates[i], coordinates[goal]); // Menggunakan jarak Euclidean
78                      fValues[i] = gValues[i] + hValues[i];
79                      if (!openSet[i])
80                          openSet[i] = true;
```

```
                }
            }
        }
    }

    // Mencetak jalur yang ditemukan
    printf("Path: ");
    printPath(cameFrom, goal);
}

int main() {
    // Representasi matriks graf
    int graph[NUM_NODES][NUM_NODES] = {
        {0, 22, 0, 0, 9, 0, 0, 0, 0},
        {22, 0, 12, 0, 0, 18, 0, 0, 0},
        {0, 12, 0, 20, 0, 0, 13, 0, 0},
        {0, 0, 20, 0, 0, 0, 0, 0, 10},
        {9, 0, 0, 0, 0, 11, 0, 0, 0},
        {0, 18, 0, 0, 11, 0, 30, 0, 0},
        {0, 0, 13, 0, 0, 30, 0, 9, 0},
        {0, 0, 0, 0, 0, 0, 9, 0, 17},
        {0, 0, 0, 10, 0, 0, 0, 17, 0}
    };

    // Koordinat titik
    Point coordinates[NUM_NODES] = {
        {-15, 5},  // Titik 0
        {-10, 4}, // Titik 1
        {-7, 5}, // Titik 2
        {-3, 5},// Titik 3
        {-15, 0},  // Titik 4
        {-12, -4},// Titik 5
        {-7, -3},// Titik 6
        {-4, -4},  // Titik 7
        {0, 0}   // Titik 8 (Tujuan)
    };

    int start = 0; // Simpul awal
    int goal = 8; // Simpul tujuan

    // Menemukan jalur terpendek dari titik awal ke titik tujuan menggunakan A*
    AStar(graph, coordinates, start, goal);

    return 0;
}
```

Lampiran 5. Hasil Output Menggunakan Program C

0.1 Hasil Algoritma Dijkstra

0.1.1 Titik A Ke Titik I

