BAB II
TINJAUAN PUSTAKA

Pada Bab ini berisikan penelitian terdahulu sebagai acuan dalam peneliitian ini. Bab ini juga berisi pengertian algoritma, aplikasi dan objek yang digunakan.

2.1 Pencapain Terdahulu

Penelitian ini menggunakan berbagai referensi terdahulu mengenai penggunaan Algoritma Johnson untuk pencarian jalur terdekat. Dengan referensi terdahulu dapat menjadi acuan dalam melaksanakan penelitian ini. Berikut adalah beberapa penelitian terdahulu yang berhubungan dengan penelitian ini.

Tabel 2.1 Hasil Peneltian Terdahulu

No	Nama (Tahun)	Judul	Hasil
1.	Ana Faridatun Nafiah (2020)	Perancangan Aplikasi Pencarian Rute Terdekat Jasa Binatu *Online* Berbasis Android Dengan Menggunakan Metode Dijkstra	Tujuan dari penelitian ini adalah untuk memberikan informasi kepada konsumen tentang lokasi jasa laundry di daerahnya. Banyak terdapat jasa laundry khususnya di kawasan Ngelom Taman Sidoarjo, namun seringkali sulit untuk mencari informasi yang akurat. Dengan dikembangkannya aplikasi ini, permasalahan tersebut dapat diatasi dengan menggunakan bantuan untuk mencari tempat cuci mobil terdekat dari lokasi pengguna, menyediakan rute terpendek dan memberikan informasi lengkap tentang setiap tempat cuci mobil. Aplikasi ini bertujuan untuk membantu pengguna secara efisien di mana saja dan kapan saja.
2.	Nada Filsa Chaitra (2022	Penentuan Rute Terdekat Tempat Pelayanan Tes Covid-19 Menggunakan Metode Bellman-Ford	Penelitian yang dilakukan bertujuan untuk mengetahui akurasi dari algoritma Bellman-Ford dalam menemukan rute terdekat tempat pelayanan tes Covid-19. Metode Bellman-Ford dipilih karena dapat memberikan pendekatan kesimpulan yang

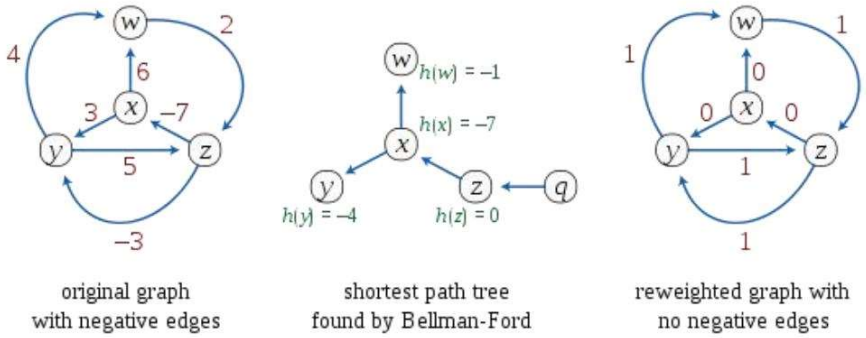
			tepat karena metode ini melakukan penulusuran pada setiap nodenya.
3.	Ester Tetri pada 2019	Pencarian Rute Terdekat Menggunakan Algoritma Bellman-Ford (Studi Kasus: PT. JNE Medan)	Pada penelitian ini peneliti menggunakan 13 nama jalan sebagai input untuk pencarian jalan terdekat. Algoritma Bellman-Ford digunakan untuk menghitung jalur terpendek saat mencari rute terpendek. Hasil penelitian ini menjelaskan jarak minimum yang harus ditempuh seseorang untuk mencapai tujuan.

2.2 Tinjauan Teoritis

Tinjauan teoritis merupakan landasan teori yang terkait dengan pokok masalah dalam penelitian. Tinjauan teori harus mendukung identifikasi terhadap kelanjutan di bab yang selanjutnya. Berikut ini adalah teori-teori berkaitan dengan variabel dalam pokok masalah yang teliti, yaitu.

