KASUS PENCARIAN RUTE TERPENDEK DARI RUMAH ALIEF (LEMBANG) KE KAMPUS UNIKOM (DIPATIUKUR)

Disusun Untuk Memenuhi Tugas Besar Mata Kuliah Riset Operasional

Disusun Oleh:

Rifai Nugroho	10121295
Revan Fakhriansyah	10121296
Much Firman Alfi Hamdani	10121301
Alief Sidik Gunawan	10121305
Cecep Ramdan Suryadi	10121307
Rizki Ananda Saputra	10121907

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK DAN ILMU KOMPUTER

UNIVERSITAS KOMPUTER INDONESIA

2023

Menentukan Rute Terpendek Dari Rumah Alief (Lembang) ke Kampus Unikom (Dipatiukur)

Rifai Nugroho[1], Revan Fakhriansyah [2] , Much Firman Alfi Hamdani [3] ,
Alief Sidik Gunawan[4] ,Cecep Ramdan Suryadi[5] , Rizki Ananda Saputra [6]

10121295[1] , 10121296[2] , 10121301[3] , 10121305[4] ,10121307[5] , 10121907[6]

[1)]Program Studi Teknik Informatika, Fakultas Teknik dan Ilmu Komputer, Universitas Komputer Indonesia
Jl. Dipati Ukur No. 112 – 116, Bandung, Indonesia 40132

ABSTRAK *– Penentuan rute terpendek dalam perjalanan telah menjadi aspek penting dalam mobilitas modern. Penelitian ini bertujuan untuk mengeksplorasi dan menerapkan metode algoritma *brute force* dalam menentukan rute terpendek dari Rumah Alief di Lembang menuju Kampus Universitas Komputer Indonesia (Unikom) yang terletak di Dipatiukur. Metode algoritma *brute force* digunakan untuk melakukan pencarian seluruh kemungkinan rute dengan mempertimbangkan jarak dan topologi jaringan jalan tanpa memerlukan struktur data yang kompleks. Pada tahap awal, data mengenai jalan, jarak, dan kondisi lalu lintas dianalisis untuk membangun representasi jaringan jalan antara Rumah Alief dan Kampus Unikom. Kemudian, metode algoritma *brute force* diterapkan untuk menghitung total jarak dari setiap rute yang mungkin dilalui. Meskipun pendekatan ini memiliki kompleksitas yang meningkat secara eksponensial seiring bertambahnya simpul dan panjang rute, metode *brute force* memiliki potensi untuk memberikan solusi yang akurat dan pasti. Hasil penelitian menunjukkan bahwa metode algoritma *brute force* mampu menemukan rute terpendek dari Rumah Alief ke Kampus Unikom dengan tingkat akurasi yang tinggi. Namun, disadari bahwa pendekatan *brute force* memiliki keterbatasan dalam efisiensi komputasi terutama pada kasus dengan jaringan jalan yang kompleks. Oleh karena itu, rekomendasi tambahan adalah untuk menerapkan algoritma ini pada skala yang terukur dan mempertimbangkan penggunaannya dalam konteks yang lebih spesifik. Penelitian ini memiliki potensi untuk memberikan wawasan tentang penerapan metode *brute force* dalam masalah penentuan rute terpendek. Namun, untuk skenario perjalanan yang lebih kompleks dan mempertimbangkan faktor waktu dan kondisi lalu lintas yang dinamis, pendekatan algoritma lain yang lebih canggih, seperti algoritma Dijkstra atau algoritma A*, mungkin lebih sesuai dalam mengoptimalkan waktu komputasi dan akurasi solusi.*

Kata Kunci *: Algoritma BruteForce, Rute tercepat dengan BruteForce, Lembang ke Unikom. Ilmu algoritma BruteForce*

Determine the Shortest Route from Lembang to the Unikom Campus (Dipatiukur)

