IMPLEMENTASI ALGORITMA DIJKSTRA DALAM MENCARI RUTE TERPENDEK DARI UNIVERSITAS NEGERI MEDAN KE MUSEUM NEGERI SUMATERA UTARA

Thania Dealva Arsyad, Musa Dwi Cahyo Nababan, Rut Kezia Imburi, Putri Harliana
Ilmu Komputer, Universitas Negeri Medan
Jalan Williem Iskandar Ps. V Medan, Indonesia
thaniadealva.4233250002@mhs.unimed.ac.id

ABSTRAK

Masalah *Travelling Salesman Problem* (TSP) merupakan persoalan pencarian rute terpendek yang penting dalam meningkatkan efisiensi waktu dan biaya perjalanan. Penelitian ini mengangkat permasalahan variasi jarak dan kondisi jalan antara Universitas Negeri Medan dan Museum Negeri Sumatera Utara, yang memengaruhi waktu tempuh perjalanan. Tujuan dari penelitian ini adalah mengimplementasikan algoritma Dijkstra untuk menentukan rute terpendek antara kedua lokasi tersebut. Metode penelitian yang digunakan adalah eksperimen komputasional dengan pendekatan studi kasus, memanfaatkan data geografis berupa koordinat GPS atau peta yang dimodelkan sebagai graf berbobot. Dalam graf ini, simpul merepresentasikan lokasi, dan sisi merepresentasikan jalur dengan bobot jarak atau waktu tempuh. Hasil penelitian menunjukkan bahwa rute terpendek yang diperoleh dari perhitungan manual dan implementasi algoritma Dijkstra menggunakan Python sesuai dengan rute yang direkomendasikan oleh Google Maps. Rute terdekat yang ditemukan adalah A => B => C => E => F => H => I dengan total jarak 5.627 meter, di mana A merepresentasikan Universitas Negeri Medan, B = Jln. Selamet Ketaren, C = Jln. Williem Iskandar, E = Jln. Aksara, F = Jln. Arief Rahman Hakim, H = Jln. MH. Joni, dan I = Museum Negeri Sumatera Utara. Studi ini diharapkan mampu menjadi referensi untuk penelitian selanjutnya mengenai *Travelling Salesman Problem*.

Kata kunci : *Teori Graph, Travelling Salesman Problem, Algoritma Dijkstra, Algoritma, Pencarian Rute Terdekat.*

1. PENDAHULUAN

Efisiensi waktu dan biaya dalam perjalanan dari satu lokasi ke lokasi lainnya menjadi salah satu kebutuhan utama di era modern. Pemanfaatan model komputasi untuk memecahkan masalah rute perjalanan tidak hanya relevan tetapi juga menjadi kebutuhan yang mendesak, terutama dalam mendukung perencanaan transportasi yang efisien. Salah satu pendekatan yang umum digunakan untuk menyelesaikan masalah rute perjalanan adalah algoritma optimisasi, seperti yang diterapkan dalam TSP (Travelling Salesman Problem). Algoritma yang sering digunakan untuk menyelesaikan masalah pencarian jalur atau jalur terpendek adalah algoritma Dijkstra. Algoritma ini bekerja dengan menentukan jarak terpendek antara dua simpul dalam graf berbobot, memastikan total bobotnya paling kecil. Prosesnya melibatkan pencarian jarak terpendek dari simpul awal ke semua simpul lainnya, sehingga jalur dari simpul awal ke simpul tujuan memiliki bobot total terkecil. [1]

Terdapat beberapa penelitian terkait optimasi rute perjalanan menggunakan algoritma ini, salah satu diantaranya adalah Wulandari, et al [1] yaitu menentukan rute terpendek menuju pelayanan kesehatan dengan algoritma dijkstra. Pada sampel uji dengan lokasi tujuan yang berbeda, hasil pengujian pada aplikasi menampilkan tujuan jalur dari koordinat pengguna. Akan tetapi, jurnal ini tidak menyertakan implementasi kode atau algoritma Dijkstra dalam bentuk pemrograman. Hal ini dapat mengurangi nilai praktis dari penelitian, karena pembaca tidak dapat melihat bagaimana algoritma diterapkan secara konkret dalam kode, sehingga sulit untuk mereplikasi hasil penelitian atau mengadaptasinya untuk aplikasi lain. Selain itu, penelitian oleh Cantona, et al yang berjudul "Implementasi Algoritma Dijkstra Pada Pencarian Rute Terpendek ke Museum di Jakarta" [3] juga menunjukkan bahwa algoritma dijkstra dinilai efektif untuk mendapatkan rute terpendek sebab dengan total 20 bobot hanya memerlukan 7 bobot yang mana hanya perlu 35% bobot untuk mencapai lokasi tujuan. Kekurangan dari jurnal ini adalah, jurnal ini menjelaskan penggunaan algoritma Dijkstra untuk mencari rute terpendek, tetapi tidak menyertakan implementasi kode atau algoritma dalam bentuk pemrograman. Hal ini berarti tidak ada contoh konkret tentang bagaimana algoritma tersebut diterapkan dalam aplikasi yang dikembangkan.