2.2.1 Algoritma Johnson

Algoritma Johnson adalah metode yang digunakan untuk menemukan jalur terpendek dalam grafik berarah dan berbobot tepi. Algoritma ini berfokus pada masalah optimasi dengan tujuan meminimalkan biaya dan waktu (Mukhtar., 2022). Salah satu keuntungan dari algoritma Johnson adalah dapat menangani banyak sisi bobot dengan nilai negatif, tetapi tidak memungkinkan siklus dengan bobot negatif. Algoritma ini menggunakan algoritma Bellman-Ford untuk menghitung transformasi plot input, yang kemudian menghilangkan semua bobot negatif dan memungkinkan penggunaan algoritma Dijkstra pada plot yang diubah.



Gambar 2.1 *Reweighted Graph*

Algoritma Johnson digunakan untuk menemukan jalur terdekat pada grafik. Berikut adalah langkah-langkah yang diambil algoritma Johnson.

1) Pertama, sebuah simpul baru q ditambahkan ke graf dan terhubung ke setiap simpul lainnya dengan sisi berbobot nol,
2) Kemudian, dimulai dari node baru q, algoritma Bellman-Ford digunakan untuk mencari bobot minimum H(v) dari lintasan dari q ke setiap node v. Jika siklus negatif terdeteksi pada titik ini, algoritma berhenti,
3) Kemudian, tepi grafik asli dibobot ulang dengan nilai yang dihitung oleh algoritma Bellman-Ford. Bobot baru dari titik u ke simpul v adalah W(u, v) + H(u) - H(v),
4) Akhirnya, titik -q dihilangkan dan algoritma Dijkstra digunakan untuk menemukan jalur terpendek dari setiap titik ke setiap simpul lain dari graf yang dibobot ulang.

Grafik di sebelah kiri menunjukkan jalur dengan bobot berbeda. Bagan memiliki dua garis bobot negatif, tetapi tidak ada siklus negatif pada bagan. Sebuah simpul baru q ditambahkan ke plot tengah dan algoritma Bellman-Ford digunakan untuk menghitung jalur terpendek dari q ke setiap simpul lainnya dengan menghitung nilai h(v) pada setiap titik tersebut. Grafik di sebelah kanan menunjukkan bobot yang diubah, di mana semua bobot adalah non-negatif setelah proses perhitungan. Algoritma Johnson kemudian menghilangkan titik-q dan menggunakan algoritma Dijkstra untuk menemukan jalur terpendek dari setiap simpul ke setiap simpul lainnya pada plot berbobot.

2.2.2 Algoritma Bellman-Ford

Algoritma Bellman-Ford adalah algoritma yang digunakan untuk mencari jalur terpendek dalam graf berbobot, termasuk pembobotan negatif. Algoritma Bellman-Ford merupakan pengembangan lebih lanjut dari algoritma Dijkstra dan dapat memberikan hasil yang benar selama tidak ada siklus dengan bobot negatif yang dicapai dari node sumber (Bawole, 2019). Berikut merupakan langkah–langkah cara kerja algoritma Bellman-Ford.

1. Tentukan vertex sumber (source) dan daftar semua vertex serta edges yang ada dalam graf.
2. Assign nilai awal untuk distance dari vertex sumber sebagai 0, dan nilai distance dari vertex lainnya sebagai tak terhingga (infinity).
3. Mulai iterasi untuk semua vertex yang terhubung dengan vertex sumber menggunakan rumus berikut.
   a) U = Vertex asal,
   b) V = Vertex Tujuan,
   c) UV = Edges yang menghubungkan U dan V
   d) Jika distance V lebih besar dari distance U + bobot UV, maka distance V akan diupdate menjadi distance U + bobot UV,
   e) Lanjutkan iterasi hingga semua vertex telah terjelajahi.