*ABSTRACT – Determining the shortest route for travel has become an important aspect of modern mobility. This study aims to explore and apply the *brute force* algorithm method in determining the shortest route from Alief's house in Lembang to the Indonesian Computer University (Unikom) campus located in Dipatiukur. The *brute force* algorithm method is used to search all possible routes by considering the distance and road network topology without requiring complex data structures. In the early stages, data regarding roads, distances, and traffic conditions were analyzed to build a representation of the road network between Alief's House and the Unikom Campus. Then, the *brute force* algorithm method is applied to calculate the total distance of each possible route. Although this approach has exponentially increasing complexity as nodes and route length increase, the *brute force* method has the potential to provide accurate and definitive solutions. The results showed that the *brute force* algorithm method was able to find the shortest route from Alief's House to the Unikom Campus with a high degree of accuracy. However, it is realized that the *brute force* approach has limitations in computational efficiency, especially in cases with complex road networks. Therefore, an additional recommendation is to apply this algorithm on a scalable scale and consider its use in more specific contexts. This research has the potential to provide insight into the application of the *brute force* method in determining the shortest route. However, for more complex travel scenarios and considering time factors and dynamic traffic conditions, other, more sophisticated algorithmic approaches, such as Dijkstra's algorithm or A* algorithm, may be more suitable in optimizing computation time and solution accuracy.*

Keywords *: BruteForce Algorithm, The fastest route with BruteForce, Lembang to Unikom. BruteForce algorithm science*

---

1. PENDAHULUAN

Dalam era mobilitas yang semakin berkembang, penentuan rute terpendek dan efisien dalam perjalanan memiliki peran yang krusial. Kemajuan teknologi dan ketersediaan informasi telah memungkinkan pengguna untuk memiliki akses lebih baik terhadap panduan perjalanan yang akurat dan cepat. Salah satu tantangan utama dalam konteks ini adalah menemukan rute terpendek dari suatu lokasi ke tujuan tertentu. Penelitian ini berfokus pada penentuan rute terpendek dari Rumah Alief di Lembang menuju Kampus Universitas Komputer Indonesia (Unikom) yang terletak di Dipatiukur.

Dalam mengatasi tantangan ini, berbagai pendekatan telah dikembangkan, termasuk metode *brute force*. Metode ini, meskipun sederhana dalam konsep, memiliki potensi untuk memberikan solusi yang akurat tanpa memerlukan kompleksitas struktur data. Melalui pendekatan ini, seluruh kemungkinan rute dijelajahi untuk menemukan yang memiliki total jarak terpendek.

Pada awal penelitian, data mengenai jalan, jarak, serta kondisi lalu lintas dianalisis untuk membangun representasi jaringan jalan antara titik awal (Rumah Alief) dan titik tujuan (Kampus Unikom). Penggunaan metode *brute force* dalam konteks ini menjadi esensi untuk memahami potensi serta keterbatasannya dalam mencari rute terpendek.

Pentingnya penelitian ini tidak hanya dalam aspek mobilitas sehari-hari, tetapi juga dalam pengembangan teknologi navigasi dan aplikasi berbasis peta yang semakin canggih. Dengan demikian, pendekatan yang digunakan dan temuan dalam penelitian ini akan memberikan kontribusi dalam merancang solusi navigasi yang lebih baik di masa depan. Meskipun begitu, penting juga untuk mengakui bahwa dalam kasus perjalanan yang lebih kompleks, seperti faktor waktu tempuh dan kondisi lalu lintas yang bervariasi, pendekatan lain yang lebih canggih mungkin diperlukan untuk memberikan solusi yang optimal.

Pada latar belakang ini, penelitian ini bertujuan untuk menjelajahi potensi dan keterbatasan metode algoritma *brute force* dalam menentukan rute terpendek dari Rumah Alief ke Kampus Unikom. Dalam konteks peningkatan mobilitas dan kebutuhan akan panduan perjalanan yang akurat, pendekatan ini menjadi subjek penelitian yang relevan dan menarik untuk diinvestigasi lebih lanjut.

1.1 Latar belakang

Dalam kehidupan sehari-hari, mobilitas dan perjalanan menjadi bagian integral dari aktivitas manusia. Dengan kemajuan teknologi, terutama di bidang transportasi dan informasi, penggunaan navigasi untuk menemukan rute terpendek dan efisien telah menjadi semakin umum. Penentuan rute terpendek tidak hanya menghemat waktu, tetapi juga berkontribusi pada efisiensi bahan bakar dan mengurangi dampak lingkungan.

Lembang, sebuah kota indah di Indonesia, memiliki peran penting dalam pariwisata dan kehidupan sehari-hari. Dalam konteks ini, banyak orang memerlukan panduan perjalanan yang tepat dan efektif, terutama saat mereka ingin mencapai tujuan tertentu seperti Kampus Universitas Komputer Indonesia (Unikom) yang terletak di Dipatiukur. Untuk memenuhi kebutuhan ini, solusi navigasi yang mampu menemukan rute terpendek menjadi sangat penting.