Pada penelitian ini, kasus TSP diaplikasikan untuk menentukan rute terpendek dari Universitas Negeri Medan ke Museum Negeri Sumatera Utara, dengan memanfaatkan algoritma Dijkstra. Algoritma ini sangat efisien dalam menyelesaikan permasalahan jalur optimal untuk semua pasangan simpul dalam graf. [2]

Penelitian ini bertujuan untuk mengimplementasikan algoritma Dijkstra dalam mencari rute terpendek antara Universitas Negeri Medan (UNIMED) dan Museum Negeri Sumatera

Utara (MNSU). Permasalahan yang diangkat adalah banyaknya ruas jalan dengan kondisi dan jarak yang bervariasi, yang mempengaruhi waktu tempuh perjalanan antara kedua lokasi tersebut. Dalam penelitian ini, jaringan jalan yang menghubungkan UNIMED dan MNSU akan dipetakan menjadi sebuah graf, kemudian algoritma Dijkstra akan digunakan untuk menghitung rute terpendek berdasarkan jarak atau waktu tempuh. Hasil implementasi algoritma ini diharapkan dapat memberikan rute yang optimal, yang disajikan dalam bentuk visualisasi peta atau tabel, serta membandingkan beberapa alternatif rute untuk memudahkan pengguna dalam memilih jalur terbaik. Penelitian ini bertujuan untuk memberikan kontribusi dalam pengembangan sistem informasi transportasi dan aplikasi pemetaan rute berbasis algoritma Dijkstra.

Dengan memanfaatkan algoritma Dijkstra, penelitian ini diharapkan dapat memberikan kontribusi praktis dalam pengelolaan transportasi lokal. Rekomendasi rute yang dihasilkan diharapkan tidak hanya mengurangi waktu tempuh dan biaya perjalanan tetapi juga memberikan panduan yang lebih baik bagi pengguna transportasi umum di Kota Medan.

2. TINJAUAN PUSTAKA

2.1. Teori Graph

Teori Graf adalah cabang matematika dan ilmu komputer yang mempelajari struktur graf, yaitu representasi matematis yang terdiri dari himpunan V berisi titik-titik (vertex atau node). Graf dapat direpresentasikan dalam berbagai jenis struktur. Salah satu bentuk graf adalah graf berarah (directed graph), yang memiliki sisi dengan arah tertentu. Jika graf berarah ini juga memiliki bobot pada setiap sisinya, maka graf tersebut disebut jaringan berbobot. [4]

Teori Graf pertama kali dikenalkan oleh seorang ahli matematika Swiss bernama Leonard Euler pada tahun 1736 melalui penyelesaian masalah jembatan Kőnigsberg. Dalam konteks ini, graf adalah representasi matematis yang terdiri dari dua himpunan, yaitu simpul dan sisi, yang didefinisikan sebagai G=(V,E). V adalah himpunan simpul yang tidak kosong, dan E adalah himpunan sisi yang menghubungkan pasangan simpul dalam graf tersebut. [5]

2.2. Travelling Salesman Problem

Masalah Traveling Salesman (TSP) merupakan kasus galat satu yang meningkat optimal yang telah menjadi sorotan perhatian para peneliti selama beberapa dekade. Inti permasalahan TSP adalah bagaimana seseorang salesman mengunjungi sejumlah lokasi menggunakan jeda yg telah diketahui, lalu pulang ke lokasi awal melalui rute terpendek, pada mana setiap lokasi hanya boleh dikunjungi sekali.

Fokus permasalahan TSP ini merupakan bagaimana menemukan rute termurah berdasarkan suatu kota & mengunjungi seluruh kota lainnya menggunakan hanya dikunjungi satu kali buat setiap kota & wajib pulang ke kota asal. [6] [7]

2.3. Algoritma

Algoritma merupakan kemampuan seorang manusia untuk berpikir secara logika tentang suatu permasalahan yang menghasilkan sebuah kebenaran yang dapat dibuktikan.

[8] Suatu algoritma haruslah benar. Artinya apapun masukan (input) yang diberikan, suatu algoritma harus mengeluarkan keluaran (output) yang benar. Suatu algoritma dikatakan benar ketika ia mampu memberikan hasil yang mendekati nilai asli. Algoritma juga haruslah efisien. Jika suatu algoritma memberikan hasil yang benar tetapi memakan waktu yang lama, itu berarti algoritma tersebut belum baik dan harus dioptimalisasi.

2.4. Algoritma Dijkstra

Algoritma Dijkstra adalah algoritma yang menggunakan pendekatan serakah. Istilah “serakah” dalam algoritma Dijkstra mengacu pada pendekatan dimana algoritma selalu memilih node dengan estimasi terendah. Algoritma Dijkstra secara efisien menemukan jalur terpendek pada graf dengan memilih sisi berbobot terkecil yang menghubungkan simpul yang sudah diproses dengan simpul lain yang belum diproses.