Dengan mengikuti langkah-langkah di atas, algoritma Bellman-Ford akan menghitung jalur terpendek dari vertex sumber ke setiap vertex lainnya dalam graf berbobot

2.2.3 Algoritma Dijkstra

Algoritma Dijkstra adalah salah satu algoritma dalam mencari rute. Algoritma ini digunakan untuk mencari lintasan terpendek dengan satu sumber pada sebuah graf. Algoritma Dijkstra dapat menyelesaikan masalah mencari jalur terpendek dengan asumsi bahwa graf yang digunakan tidak memiliki bobot negatif. Algoritma ini menghasilkan jalur terpendek dari sumber ke tujuan yang ditentukan (Cantona, 2020). Berikut adalah langkah-langkah dalam algoritma Dijkstra.

1) Inisialisasi: Setiap simpul dalam graf diberi nilai jarak tak terbatas (*infinity*) kecuali simpul awal, yang diberi nilai jarak 0. Selain itu, setiap

simpul juga diberi nilai awal tak terdefinisi (undefined) sebagai simpul sebelumnya dalam jalur terpendek.

2) Pemilihan simpul terdekat: Algoritma Dijkstra memulai dengan simpul awal dan secara berulang memilih simpul dengan jarak terpendek yang belum dikunjungi. Pemilihan simpul ini dilakukan dengan membandingkan jarak terpendek saat ini ke simpul-simpul yang masih tersisa.
3) Relaksasi: Setelah memilih simpul terdekat, algoritma Dijkstra mengevaluasi setiap tepian yang terhubung dari simpul tersebut. Jika ada tepian (u, v) dengan bobot w, algoritma memeriksa apakah jarak dari simpul awal ke simpul v melalui simpul u lebih kecil dari jarak sebelumnya. Jika ya, maka jarak tersebut diperbarui dan simpul u ditetapkan sebagai simpul sebelumnya dalam jalur terpendek.
4) Tandai simpul yang dikunjungi: Setelah melakukan relaksasi pada semua tepian yang terhubung dari simpul terdekat, simpul tersebut ditandai sebagai sudah dikunjungi. Hal ini memastikan bahwa algoritma tidak akan kembali ke simpul tersebut saat mencari jalur terpendek berikutnya.
5) Ulangi langkah 2-4: Langkah-langkah 2 hingga 4 diulang secara berulang sampai semua simpul telah dikunjungi atau jika simpul tujuan telah dicapai. Algoritma akan berhenti ketika semua simpul telah dikunjungi atau ketika simpul tujuan telah mencapai jarak terpendek.
6) Hasil: Setelah selesai, algoritma Dijkstra menghasilkan hasil akhir dalam bentuk array yang berisi jarak terpendek dari simpul awal ke semua simpul lainnya. Jika terdapat jalur terpendek dari simpul awal ke simpul lainnya, array simpul sebelumnya juga memberikan informasi tentang jalur terpendek tersebut.

2.2.4 *Website*

Website adalah sebuah kumpulan halaman web yang saling terhubung dan dapat diakses melalui internet. Halaman-halaman web ini dapat berisi berbagai jenis informasi, konten, atau aplikasi yang disajikan dalam format teks, gambar, audio, video, dan elemen interaktif lainnya (Romadhon, 2021). *Website* umumnya digunakan untuk tujuan komunikasi, informasi, pendidikan, hiburan, perdagangan

elektronik, atau sebagai wadah untuk berbagi konten dan pengalaman dengan pengguna internet.

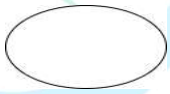
2.2.5 *Unified Modelling Language (UML)*

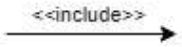
UML atau *Unified Modeling Language* adalah bahasa visual yang digunakan untuk pemodelan dan komunikasi sistem. *UML* menggunakan diagram dan teks pendukung untuk menggambarkan berbagai aspek sistem, termasuk struktur, perilaku, dan interaksi antar komponen (Sudrajat.,2018). Peneliti menggunakan *UML* untuk merancang, memahami, dan mengkomunikasikan sistem perangkat lunak yang kompleks. Berikut beberapa diagram yang terdapat pada *UML*.

a) Use Case Diagram

Use Case diagram adalah diagram dalam *UML* (Unified Modeling Language) yang menggambarkan interaksi antara sistem dan aktor eksternal. Aktor adalah entitas eksternal yang berinteraksi dengan sistem sedangkan *Use Case* adalah fungsi atau fitur dari sistem. Diagram ini membantu memodelkan persyaratan fungsional sistem dan menunjukkan bagaimana sistem digunakan oleh aktor (Nurmiyati, & Wulandari., 2022).