Meskipun ada banyak pendekatan algoritma yang digunakan untuk menentukan rute terpendek, seperti algoritma Dijkstra, algoritma A*, dan algoritma Floyd-Warshall, pendekatan *brute force* juga memiliki tempatnya dalam literatur penelitian. *Brute force* adalah metode eksplorasi yang langsung menghitung seluruh kemungkinan solusi tanpa mengandalkan pengoptimalan atau heuristik. Ini dapat memberikan jawaban pasti, namun sering kali memerlukan waktu komputasi yang signifikan, terutama saat kompleksitas masalah meningkat. Dalam konteks penelitian ini, penerapan metode *brute force* untuk menentukan rute terpendek dari Rumah Alief di Lembang ke Kampus Unikom di Dipatiukur menjadi fokus utama. Penggunaan pendekatan ini akan memberikan wawasan tentang kemampuan metode *brute force* dalam mengatasi masalah penentuan rute terpendek dalam konteks perkembangan teknologi dan mobilitas.

Dengan menggabungkan pengetahuan mengenai geografi, jaringan jalan, serta teknologi komputasi, penelitian ini diharapkan akan memberikan pemahaman yang lebih baik tentang kemampuan dan batasan algoritma *brute force* dalam memecahkan masalah penentuan rute terpendek. Dalam kerangka ini, penelitian ini memiliki potensi untuk memberikan wawasan yang bermanfaat bagi pengembangan solusi navigasi dan mobilitas yang lebih baik di masa depan.

1.2 Tujuan dan Manfaat

Tujuan Penelitian:

Penelitian ini memiliki tujuan untuk menerapkan dan mengevaluasi metode algoritma *brute force* dalam menentukan rute terpendek dari Rumah Alief di Lembang ke Kampus Universitas Komputer Indonesia (Unikom) yang terletak di Dipatiukur. Tujuan utama dari penelitian ini adalah untuk menguji efektivitas dan akurasi metode *brute force* dalam menemukan rute terpendek dalam konteks perjalanan sehari-hari.

Manfaat Penelitian:

Penelitian ini memiliki manfaat yang berpotensi untuk:

1. Solusi Navigasi yang Akurat: Hasil dari penelitian ini dapat memberikan kontribusi dalam pengembangan solusi navigasi yang lebih akurat dan dapat diandalkan. Metode *brute force* dapat memberikan solusi pasti dalam menentukan rute terpendek, yang dapat digunakan sebagai acuan dalam perancangan aplikasi navigasi.
2. Pemahaman tentang Metode Brute Force: Penelitian ini akan memberikan wawasan lebih dalam tentang potensi dan keterbatasan metode algoritma *brute force* dalam konteks penentuan rute terpendek. Hal ini dapat membantu para peneliti dan praktisi dalam memilih metode yang paling sesuai untuk masalah spesifik.
3. Basis Penelitian Selanjutnya: Hasil dari penelitian ini dapat menjadi dasar bagi penelitian-penelitian berikutnya dalam mengembangkan pendekatan baru atau menggabungkan metode *brute force* dengan teknik lain untuk mengoptimalkan solusi penentuan rute terpendek.
4. Pengembangan Aplikasi Navigasi Lokal: Hasil penelitian ini dapat digunakan dalam pengembangan aplikasi navigasi lokal yang dapat membantu pengguna dalam perjalanan sehari-hari, terutama dalam konteks kota seperti Lembang.
5. Kontribusi pada Pendidikan: Penelitian ini dapat memberikan kontribusi pada bidang pendidikan dan akademik, terutama dalam mata kuliah yang berkaitan dengan algoritma dan pemrosesan data, dengan menyajikan contoh penerapan metode *brute force* dalam dunia nyata.

Melalui tujuan dan manfaat yang diuraikan di atas, penelitian ini diharapkan dapat memberikan kontribusi yang berharga dalam memahami dan memecahkan masalah penentuan rute terpendek dalam konteks mobilitas modern.