```
Jarak terpendek titik 0 ke titik 1 = 22
Jalur terpendek: 1 <- 0

Jarak terpendek titik 0 ke titik 2 = 34
Jalur terpendek: 2 <- 1 <- 0

Jarak terpendek titik 0 ke titik 3 = 54
Jalur terpendek: 3 <- 2 <- 1 <- 0

Jarak terpendek titik 0 ke titik 4 = 9
Jalur terpendek: 4 <- 0

Jarak terpendek titik 0 ke titik 5 = 20
Jalur terpendek: 5 <- 4 <- 0

Jarak terpendek titik 0 ke titik 6 = 47
Jalur terpendek: 6 <- 2 <- 1 <- 0

Jarak terpendek titik 0 ke titik 7 = 56
Jalur terpendek: 7 <- 6 <- 2 <- 1 <- 0

Jarak terpendek titik 0 ke titik 8 = 64
Jalur terpendek: 8 <- 3 <- 2 <- 1 <- 0

Total Jaraknya adalah 64
```

0.1.2 Titik B ke Titik I

```
Jarak terpendek titik 1 ke titik 0 = 22
Jalur terpendek: 0 <- 1

Jarak terpendek titik 1 ke titik 2 = 12
Jalur terpendek: 2 <- 1

Jarak terpendek titik 1 ke titik 3 = 32
Jalur terpendek: 3 <- 2 <- 1

Jarak terpendek titik 1 ke titik 4 = 29
Jalur terpendek: 4 <- 5 <- 1

Jarak terpendek titik 1 ke titik 5 = 18
Jalur terpendek: 5 <- 1

Jarak terpendek titik 1 ke titik 6 = 25
Jalur terpendek: 6 <- 2 <- 1

Jarak terpendek titik 1 ke titik 7 = 34
Jalur terpendek: 7 <- 6 <- 2 <- 1

Jarak terpendek titik 1 ke titik 8 = 42
Jalur terpendek: 8 <- 3 <- 2 <- 1

Total Jaraknya adalah 42
```

0.1.3 Titik E ke Titik I

```
Jarak terpendek titik 4 ke titik 0 = 9
Jalur terpendek: 0 <- 4

Jarak terpendek titik 4 ke titik 1 = 29
Jalur terpendek: 1 <- 5 <- 4

Jarak terpendek titik 4 ke titik 2 = 41
Jalur terpendek: 2 <- 1 <- 5 <- 4

Jarak terpendek titik 4 ke titik 3 = 61
Jalur terpendek: 3 <- 2 <- 1 <- 5 <- 4

Jarak terpendek titik 4 ke titik 5 = 11
Jalur terpendek: 5 <- 4

Jarak terpendek titik 4 ke titik 6 = 41
Jalur terpendek: 6 <- 5 <- 4

Jarak terpendek titik 4 ke titik 7 = 50
Jalur terpendek: 7 <- 6 <- 5 <- 4

Jarak terpendek titik 4 ke titik 8 = 67
Jalur terpendek: 8 <- 7 <- 6 <- 5 <- 4

Total Jaraknya adalah 67
```

0.2 Hasil Algoritma Floyd-Warshall

0.2.1 Titik A Ke Titik I

```
Matriks jarak terpendek: 8
0       22      INF     INF     9       INF     INF     INF     INF
22      0       12      INF     31      18      INF     INF     INF
INF     12      0       20      INF     INF     13      INF     INF
INF     INF     20      0       INF     INF     INF     INF     10
9       31      INF     INF     0       11      INF     INF     INF
INF     18      INF     INF     11      0       30      INF     INF
INF     INF     13      INF     INF     30      0       9       INF
INF     INF     INF     INF     INF     INF     9       0       17
INF     INF     INF     10      INF     INF     INF     17      0
Matriks jarak terpendek: 8
0       22      34      INF     9       40      INF     INF     INF
22      0       12      INF     31      18      INF     INF     INF
34      12      0       20      43      30      13      INF     INF
INF     INF     20      0       INF     INF     INF     INF     10
9       31      43      INF     0       11      INF     INF     INF
40      18      30      INF     11      0       30      INF     INF
INF     INF     13      INF     INF     30      0       9       INF
INF     INF     INF     INF     INF     INF     9       0       17
INF     INF     INF     10      INF     INF     INF     17      0
Matriks jarak terpendek: 8
0       22      34      54      9       40      47      INF     INF
22      0       12      32      31      18      25      INF     INF
34      12      0       20      43      30      13      INF     INF
54      32      20      0       63      50      33      INF     10
9       31      43      63      0       11      56      INF     INF
40      18      30      50      11      0       30      INF     INF
47      25      13      33      56      30      0       9       INF
INF     INF     INF     INF     INF     INF     9       0       17
INF     INF     INF     10      INF     INF     INF     17      0
Matriks jarak terpendek: 8
0       22      34      54      9       40      47      INF     64
22      0       12      32      31      18      25      INF     42
34      12      0       20      43      30      13      INF     30
54      32      20      0       63      50      33      INF     10
9       31      43      63      0       11      56      INF     73
40      18      30      50      11      0       30      INF     60
47      25      13      33      56      30      0       9       43
INF     INF     INF     INF     INF     INF     9       0       17
64      42      30      10      73      60      43      17      0
Matriks jarak terpendek: 8
0       22      34      54      9       20      47      INF     64
22      0       12      32      31      18      25      INF     42
34      12      0       20      43      30      13      INF     30
54      32      20      0       63      50      33      INF     10
9       31      43      63      0       11      56      INF     73
20      18      30      50      11      0       30      INF     60
47      25      13      33      56      30      0       9       43
INF     INF     INF     INF     INF     INF     9       0       17
```