Tabel 2.2 *Use Case* Diagram

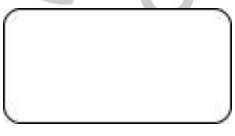
Nama	Simbol	Keterangan
Use Case		Digunakan untuk menunjukkan fungsi atau aktivitas yang dilakukan oleh sistem
Actor		berbentuk orang yang digunakan untuk merepresentasikan pengguna atau sistem lain yang berinteraksi dengan sistem yang diwakili oleh *Use Case* diagram

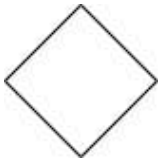
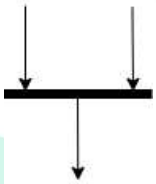
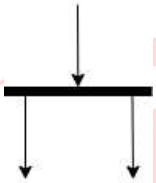
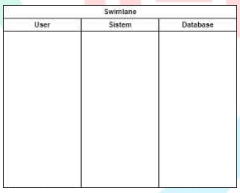
Association		digunakan untuk menggambarkan hubungan antara *actor* dan *Use Case*. *Association* menghubungkan *actor* dengan *Use Case* yang terlibat dalam interaksi, dan menunjukkan bahwa *actor* tersebut terlibat dalam aktivitas *Use Case* tersebut.
Include		digunakan untuk menggambarkan hubungan dimana suatu *Use Case* membutuhkan fungsionalitas yang disediakan oleh *Use Case* lain
Exclude		digunakan untuk menggambarkan hubungan dimana suatu *Use Case* tidak memerlukan atau menghindari fungsionalitas yang disediakan oleh *Use Case* lain

b) *Activity* Diagram

Activity Diagram adalah diagram dalam *UML* yang digunakan untuk menggambarkan alur kerja atau urutan aktivitas dalam suatu proses atau sistem. Hal ini memungkinkan untuk secara visual memodelkan aktivitas, aliran, keputusan, dan kondisi dari suatu proses (Nurmiyati, & Wulandari., 2022).

Tabel 2.3 *Activity* Diagram

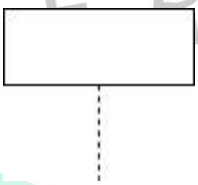
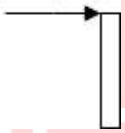
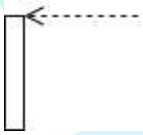
Nama	Simbol	Keterangan
Start Node		digunakan untuk menandakan awal dari aktivitas suatu aktivitas.
Activity		digunakan untuk menyimpan aktivitas yang dilakukan oleh sistem,

Decision		digunakan untuk menggambarkan suatu percabangan atau pilihan aktivitas yang lebih dari satu dalam alur proses.
Final Node		digunakan untuk menandakan akhir dari aktivitas suatu proses.
Join		digunakan untuk menggabungkan kembali suatu *activity* yang parallel.
Fork		digunakan untuk menggambarkan suatu proses *activity* secara parallel.
Swimlane		digunakan sebagai pemisah sebuah sistem yang sedang berjalan

c) *Sequence* Diagram

Sequence diagram adalah diagram dalam *UML* yang menggambarkan urutan interaksi antara objek dari suatu sistem atau proses. Ini memungkinkan untuk memodelkan dalam urutan kronologis pesan yang dikirimkan antar objek (Nurmiyati, & Wulandari., 2022).

Tabel 2.4 *Sequence* Diagram

	Simbol	Keterangan
Actor		digunakan untuk menggambarkan seseorang atau pengguna yang berinteraksi dengan sistem.
Lifelines		berfungsi sebagai *entity* atau antarmuka yang saling berinteraksi.
Activation Box		digunakan untuk mempresentasikan waktu pada suatu objek untuk menyelesaikan tugas tertentu.
Message		digunakan untuk menunjukkan komunikasi tertentu dari sebuah interaksi.
Return Message		digunakan untuk menunjukkan pesan atau Informasi yang dikirimkan kembali menuju suatu objek.