1.3 Batasan Masalah

Dalam penelitian ini, beberapa batasan masalah diidentifikasi untuk menjaga fokus dan relevansi dari penelitian:

1. Pemilihan Lokasi: Penelitian ini hanya mempertimbangkan rute terpendek dari Rumah Alief di Lembang menuju Kampus Universitas Komputer Indonesia (Unikom) yang terletak di Dipatiukur. Lokasi lain di luar rute ini tidak dipertimbangkan.
2. Pemilihan Metode: Penelitian ini difokuskan pada penerapan metode algoritma *brute force* dalam menentukan rute terpendek. Metode-metode algoritma lainnya, seperti algoritma Dijkstra atau algoritma A*, tidak akan dipelajari secara mendalam dalam penelitian ini.
3. Pertimbangan Jalan: Penelitian ini memperhitungkan data jalan dan jarak antara simpul-simpul jaringan jalan. Faktor-faktor seperti kondisi lalu lintas saat ini, rambu-rambu, atau kecepatan rata-rata tidak diikutsertakan dalam analisis.
4. Waktu Tempuh Statis: Penelitian ini mengasumsikan bahwa waktu tempuh antara simpul-simpul jaringan jalan bersifat statis dan tidak berubah seiring waktu. Variabilitas waktu tempuh akibat perubahan lalu lintas tidak dipertimbangkan.
5. Kompleksitas Jaringan Jalan: Meskipun penelitian ini dapat diterapkan pada jaringan jalan yang lebih luas, skala jaringan jalan yang rumit dan besar mungkin menghasilkan waktu komputasi yang lama dan kurang efisien.
6. Tingkat Akurasi: Penelitian ini mengasumsikan bahwa data mengenai jaringan jalan dan jarak memiliki tingkat akurasi yang memadai. Faktor seperti jalan buntu atau jalan yang tidak dapat dilalui mungkin tidak dipertimbangkan.
7. Konteks Geografis: Penelitian ini mengasumsikan bahwa kondisi geografis dan topografi tetap konstan. Faktor-faktor seperti cuaca buruk atau perubahan topografi tidak dipertimbangkan.
8. Aspek Kemanfaatan: Penelitian ini tidak akan melakukan analisis mendalam mengenai dampak sosial atau lingkungan dari solusi rute terpendek yang dihasilkan.

Batasan-batasan di atas diambil untuk memastikan penelitian tetap terfokus pada penentuan rute terpendek dengan menggunakan metode algoritma *brute force* dalam konteks spesifik yang dijelaskan.

2. METODOLOGI PENELITIAN

Metodologi penelitian ini diarahkan untuk menerapkan metode algoritma *brute force* dalam menentukan rute terpendek dari Rumah Alief di Lembang ke Kampus Universitas Komputer Indonesia (Unikom) yang terletak di Dipatiukur. Tahapan metodologi penelitian ini dapat diuraikan sebagai berikut:

1. Pengumpulan Data:
   Data jalan, jarak antar simpul jaringan jalan, serta informasi topografi dan geografis terkait akan dikumpulkan dari sumber-sumber yang sah, seperti peta digital dan sumber data terkait.

2. Analisis Jaringan Jalan:
   Data yang telah dikumpulkan akan dianalisis untuk membangun representasi jaringan jalan antara Rumah Alief dan Kampus Unikom. Ini mencakup pengaturan simpul sebagai titik awal dan tujuan serta penghubung jalan antara mereka.

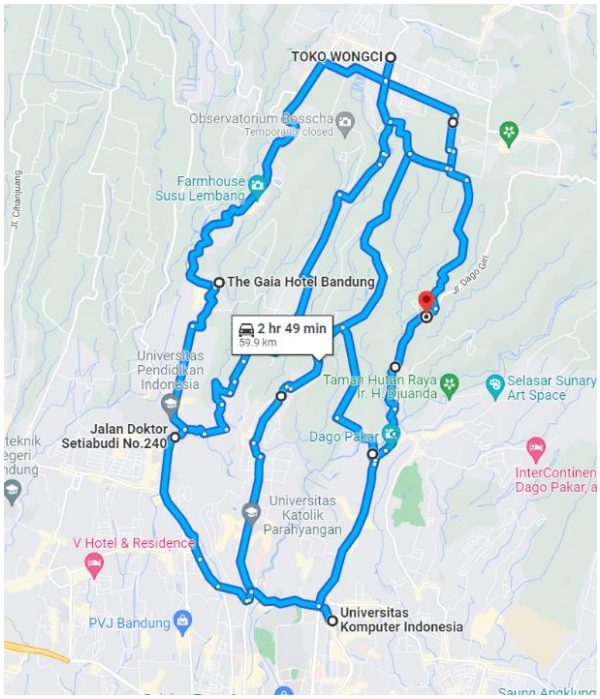
3. Penerapan Algoritma Brute Force:
   Metode algoritma *brute force* akan diimplementasikan untuk menghitung total jarak dari setiap rute yang mungkin dilalui antara Rumah Alief dan Kampus Unikom. Proses ini akan menjalankan perulangan untuk mengevaluasi seluruh kemungkinan solusi.