```
64      42      30      10      73      60      43      17      0
Matriks jarak terpendek: 8
0       22      34      54      9       20      47      INF     64
22      0       12      32      29      18      25      INF     42
34      12      0       20      41      30      13      INF     30
54      32      20      0       61      50      33      INF     10
9       29      41      61      0       11      41      INF     71
20      18      30      50      11      0       30      INF     60
47      25      13      33      41      30      0       9       43
INF     INF     INF     INF     INF     INF     9       0       17
64      42      30      10      71      60      43      17      0
Matriks jarak terpendek: 8
0       22      34      54      9       20      47      56      64
22      0       12      32      29      18      25      34      42
34      12      0       20      41      30      13      22      30
54      32      20      0       61      50      33      42      10
9       29      41      61      0       11      41      50      71
20      18      30      50      11      0       30      39      60
47      25      13      33      41      30      0       9       43
56      34      22      42      50      39      9       0       17
64      42      30      10      71      60      43      17      0
Matriks jarak terpendek: 8
0       22      34      54      9       20      47      56      64
22      0       12      32      29      18      25      34      42
34      12      0       20      41      30      13      22      30
54      32      20      0       61      50      33      42      10
9       29      41      61      0       11      41      50      67
20      18      30      50      11      0       30      39      56
47      25      13      33      41      30      0       9       26
56      34      22      42      50      39      9       0       17
64      42      30      10      67      56      26      17      0
Matriks jarak terpendek: 8
0       22      34      54      9       20      47      56      64
22      0       12      32      29      18      25      34      42
34      12      0       20      41      30      13      22      30
54      32      20      0       61      50      33      27      10
9       29      41      61      0       11      41      50      67
20      18      30      50      11      0       30      39      56
47      25      13      33      41      30      0       9       26
56      34      22      27      50      39      9       0       17
64      42      30      10      67      56      26      17      0
```

0.2.2 Titik B Ke Titik I

```
Matriks jarak terpendek: 8
0       22      12      INF     INF     18      INF     INF     INF
22      0       34      INF     9       40      INF     INF     INF
12      34      0       20      INF     30      13      INF     INF
INF     INF     20      0       INF     INF     INF     INF     10
INF     9       INF     INF     0       11      INF     INF     INF
18      40      30      INF     11      0       30      INF     INF
INF     INF     13      INF     INF     30      0       9       INF
INF     INF     INF     INF     INF     INF     9       0       17
INF     INF     INF     10      INF     INF     INF     17      0
Matriks jarak terpendek: 8
0       22      12      INF     31      18      INF     INF     INF
22      0       34      INF     9       40      INF     INF     INF
12      34      0       20      43      30      13      INF     INF
INF     INF     20      0       INF     INF     INF     INF     10
31      9       43      INF     0       11      INF     INF     INF
18      40      30      INF     11      0       30      INF     INF
INF     INF     13      INF     INF     30      0       9       INF
INF     INF     INF     INF     INF     INF     9       0       17
INF     INF     INF     10      INF     INF     INF     17      0
Matriks jarak terpendek: 8
0       22      12      32      31      18      25      INF     INF
22      0       34      54      9       40      47      INF     INF
12      34      0       20      43      30      13      INF     INF
32      54      20      0       63      50      33      INF     10
31      9       43      63      0       11      56      INF     INF
18      40      30      50      11      0       30      INF     INF
25      47      13      33      56      30      0       9       INF
INF     INF     INF     INF     INF     INF     9       0       17
INF     INF     INF     10      INF     INF     INF     17      0
Matriks jarak terpendek: 8
0       22      12      32      31      18      25      INF     42
22      0       34      54      9       40      47      INF     64
12      34      0       20      43      30      13      INF     30
32      54      20      0       63      50      33      INF     10
31      9       43      63      0       11      56      INF     73
18      40      30      50      11      0       30      INF     60
25      47      13      33      56      30      0       9       43
INF     INF     INF     INF     INF     INF     9       0       17
42      64      30      10      73      60      43      17      0
Matriks jarak terpendek: 8
0       22      12      32      31      18      25      INF     42
22      0       34      54      9       20      47      INF     64
12      34      0       20      43      30      13      INF     30
32      54      20      0       63      50      33      INF     10
31      9       43      63      0       11      56      INF     73
18      20      30      50      11      0       30      INF     60
25      47      13      33      56      30      0       9       43
INF     INF     INF     INF     INF     INF     9       0       17
```