2.2.6 *Hypertext Preprocessor (PHP)*

PHP merupakan singkatan dari *Hypertext Preprocessor*. *PHP* merupakan bahasa pemrograman script yang diletakkan dalam *server* (Winanjar, & Susanti., 2021). *PHP* digunakan untuk membuat halaman *web* dinamis dengan memproses kode sisi *server*, sehingga memungkinkan halaman *web* direnderer secara dinamis berdasarkan interaksi pengguna atau data dari *Database*.

2.2.7 *HyperText Markup Language (HTML)*

Framework merupakan sebuah *Software* atau aplikasi yang bisa dibilang seperti kerangka kerja yang fungsinya untuk memudahkan devloper dalam mengembangkan aplikasi aplikasi yang ada (Hasan, & Muhammad., 2020). *HTML*

terdiri dari serangkaian tag dan atribut yang digunakan untuk menentukan bagaimana konten pada halaman *web* ditampilkan dan diatur. *HTML* digunakan sebagai bahasa *markup* standar pada hampir semua halaman *web*, dengan menggunakan *HTML*, pengembang dapat membuat halaman *web* yang menarik dan terstruktur dengan mudah.

2.2.8 *Bootstrap*

Framework merupakan sebuah *software* atau aplikasi yang bisa dibilang seperti kerangka kerja yang fungsinya untuk memudahkan *developer* dalam mengembangkan aplikasi aplikasi yang ada (Wijaya, & Supariyanto., 2020). *Bootstrap* menyediakan seperangkat alat siap pakai dan komponen antarmuka pengguna, seperti *HTML*, *CSS*, dan *Javascript*, yang memungkinkan pengembang dengan mudah membuat situs *web* responsif dan aplikasi *web* yang disesuaikan dengan kebutuhan mereka. *Bootstrap* dirancang untuk membantu pengembang membangun situs *web* dan aplikasi *web* yang cepat, responsif, dan mudah dikonfigurasi dengan menggabungkan desain responsif, sistem *grid*, dan komponen antarmuka pengguna yang siap digunakan.

2.2.9 *MYSQL*

MYSQL adahlah salah satu jenis *Database server* yang sangat populer, hal ini disebabkan karena *MYSQL* menggunakan bahasa *SQL (Structured Query Language)* untuk mengelola basis data relasional dan menyediakan berbagai fitur dan fungsi, seperti penanganan transaksi, keamanan data, pengelolaan indeks, dan lain-lain. *MYSQL* juga mendukung berbagai jenis operasi basis data, seperti *insert, select, update,* dan *delete*. (Winanjar, & Susanti., 2021). *MYSQL* digunakan oleh banyak perusahaan besar dan aplikasi *web* untuk menyimpan dan mengelola data. *MYSQL* juga mendukung berbagai sistem operasi, seperti Linux, Windows, dan MacOS, serta dapat diintegrasikan dengan berbagai bahasa pemrograman, seperti *PHP*, Java, dan Python.

2.2.10 *PHPMyAdmin*

PHPMyAdmin adalah alat untuk manajemen visual database *MySQL* dan server *MySQL*. *PHPMyAdmin* memungkinkan pengguna untuk dengan mudah melakukan berbagai operasi pada database tanpa menulis kueri *SQL* secara manual. Alat ini sangat populer dan sering menjadi bagian dari paket triad *PHPMyAdmin*,

seperti halnya XAMPP, yang sudah terpasang. *PHPMyAdmin* memungkinkan pengguna untuk mengelola database *MySQL* dengan mudah melalui antarmuka yang intuitif dan *user-friendly*. (Siswanto, & Wibawa., 2021). *PHPMyAdmin* menyediakan berbagai fitur dan fungsi, seperti membuat dan menghapus tabel, mengelola indeks, menjalankan kueri *SQL*, mengimpor dan mengekspor data, dan lainnya. *PHPMyAdmin* juga menyediakan antarmuka yang intuitif dan mudah digunakan, memungkinkan pengguna untuk dengan mudah melakukan tugas *Database MYSQL* yang sederhana dan kompleks.