[1] Algoritma ini termasuk dalam kategori algoritma pencarian grafik dan dirancang untuk menyelesaikan masalah jalur terpendek sumber tunggal pada grafik yang bukan pohon jalur terpendek. Algoritma Dijkstra digunakan untuk menemukan lintasan terpendek pada graf berarah, namun algoritma ini juga dapat diterapkan pada graf tak berarah. [3] [9] [10]

2.5. Python

Python adalah bahasa pemrograman yang banyak digunakan oleh perusahaan dan pengembang besar untuk mengembangkan berbagai jenis aplikasi, termasuk aplikasi desktop, web, dan seluler. Bahasa ini dikembangkan oleh Guido van Rossum di Belanda pada tahun 1990. Nama Python diambil dari acara televisi favoritnya, Monty Python's Flying Circus. Van Rossum awalnya mengembangkan Python sebagai proyek hobi, namun kemudian menjadi bahasa pemrograman populer di industri dan pendidikan.Kepopulerannya karena perpaduan sifat sederhana ringkas dengan sintaks intuitif serta kumpulan perpustakaan yang sangat luas. [11]

3. METODE PENELITIAN

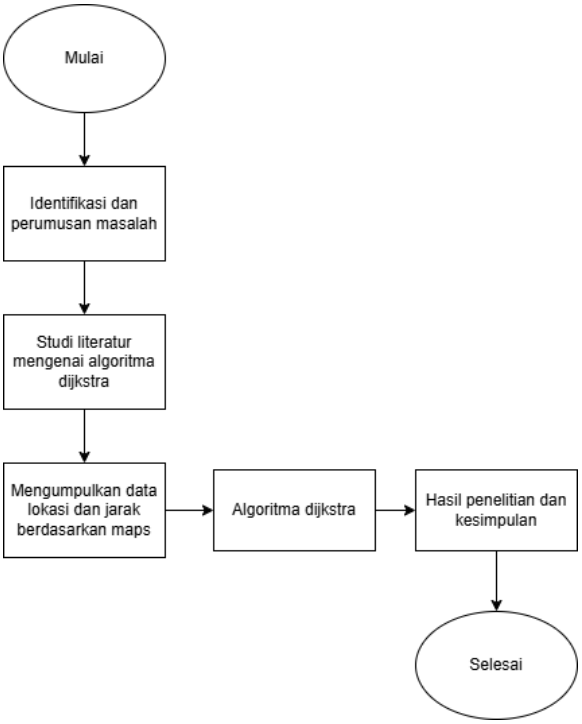
Metode penelitian yang digunakan adalah eksperimen komputasional dengan pendekatan studi kasus, di mana data geografis seperti koordinat GPS atau peta dari Universitas Negeri Medan ke Museum Negeri Sumatera Utara dikumpulkan dan dimodelkan sebagai graf berbobot, dengan simpul merepresentasikan lokasi dan sisi sebagai jalur dengan bobot jarak atau waktu tempuh.

Algoritma Dijkstra diterapkan untuk menghitung rute terpendek. Langkah-langkahnya adalah:

a. Inisialisasi jarak awal ke semua simpul dengan nilai tak hingga (∞), kecuali simpul awal (Universitas Negeri Medan) yang diberi jarak 0.
b. Iterasi melalui simpul-simpul tetangga dan perbarui jarak jika jalur baru lebih pendek.
c. Tandai simpul yang sudah dikunjungi dan lanjutkan ke simpul berikutnya dengan jarak terkecil.
d. Ulangi hingga simpul tujuan (Museum Negeri Sumatera Utara) ditemukan.

Algoritma Dijkstra kemudian diimplementasikan menggunakan bahasa pemrograman seperti Python, dengan library seperti NetworkX untuk mencari rute terpendek. Hasil simulasi diuji dan divalidasi dengan membandingkannya dengan rute dari aplikasi seperti Google Maps, serta dilakukan analisis kuantitatif terhadap waktu eksekusi dan keakuratan algoritma.

3.1. *Flowchart* Penelitian



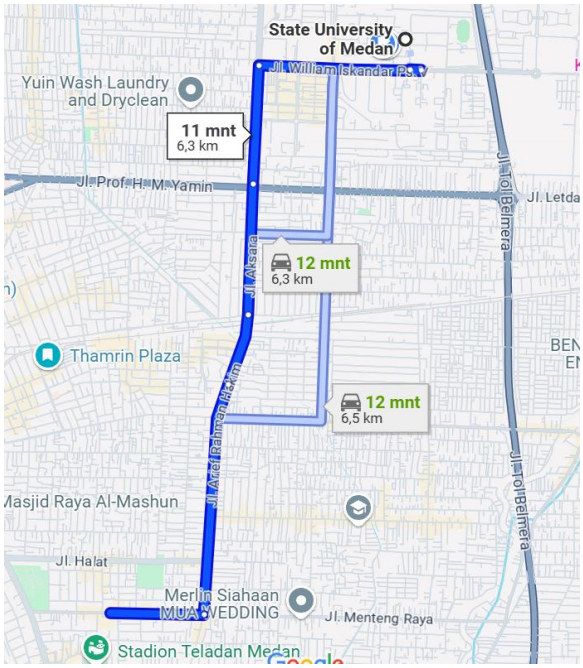
Gambar 1. Flowchart penelitian

4. HASIL DAN PEMBAHASAN

4.1. Pengumpulan Data

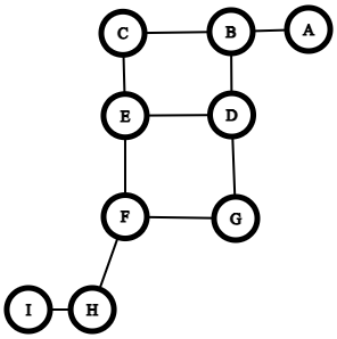
4.1.1. Data Lokasi

Pada tahap ini dilakukan pemetaan lokasi dari Universitas Negeri Medan ke Museum Negeri Sumatera Utara. Pada tahap ini pula ditentukan node awal dan node akhir beserta sisi sisinya. Node awal berada di Universitas Negeri Medan sedangkan node akhir berada di Museum Negeri Sumatera Utara. Sisi/edge ditunjukkan oleh garis berwarna biru, yaitu jalan-jalan yang menghubungkan antara Universitas Negeri Medan dan Museum Negeri Sumatera Utara.