```
42      64      30      10      73      60      43      17      0
Matriks jarak terpendek: 8
0       22      12      32      29      18      25      INF     42
22      0       34      54      9       20      47      INF     64
12      34      0       20      41      30      13      INF     30
32      54      20      0       61      50      33      INF     10
29      9       41      61      0       11      41      INF     71
18      20      30      50      11      0       30      INF     60
25      47      13      33      41      30      0       9       43
INF     INF     INF     INF     INF     INF     9       0       17
42      64      30      10      71      60      43      17      0
Matriks jarak terpendek: 8
0       22      12      32      29      18      25      34      42
22      0       34      54      9       20      47      56      64
12      34      0       20      41      30      13      22      30
32      54      20      0       61      50      33      42      10
29      9       41      61      0       11      41      50      71
18      20      30      50      11      0       30      39      60
25      47      13      33      41      30      0       9       43
34      56      22      42      50      39      9       0       17
42      64      30      10      71      60      43      17      0
Matriks jarak terpendek: 8
0       22      12      32      29      18      25      34      42
22      0       34      54      9       20      47      56      64
12      34      0       20      41      30      13      22      30
32      54      20      0       61      50      33      42      10
29      9       41      61      0       11      41      50      67
18      20      30      50      11      0       30      39      56
25      47      13      33      41      30      0       9       26
34      56      22      42      50      39      9       0       17
42      64      30      10      67      56      26      17      0
Matriks jarak terpendek: 8
0       22      12      32      29      18      25      34      42
22      0       34      54      9       20      47      56      64
12      34      0       20      41      30      13      22      30
32      54      20      0       61      50      33      27      10
29      9       41      61      0       11      41      50      67
18      20      30      50      11      0       30      39      56
25      47      13      33      41      30      0       9       26
34      56      22      27      50      39      9       0       17
42      64      30      10      67      56      26      17      0
```

0.2.3 Titik E Ke Titik I

```
Matriks jarak terpendek: 8
0       9       INF     INF     INF     11      INF     INF     INF
9       0       22      INF     INF     20      INF     INF     INF
INF     22      0       12      INF     18      INF     INF     INF
INF     INF     12      0       20      INF     13      INF     INF
INF     INF     INF     20      0       INF     INF     INF     10
11      20      18      INF     INF     0       30      INF     INF
INF     INF     INF     13      INF     30      0       9       INF
INF     INF     INF     INF     INF     INF     9       0       17
INF     INF     INF     INF     10      INF     INF     17      0
Matriks jarak terpendek: 8
0       9       31      INF     INF     11      INF     INF     INF
9       0       22      INF     INF     20      INF     INF     INF
31      22      0       12      INF     18      INF     INF     INF
INF     INF     12      0       20      INF     13      INF     INF
INF     INF     INF     20      0       INF     INF     INF     10
11      20      18      INF     INF     0       30      INF     INF
INF     INF     INF     13      INF     30      0       9       INF
INF     INF     INF     INF     INF     INF     9       0       17
INF     INF     INF     INF     10      INF     INF     17      0
Matriks jarak terpendek: 8
0       9       31      43      INF     11      INF     INF     INF
9       0       22      34      INF     20      INF     INF     INF
31      22      0       12      INF     18      INF     INF     INF
43      34      12      0       20      30      13      INF     INF
INF     INF     INF     20      0       INF     INF     INF     10
11      20      18      30      INF     0       30      INF     INF
INF     INF     INF     13      INF     30      0       9       INF
INF     INF     INF     INF     INF     INF     9       0       17
INF     INF     INF     INF     10      INF     INF     17      0
Matriks jarak terpendek: 8
0       9       31      43      63      11      56      INF     INF
9       0       22      34      54      20      47      INF     INF
31      22      0       12      32      18      25      INF     INF
43      34      12      0       20      30      13      INF     INF
63      54      32      20      0       50      33      INF     10
11      20      18      30      50      0       30      INF     INF
56      47      25      13      33      30      0       9       INF
INF     INF     INF     INF     INF     INF     9       0       17
INF     INF     INF     INF     10      INF     INF     17      0
Matriks jarak terpendek: 8
0       9       31      43      63      11      56      INF     73
9       0       22      34      54      20      47      INF     64
31      22      0       12      32      18      25      INF     42
43      34      12      0       20      30      13      INF     30
63      54      32      20      0       50      33      INF     10
11      20      18      30      50      0       30      INF     60
56      47      25      13      33      30      0       9       43
INF     INF     INF     INF     INF     INF     9       0       17
```