4. Evaluasi Solusi:
   Hasil dari algoritma *brute force* akan dievaluasi untuk menentukan rute terpendek berdasarkan total jarak yang dihitung. Solusi ini akan dibandingkan dengan rute alternatif, jika ada, untuk mengukur akurasi dan efektivitasnya.

5. Analisis Kinerja:
   Kinerja metode *brute force* akan dievaluasi berdasarkan waktu komputasi yang diperlukan untuk mencari rute terpendek. Efisiensi waktu akan dianalisis terutama dalam konteks jaringan jalan yang kompleks.

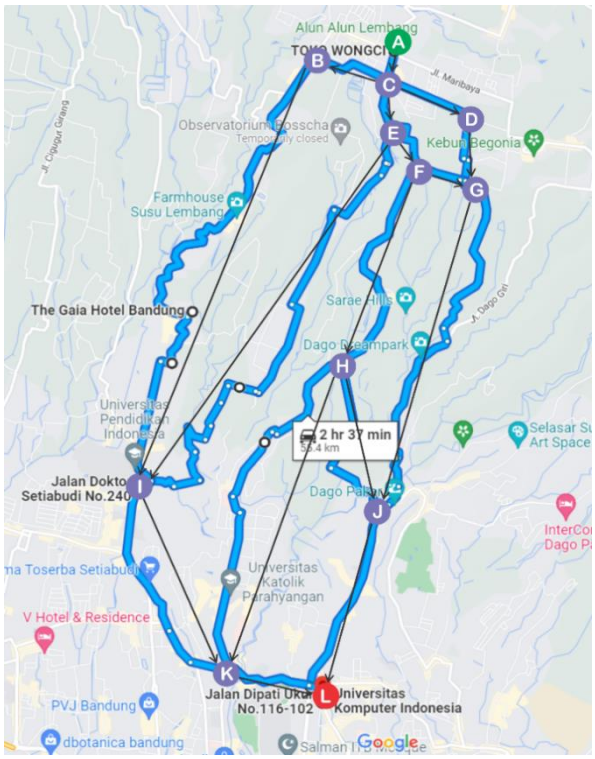
6. Diskusi dan Kesimpulan:
   Hasil analisis akan didiskusikan untuk mengidentifikasi potensi dan keterbatasan dari penerapan metode *brute force* dalam penentuan rute terpendek. Kesimpulan akan diambil untuk mengevaluasi apakah metode ini sesuai dan efektif dalam kasus ini.

Metodologi penelitian ini akan menghasilkan wawasan tentang kemampuan dan batasan metode *brute force* dalam menentukan rute terpendek. Evaluasi ini akan menjadi dasar untuk membahas keunggulan dan tantangan metode ini dalam konteks penentuan rute dalam mobilitas sehari-hari.Visual seluruh rute tersedia dari dari Lembang(Rumah Alief) menuju Kampus (Unikom) diambil dari Google Maps sebagai gambar untuk menentukan titik untuk setiap node rute yang akan dipilih.



Gambar 1. Seluruh rute dari Lembang ke Unikom

Menentukan semua titik node yang ada disetiap persimpangan berdasarkan visual Gambar 1



Gambar 2. Node

Dari data jarak tempuh yang didapatkan dari google maps maka berikut adalah tabel data yang didapatkan dan contoh perhitungan manual dengan algoritma Brute Force