2.2.11 *XAMPP*

XAMPP adalah perangkat lunak open-source yang digunakan untuk membuat dan mengelola lingkungan pengembangan *web* pada sistem operasi Windows, MacOS, dan Linux. *XAMPP* terdiri dari beberapa aplikasi *server* dan utilitas, termasuk Apache, *MYSQL* , *PHP*, Perl, dan *PHPMyAdmin*. (Siswanto, & Wibawa., 2021). *XAMPP* menyediakan berbagai fitur dan fungsi, seperti manajemen basis data, pemrosesan bahasa skrip, manajemen file, dan lainnya. *XAMPP* juga memiliki antarmuka yang mudah digunakan. Menggunakan *XAMPP*, pengembang dapat membangun dan menjalankan aplikasi *web* secara lokal di komputer tanpa mengakses *server web* jarak jauh. Ini memungkinkan pengembang untuk dengan mudah mengembangkan dan menguji aplikasi *web* secara efisien tanpa mengkhawatirkan masalah konfigurasi *server*.

2.2.12 *JQuery*

JQuery adalah framework atau library JavaScript yang berfokus pada interaksi antara JavaScript dan HTML (Ahmad., 2020). Beberapa fitur utama *JQuery* adalah.

a) Opsi untuk menggunakan elemen dokumen.
   JQuery memungkinkan pengguna untuk dengan mudah mengakses elemen halaman web tertentu tanpa harus mengikuti aturan khusus JavaScript untuk objek dokumen.
b) Ubah tata letak halaman aplikasi.
   JQuery memudahkan untuk mengatur dan mempercantik tampilan halaman web menggunakan *CSS*.
c) Mengubah isi dari dokumen.

JQuery tidak hanya menawarkan tampilan visual yang menyenangkan, tetapi juga memudahkan pengguna untuk dengan mudah mengubah konten dokumen hanya dengan beberapa baris perintah.

d) Merespon interaksi *User.*

JQuery menyediakan berbagai event handler yang memungkinkan pengguna untuk merespon interaksi pengguna seperti klik.

e) Animasi pada dokumen.

Animasi banyak digunakan untuk mempercantik halaman web dan *JQuery* menyediakan kemampuan untuk membuat animasi dengan mudah.

f) Mengambil Informasi dari *server* tanpa harus me-refresh halaman.

JQuery memungkinkan untuk mengambil data dari server menggunakan konsep *AJAX (Asynchronous JavaScript and XML)* tanpa harus me-*refresh* seluruh halaman.

g) Menyederhanakan penelitian sintaks *Javascript*

Moto *JQuery* adalah "menulis lebih sedikit, lakukan lebih banyak", yang berarti pengguna dapat menulis kode dengan tata letak yang lebih sederhana namun menarik.

2.2.12 *EVCS*

EVCS adalah singkatan dari *Electric Vehicle Charging Station*, yang biasanya mengacu pada stasiun pengisian untuk kendaraan listrik. *EVCS* adalah infrastruktur yang dirancang khusus untuk mengisi daya kendaraan listrik, seperti mobil listrik atau sepeda listrik.

Stasiun pengisian kendaraan listrik (*EVCS*) menyediakan energi listrik yang dibutuhkan untuk mengisi baterai kendaraan listrik. Stasiun pengisian ini dapat memiliki jenis dan tingkat kapasitas pengisian yang berbeda, mulai dari stasiun pengisian rumah berkapasitas rendah hingga stasiun pengisian umum berkapasitas lebih tinggi.

Stasiun pengisian kendaraan listrik dapat ditemukan di berbagai lokasi, antara lain rumah pribadi, gedung perkantoran, pusat perbelanjaan, *rest area* di jalan tol, atau stasiun pengisian khusus untuk kendaraan umum.

EVCS memiliki peran penting dalam memfasilitasi penggunaan kendaraan listrik dengan memastikan ketersediaan infrastruktur pengisian daya yang

memadai. Berkat *EVCS* yang besar dan mudah diakses, pengguna kendaraan listrik dapat dengan mudah mengisi ulang baterainya, meningkatkan kenyamanan dan kemudahan penggunaan kendaraan listrik sehari-hari.