Gambar 2. Peta dari Universitas Negeri Medan ke Museum Negeri Sumatera Utara

Setelah dilakukan pemetaan pada peta, selanjutnya kita akan memvisualisasikan informasi ini ke dalam bentuk graf. Dalam graf ini, setiap persimpangan jalan diwakili sebagai simpul dan setiap jalan penghubungnya diwakili sebagai sisi (edge). Setiap sisi diberi bobot yang merupakan jarak antar jalan-jalannya. Berikut visualisasi graf dari peta di atas.



Gambar 3. Visualisasi graph dari peta

Keterangan:
A = Universitas Negeri Medan
B = Jln. Selamet Ketaren
C = Jln. Williem Iskandar
D = Jln. Pukat II
E = Jln. Aksara
F = Jln. Arief Rahman Hakim
G = Jln. Denai
H = Jln. MH. Joni
I = Museum Negeri Sumatera Utara
Dari keterangan diatas, ditentukan node awal adalah titik yang berhuruf A dan node akhir yang berhuruf I.

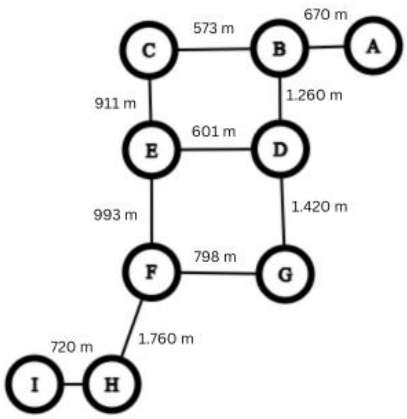
4.1.2. Bobot Graph

Setelah memetakan dan memvisualisasikan peta ke dalam graf, langkah selanjutnya adalah menentukan bobot setiap sisi yang mewakili jarak antar jalan. Berikut rinciannya:

Tabel 1. Tabel bobot sisi graph

No.	Simpul	Total jarak
1.	A → B	670 m
2.	B → C	573 m
3.	B → D	1.260 m
4.	C → E	911 m
5.	D → E	601 m
6.	D → G	1.420 m
7.	E → F	993 m
8.	F → G	798 m
9.	F → H	1.760 m
10.	H → I	720 m

Dari data tabel bobot sisi graph diatas, maka dapat digambarkan graph berbobot sebagai berikut.



Gambar 4. Graph berbobot

4.2. Perhitungan Graph

Setelah dicari bobot setiap sisi pada graph, selanjutnya adalah menetapkan node awal dan node akhir. Dalam hal ini, simpul A adalah node awal dan simpul I adalah node akhir. Langkah-langkah perhitungannya adalah sebagai berikut:

a. Inisialisasi jarak awal dengan tak hingga (∞) kecuali simpul awal dengan jarak 0
   - Jarak A = 0
   - Jarak B = ∞
   - Jarak C = ∞
   - Jarak D = ∞
   - Jarak E = ∞
   - Jarak F = ∞
   - Jarak G = ∞
   - Jarak H = ∞
   - Jarak I = ∞

Tabel 2. Inisialisasi jarak awal

	A	B	C	D	E	F	G	H	I
A	0	∞	∞	∞	∞	∞	∞	∞	∞

Dari tabel 2, kita menginisiasikan jarak awal dengan tak hingga kecuali simpul awal dengan jarak 0.

b. Tentukan simpul terdekat dari simpul awal. Simpul A hanya terhubung dengan simpul B dengan jarak 670 m.

Tabel 3. Update jarak terdekat dari simpul awal

	A	B	C	D	E	F	G	H	I
A	0	670 AB	∞	∞	∞	∞	∞	∞	∞

Dari tabel 3, kita melakukan update jarak dengan simpul terdekat dari simpul awal dan menetapkan jarak tak hingga ke simpul lain yang tidak berdekatan langsung dengan simpul awal. *Highlight* berwarna kuning menunjukkan bahwa simpul tersebut yang akan dikunjungi selanjutnya. Sehingga jarak terpendek diambil dari simpul A ke B.

c. Perhitungan selanjutnya dimulai dari titik yang memiliki bobot terpendek dari simpul B. Simpul B memiliki 2 simpul yang terhubung yaitu simpul C dan simpul D. Simpul B ke simpul C memiliki jarak 573 m, sedangkan simpul B ke simpul D memiliki jarak 1.260 m. Jumlah jarak diambil dari jumlah jarak simpul A ke simpul B dan simpul B ke simpul terhubung.