Berikut penjelasan untuk setiap Node yang terdapat pada Gambar 3 :
A-C = Jl. Panorama
C-B = Jl. Grand Hotel
B-I = Jl. Raya Lembang
I-K = Jl. Dr. Setiabudi
K-L = Jl. Siliwangi
C-E = Jl. Punclut
E-I = Jl. Cijengkol
E-F = Jl. Punclut
F-G = Jl. Pager Wangi
F-H = Jl. Punclut
H-K = Jl. Punclut
H-J = Jl. Citra Green
C-D = Jl. Kayu Ambon
D-G = Jl. Buka Nagara
G-J = Jl. Dago Giri
J-L = Jl. Ir. H. Djuanda

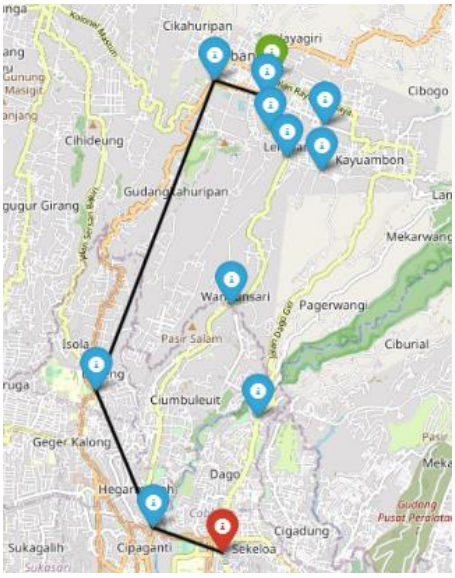
Jarak (KM) untuk setiap node ke node yang diketahui adalah sebagai berikut :
A-C = 0.33
C-B = 0.94
B-I = 5.60
I-K = 2.52
K-L = 1.23
C-E = 0.58
E-I = 5.28
E-F = 0.53
F-G = 0.62
F-H = 2.65
H-K = 4.01
H-J = 1.96
C-D = 1.09
D-G = 0.78
G-J = 4.30
J-L = 2.36

3. HASIL PENELITIAN DAN PEMBAHASAN

A. Percobaan pencarian rute terdekat

Berikut adalah tampilan interface aplikasi pencarian rute terpendek yang berjalan pada sistem operasi pada komputer.

1. Percobaan Pertama



Gambar 3.1 Visual Map dengan Node Rute Percobaan Pertama

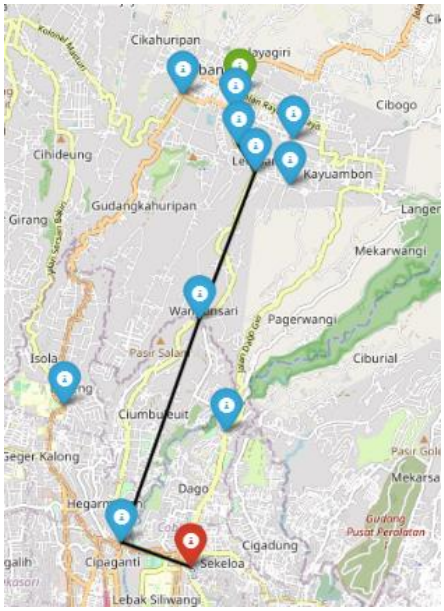
Jarak dari Node A ke Node C: 0.33 km - Jl. Panorama
Jarak dari Node C ke Node B: 0.94 km - Jl. Grand Hotel
Jarak dari Node B ke Node I: 5.60 km - Jl. Raya Lembang
Jarak dari Node I ke Node K: 2.52 km - Jl. Dr. Setiabudi
Jarak dari Node K ke Node L: 1.23 km - Jl. Siliwangi
Total jarak rute: 10.63 km

2. Percobaan Kedua

Gambar 3.2 Visual Map dengan Node Rute Percobaan Kedua

Jarak dari Node A ke Node C: 0.33 km - Jl. Panorama
Jarak dari Node C ke Node E: 0.58 km - Jl. Punclut
Jarak dari Node E ke Node I: 5.28 km - Jl. Cijengkol
Jarak dari Node I ke Node K: 2.52 km - Jl. Dr. Setiabudi
Jarak dari Node K ke Node L: 1.23 km - Jl. Siliwangi
Total jarak rute: 9.95 km