```
73      64      42      30      10      60      43      17      0
Matriks jarak terpendek: 8
0       9       29      41      61      11      41      INF     71
9       0       22      34      54      20      47      INF     64
29      22      0       12      32      18      25      INF     42
41      34      12      0       20      30      13      INF     30
61      54      32      20      0       50      33      INF     10
11      20      18      30      50      0       30      INF     60
41      47      25      13      33      30      0       9       43
INF     INF     INF     INF     INF     INF     9       0       17
71      64      42      30      10      60      43      17      0
Matriks jarak terpendek: 8
0       9       29      41      61      11      41      50      71
9       0       22      34      54      20      47      56      64
29      22      0       12      32      18      25      34      42
41      34      12      0       20      30      13      22      30
61      54      32      20      0       50      33      42      10
11      20      18      30      50      0       30      39      60
41      47      25      13      33      30      0       9       43
50      56      34      22      42      39      9       0       17
71      64      42      30      10      60      43      17      0
Matriks jarak terpendek: 8
0       9       29      41      61      11      41      50      67
9       0       22      34      54      20      47      56      64
29      22      0       12      32      18      25      34      42
41      34      12      0       20      30      13      22      30
61      54      32      20      0       50      33      42      10
11      20      18      30      50      0       30      39      56
41      47      25      13      33      30      0       9       26
50      56      34      22      42      39      9       0       17
67      64      42      30      10      56      26      17      0
Matriks jarak terpendek: 8
0       9       29      41      61      11      41      50      67
9       0       22      34      54      20      47      56      64
29      22      0       12      32      18      25      34      42
41      34      12      0       20      30      13      22      30
61      54      32      20      0       50      33      27      10
11      20      18      30      50      0       30      39      56
41      47      25      13      33      30      0       9       26
50      56      34      22      27      39      9       0       17
67      64      42      30      10      56      26      17      0
```

0.3 Hasil Algoritma A*

0.3.1 Titik A Ke Titik I

```
Path: 0 -> 4 -> 5 -> 6 -> 7 -> 8
```

0.3.2 Titik B ke Titik I

```
Path: 1 -> 2 -> 6 -> 7 -> 8
```

0.3.3 Titik E ke Titik I

```
Path: 4 -> 5 -> 6 -> 7 -> 8
```

RIWAYAT HIDUP

1. **Identitas Diri**

   (a) Nama Lengkap : Chicha Amalia Putri

   (b) Tempat, tanggal lahir : Jakarta, 30 Mei 2002

   (c) Alamat Rumah : Jalan Pesing Koneng, Kedoya Utara, Kebon Jeruk

   (d) HP : 085773584629

   (e) E-mail : chichaamaliaptr@gmail.com

2. **Riwayat Pendidikan**

   (a) 2008 - 2014 : SDS YP-BDN Jakarta

   (b) 2014 - 2017 : SMPN 191 Jakarta

   (c) 2017 - 2020 : SMAS YP-BDN Jakarta

   (d) 2020 - 2024 : Universitas Islam Negeri Walisongo Semarang

3. **Organisasi**

   (a) 2021 : Anggota Himpunan Mahasiswa Jurusan Matematika UIN Walisongo Semarang

   (b) 2022 - 2023 : Ketua KSR PMI Unit UIN Walisongo Semarang