Tabel 4. Update jarak terdekat

	A	B	C	D	E	F	G	H	I
A	0	670 AB	∞	∞	∞	∞	∞	∞	∞
B	0	670 AB	1243 ABC	1930 ABD	∞	∞	∞	∞	∞

Dari tabel 4, kita melakukan update jarak dari simpul yang berdekatan dengan simpul B. Bisa dilihat dari tabel, simpul yang berdekatan dengan simpul B adalah simpul C dengan bobot total 1.243 dan simpul D dengan bobot 1.930. Dari kedua simpul, diambil bobot yang paling kecil untuk dikunjungi, dalam hal ini simpul C adalah simpul yang memiliki bobot terkecil, ditandai dengan *highlight* berwarna kuning. Sehingga jarak terpendek selanjutnya adalah A ke B ke C.

d. Perhitungan selanjutnya dimulai dari titik yang terhubung dengan simpul C. Simpul C hanya memiliki satu simpul yang terhubung dengannya, yaitu simpul E dengan bobot/jarak 911 m.

Tabel 5. Update jarak terdekat

	A	B	C	D	E	F	G	H	I
A	0	670 AB	∞	∞	∞	∞	∞	∞	∞
B	0	670 AB	1243 ABC	1930 ABD	∞	∞	∞	∞	∞
C	0	670 AB	1243 ABC	1930 ABD	2154 ABCE	∞	∞	∞	∞

Dari tabel 5, kita melakukan update jarak dari simpul yang berdekatan dengan simpul C, yaitu simpul E dengan bobot 2.154. Dari sini bisa kita lihat bahwa jarak ABCE lebih besar dari jarak ABD, maka untuk selanjutnya, simpul yang dipilih adalah dari A ke B ke D.

e. Perhitungan selanjutnya dimulai dari titik yang terhubung dengan simpul D. Simpul D memiliki dua simpul berdekatan yaitu ke simpul E yang berjarak 601 dan simpul G yang berjarak 1.420.

Tabel 6. Update jarak terdekat

	A	B	C	D	E	F	G	H	I
A	0	670 AB	∞	∞	∞	∞	∞	∞	∞
B	0	670 AB	1243 ABC	1930 ABD	∞	∞	∞	∞	∞
C	0	670 AB	1243 ABC	1930 ABD	2154 ABCE	∞	∞	∞	∞
D	0	670 AB	1243 ABC	1930 ABD	2154 ABCE	∞	3350 ABDG	∞	∞

Dari tabel 6, kita melakukan update jarak dari simpul yang berdekatan dengan simpul D yaitu simpul E dan G. Untuk ke simpul E melewati simpul D, diperoleh jarak 2.531 yang dimana diperoleh dari penjumlahan jarak dari A ke B, dari B ke D dan dari D ke E. Jarak ABDE lebih besar dari jarak ABCE, sehingga tidak terjadi update jarak di E. Untuk ke simpul G, diperoleh total jarak 3.350 berdasarkan hasil penjumlahan dari titik A ke B, B ke D, dan D ke G. Sehingga jarak terpendek diambil dari simpul A ke B ke C dan ke E.

f. Perhitungan selanjutnya dimulai dari titik yang terhubung dengan simpul E. Simpul E memiliki 2 simpul yang bertetangga dengannya, yaitu simpul D dan simpul F. Jarak simpul E ke simpul D yaitu 601, sementara jarak simpul E ke simpul F adalah 993. Jumlah jarak diambil dari total jarak A ke B, B ke C, C ke E, dan E ke simpul terdekat.

Tabel 7. Update jarak terdekat

	A	B	C	D	E	F	G	H	I
A	0	670 AB	∞	∞	∞	∞	∞	∞	∞
B	0	670 AB	1243 ABC	1930 ABD	∞	∞	∞	∞	∞
C	0	670 AB	1243 ABC	1930 ABD	2154 ABCE	∞	∞	∞	∞
D	0	670 AB	1243 ABC	1930 ABD	2154 ABCE	∞	3350 ABDG	∞	∞
E	0	670 AB	1243 ABC	1930 ABD	2154 ABCE	3147 ABCEF	3350 ABDG	∞	∞

Dari tabel 7, kita melakukan update jarak dari simpul yang berdekatan dengan simpul E yaitu simpul D dan simpul F. Jarak ke simpul D melalui ABCE, adalah 2.755 sehingga tidak ada update jarak di D. Jarak ABCEF adalah 3.147 dimana jarak ini lebih kecil dari jarak ABDG, sehingga jarak terpendek selanjutnya adalah A ke B ke C ke E ke F.

g. Perhitungan selanjutnya dimulai dari jarak yang berdekatan dengan simpul F. Simpul F berdekatan dengan simpul G dan simpul H, tetapi kita tidak akan menghitung jarak antara F ke G dikarenakan, simpul G tidak memiliki keterikatan dengan simpul akhir. Jarak simpul F ke simpul H adalah 1.760. Jumlah jarak terdekat selanjutnya diperoleh dari total jarak A ke B, B ke C, C ke E, E ke F, dan F ke H. Untuk lebih jelasnya, dapat dilihat pada tabel 8 di bawah ini.

Tabel 8. Update jarak terdekat

	A	B	C	D	E	F	G	H	I
A	0	670 AB	∞	∞	∞	∞	∞	∞	∞
B	0	670 AB	1243 ABC	1930 ABD	∞	∞	∞	∞	∞
C	0	670 AB	1243 ABC	1930 ABD	2154 ABCE	∞	∞	∞	∞
D	0	670 AB	1243 ABC	1930 ABD	2154 ABCE	∞	3350 ABDG	∞	∞

	A	B	C	D	E	F	G	H	I
E	0	670 AB	1243 ABC	1930 ABD	2154 ABCE	3147 ABCEF	3350 ABDG	∞	∞
F	0	670 AB	1243 ABC	1930 ABD	2154 ABCE	3147 ABCEF	3350 ABDG	4907 ABCEFH	∞

Dari tabel 8, dapat dilihat jarak ABDG lebih kecil daripada jarak ABCEFH, tetapi kita tidak akan mengambil ABDG sebagai jalur selanjutnya, karena untuk mencapai H, G harus melewati F terlebih dahulu, sementara jarak ABDGF lebih besar daripada jarak ABCEF. Jadi, kita akan mengambil jarak ABCEFH sebagai jalur selanjutnya.

h. Perhitungan selanjutnya dimulai dari simpul H. Simpul H berdekatan dengan simpul I dengan bobot sebesar 720 m. Untuk lebih jelasnya dapat dilihat pada tabel 9 di bawah ini.

Tabel 9. Update jarak terdekat

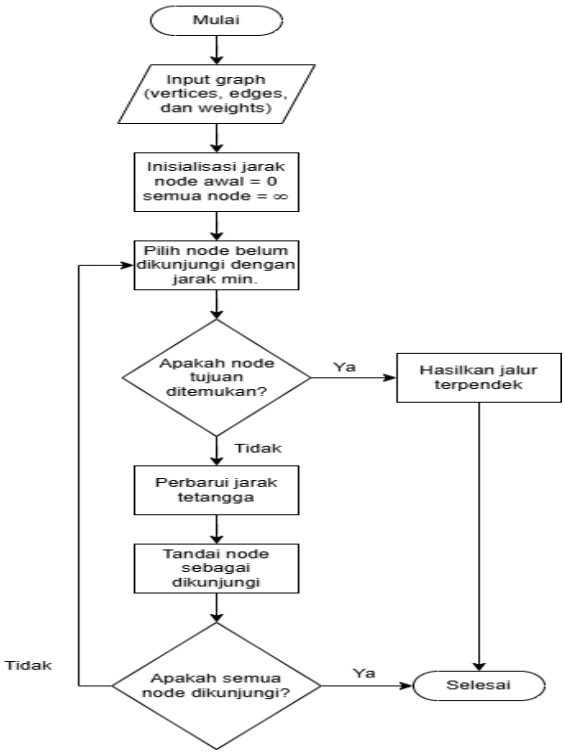
	A	B	C	D	E	F	G	H	I
A	0	670 AB	∞	∞	∞	∞	∞	∞	∞
B	0	670 AB	1243 ABC	1930 ABD	∞	∞	∞	∞	∞
C	0	670 AB	1243 ABC	1930 ABD	2154 ABCE	∞	∞	∞	∞
D	0	670 AB	1243 ABC	1930 ABD	2154 ABCE	∞	3350 ABDG	∞	∞
E	0	670 AB	1243 ABC	1930 ABD	2154 ABCE	3147 ABCEF	3350 ABDG	∞	∞
F	0	670 AB	1243 ABC	1930 ABD	2154 ABCE	3147 ABCEF	3350 ABDG	4907 ABCEFH	∞
H	0	670 AB	1243 ABC	1930 ABD	2154 ABCE	3147 ABCEF	3350 ABDG	4907 ABCEFH	5627 ABCEFHI

Dari tabel 9, dapat dilihat bahwa jarak terpendek diperoleh dari A ke B ke C ke E ke F ke H ke I dengan bobot total 5.627

4.3. Implementasi Algoritma Dijkstra

Pada tahap ini, kita akan membuktikan hasil perhitungan yang telah dijelaskan sebelumnya dengan menggunakan bahasa pemrograman Python. Namun, sebelum langsung masuk ke implementasi Python, terlebih dahulu kita akan menampilkan *flowchart* dari algoritma Dijkstra. Hal ini bertujuan untuk memberikan gambaran langkah-langkah utama dari algoritma secara lebih terstruktur dan visual.

Flowchart



Gambar 5. Flowchart algoritma dijkstra

Flowchart pada gambar di atas menunjukkan tahapan-tahapan dari algoritma dijkstra untuk mencari rute terpendek dalam sebuah graf. Berikut adalah penjelasan dari flowchart di atas:

a. Mulai
   Proses dimulai dengan menjalankan algoritma untuk mencari jalur terpendek.
b. Input graph (vertices, edges, dan weights):
   Masukkan informasi graf, termasuk node (vertices), hubungan antar node (edges), dan bobot atau jarak antar node (weights).
c. Inisialisasi jarak:
   - Tetapkan jarak dari node asal ke dirinya sendiri sebagai 0.
   - Semua node lainnya diatur memiliki jarak awal tak terhingga (∞) karena belum dihitung.
   - Pilih node yang belum dikunjungi dengan jarak minimum:

   Dari semua node yang belum diproses, pilih node dengan nilai jarak minimum.
   - Apakah node tujuan ditemukan?
   - Jika node tujuan sudah ditemukan, hasilkan jalur terpendek, dan algoritma selesai.
   - Jika belum, lanjutkan ke langkah berikutnya.
   - Perbarui jarak tetangga:

   Untuk setiap node tetangga dari node yang dipilih, hitung jarak baru melalui node tersebut. Jika jarak baru lebih kecil dari jarak sebelumnya, perbarui jaraknya.
   - Tandai node sebagai dikunjungi:

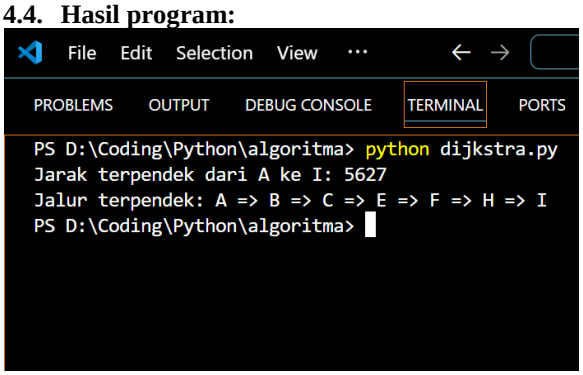
   Tandai node yang sudah diproses agar tidak dipilih kembali di langkah berikutnya.

d. Apakah semua node dikunjungi?

a. Jika semua node sudah dikunjungi, proses selesai.
b. Jika belum, kembali ke langkah memilih node dengan jarak minimum.

e. Selesai1
   Algoritma selesai setelah semua node diproses atau jalur terpendek ditemukan.
   Berdasarkan *flowchart* di atas, berikut hasil dari algoritmanya.

4.4. Hasil program:

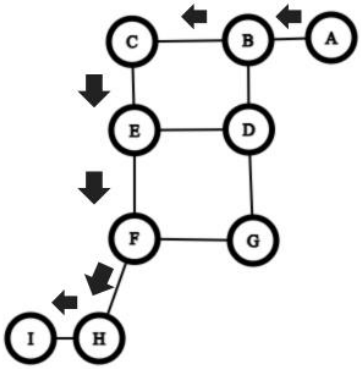


Gambar 6. Hasil program

Dari gambar di atas, dapat dilihat bahwa hasil dari program python sesuai dengan perhitungan manual yang telah kita lakukan.

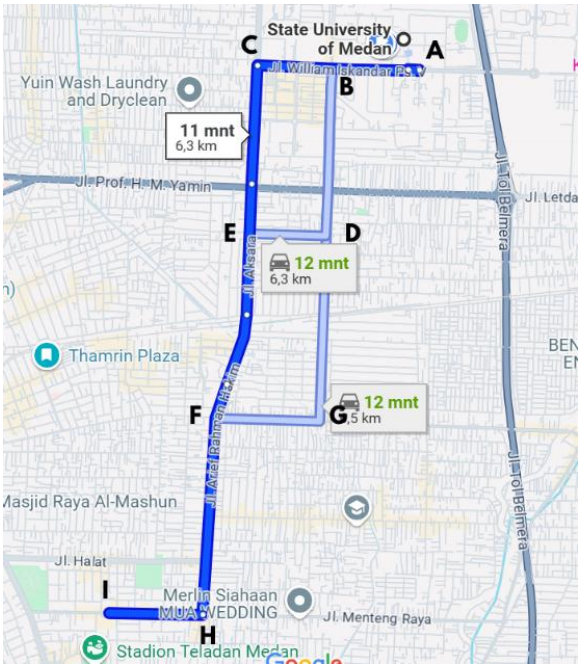
4.5. Analisis Hasil

Untuk menentukan rute terpendek antara Universitas Negeri Medan (Unimed) dan Museum Negeri Sumatera Utara, program ini menggunakan algoritma Dijkstra. Pertama, grafik dibuat berdasarkan peta jalan yang diperoleh dari sumber peta online, di mana titik-titik (node) dan jalur (edge) ditentukan dengan mengacu pada peta tersebut, beserta jarak antar titik yang relevan. Hasil perhitungan manual menggunakan algoritma Dijkstra menunjukkan bahwa rute terpendek yang dihitung sesuai dengan rekomendasi arah yang diberikan oleh aplikasi peta yang digunakan. Setelah itu, program Python yang diimplementasikan untuk perhitungan rute terpendek juga menghasilkan hasil yang identik dengan rekomendasi peta, yang menunjukkan bahwa algoritma Dijkstra yang diterapkan dalam program bekerja dengan baik dan sesuai dengan kenyataan di peta. Dengan demikian, perbandingan antara perhitungan manual dan hasil yang diperoleh dari program Python menunjukkan kesesuaian yang signifikan, yang memberikan validasi bahwa implementasi algoritma dalam program mampu menghasilkan perhitungan yang akurat dan sesuai dengan ekspektasi. Berikut ini adalah gambaran dari rute yang didapat dari hasil perhitungan.



Gambar 7. Visualisasi rute terdekat dari A ke I

Pada gambar di atas, dapat dilihat rute terdekat dari A menuju I adalah A => B => C => E => F => H => I yang sesuai dengan petunjuk arah dari *google maps* yang ditunjukkan oleh gambar di bawah.



Gambar 8. Rute google maps dari Universitas Negeri Medan ke Museum Negeri Sumatera Utara

Pada gambar 8, rute terdekat dari Universitas Negeri Medan ke Museum Negeri Sumatera Utara ditunjukkan oleh garisi yang berwarna biru, sesuai dengan yang diperhitungkan manual dengan algoritma dijkstra.

5. KESIMPULAN DAN SARAN

Berdasarkan penelitian yang telah dilakukan, didapatkan hasil bahwa algoritma dijkstra menemukan rute terpendek dari Universitas Negeri Medan ke Museum Negeri Sumatera Utara. Penggunaan algoritma Dijkstra untuk menentukan rute terpendek antara Universitas Negeri Medan (Unimed) dan Museum Negeri Sumatera Utara telah berhasil diimplementasikan dengan baik. Perhitungan manual yang dilakukan menggunakan algoritma tersebut sesuai dengan hasil yang diperoleh dari aplikasi peta online, yang memberikan validasi terhadap akurasi perhitungan. Selain itu, implementasi program Python yang menggunakan algoritma Dijkstra juga menghasilkan hasil yang identik dengan rekomendasi peta, menunjukkan bahwa program ini dapat diandalkan untuk menghitung rute terpendek dengan akurat dan efisien. Dengan demikian, algoritma Dijkstra terbukti efektif dalam menyelesaikan masalah rute terpendek dalam konteks yang diuji.

Sebagai saran, penelitian ini dapat dikembangkan lebih lanjut dengan menambahkan elemen-elemen lain yang dapat mempengaruhi pemilihan rute, seperti kondisi lalu lintas atau waktu tempuh pada jam-jam tertentu. Selain itu, implementasi algoritma Dijkstra dapat diperluas untuk mengakomodasi grafik yang lebih kompleks dengan banyak titik dan jalur, guna meningkatkan fleksibilitas program dalam berbagai situasi. Pengujian pada skala yang lebih besar juga dapat dilakukan untuk menguji kecepatan dan efisiensi algoritma, terutama pada peta dengan banyak node dan edge. Dengan pengembangan lebih lanjut, sistem ini dapat diadaptasi untuk berbagai aplikasi lain yang membutuhkan perhitungan rute terpendek.

DAFTAR PUSTAKA

[1] Wulandari and P. Sukmasetyan, "Implementasi Algoritma Dijkstra untuk Menentukan Rute Terpendek Menuju Pelayanan Kesehatan," *Jurnal Ilmiah Sistem Informasi*, vol. 1, no. 1, pp. 30-37, 2022

[2] E. C. Rufus, R. R. Riyadi, D. N. Hasibuan, E. Christian and V. H. Pranatawijaya, "Penerapan Algoritma Dijkstra Dalam Menentukan Rute Terpendek Untuk Jasa Pengiriman Barang di Palangka Raya," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 8, no. 3, pp. 3387-3391, 2024.

[3] A. Cantona, Fauziah and Winarsih, "Implementasi Algoritma Dijkstra Pada Pencarian Rute Terpendek ke Museum di Jakarta," *Jurnal Teknologi dan Manajemen Informatika*, vol. 6, no. 1, pp. 27-34, 2020

[4] F. Mahardika, "Penerapan Teori Graf Pada Jaringan Komputer Dengan," *Jurnal Informatika: Jurnal Pengembangan IT (JPIT)*, vol. 4, no. 1, pp. 48-53, 2019

[5] I. M. A. Santosa, I. P. Ramayasa and G. A. S. Wahyuni, "Implementasi Metode Graph Coloring Untuk Pemetaan Daerah Rawan Kriminal," *Jurnal Teknologi Informasi dan Komputer*, vol. 7, no. 4, pp. 421-428, 2021

[6] R. P. Sinaga and F. Marpaung, "Perbandingan Algoritma Cheapest Insertion Heuristic Dan Nearest Neighbor Dalam Menyelesaikan Travelling Salesman Problem," *Jurnal Riset Rumpun Matematika dan Ilmu Pengetahuan Alam (JURRIMIPA)*, vol. 2, no. 2, pp. 238-247, 2023.

[7] J. E. Simarmata, E. Rosmaini and N. Napitupulu, "Penerapan Algoritma Branch and Bound Pada Persoalan Pedagang Keliling (Travelling Salesman Problem)," *Jurnal Pendidikan Matematika*, vol. 1, no. 2, pp. 111-121, 2020.

[8] A. M. Reta, A. Isroqmi and T. D. Nopriyanti, "Pengaruh Penerapan Algoritma Terhadap Pembelajaran Pemrograman Komputer," *INDIKTIKA (Jurnal Inovasi Pendidikan Matematika*, vol. 2, no. 2, pp. 126-135, 2020

[9] R. Y. Sumaryo, P. Harsadi and D. Nugroho, "Implementasi Algoritma Dijkstra Dan Metode Haversine Pada Penentuan Jalur Terpendek Pendakian Gunung Merapi Jalur Selo Berbasis Android," *Jurnal TIKomSIN*, vol. 8, no. 1, pp. 61-67, 2020

[10] R. Perayoga, P. Hendradi and A. Setiawan, "Implementasi Algoritma Dijkstra Pada Pencarian Rute Terpendek Objek Wisata," *KLIK: Kajian Ilmiah Informatika dan Komputer*, vol. 4, no. 3, pp. 1471-1482, 2023

[11] M. Romzi and B. Kurniawan, "Pembelajaran Pemrograman Python Dengan Pendekatan Logika Informatika," *JTIM: Jurnal Teknik Informatika Mahakarya*, vol. 3, no. 2, pp. 37-44, 2020.