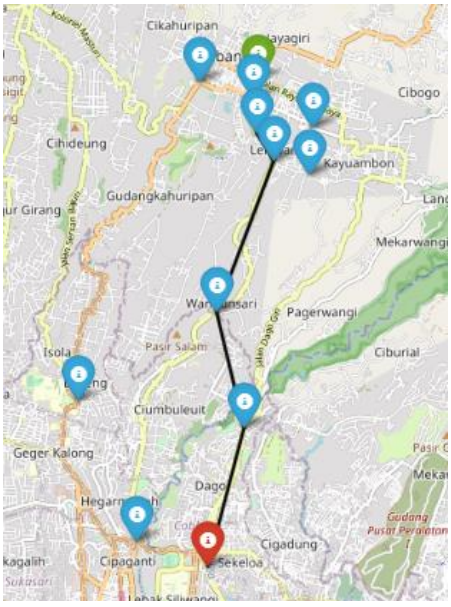
3. Percobaan Ketiga



Gambar 3.3 Visual Map dengan Node Rute Percobaan Ketiga

Jarak dari Node A ke Node C: 0.33 km - Jl. Panorama
Jarak dari Node C ke Node E: 0.58 km - Jl. Punclut
Jarak dari Node E ke Node F: 0.53 km - Jl. Punclut
Jarak dari Node F ke Node H: 2.65 km - Jl. Punclut
Jarak dari Node H ke Node K: 4.01 km - Jl. Punclut
Jarak dari Node K ke Node L: 1.23 km - Jl. Siliwangi
Total jarak rute: 9.33 km

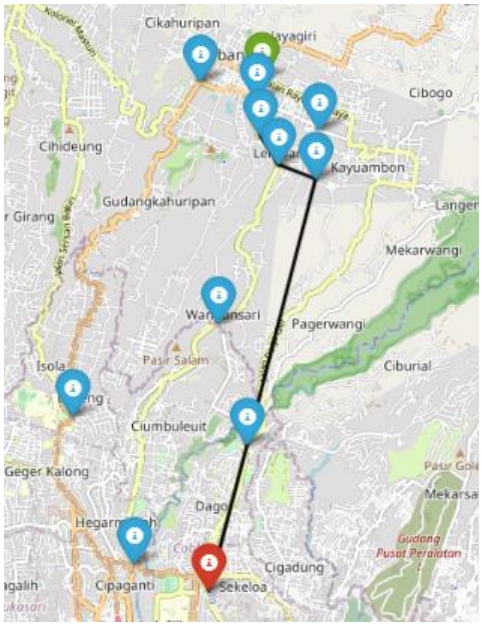
4. Percobaan Keempat



Gambar 3.4 Visual Map dengan Node Rute Percobaan Keempat

Jarak dari Node A ke Node C: 0.33 km - Jl. Panorama
Jarak dari Node C ke Node E: 0.58 km - Jl. Punclut
Jarak dari Node E ke Node F: 0.53 km - Jl. Punclut
Jarak dari Node F ke Node H: 2.65 km - Jl. Punclut
Jarak dari Node H ke Node J: 1.96 km - Jl. Citra Green
Jarak dari Node J ke Node L: 2.36 km - Jl. Ir. H. Djuanda
Total jarak rute: 8.41 km

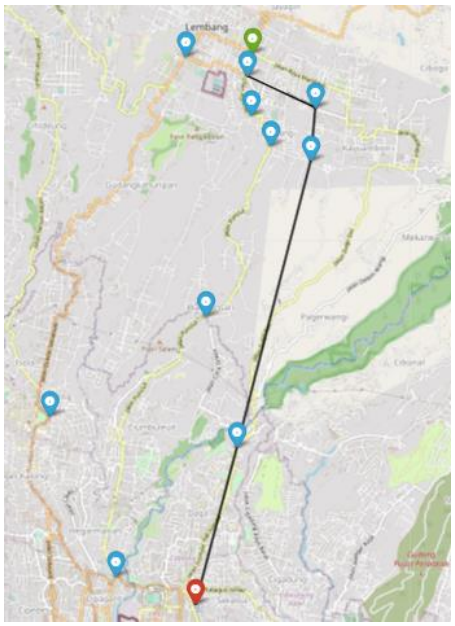
5. Percobaan Keempat



Gambar 3.5 Visual Map dengan Node Rute Percobaan Kelima

Jarak dari Node A ke Node C: 0.33 km - Jl. Panorama
Jarak dari Node C ke Node E: 0.58 km - Jl. Punclut
Jarak dari Node E ke Node F: 0.53 km - Jl. Punclut
Jarak dari Node F ke Node G: 0.62 km - Jl. Pager Wangi
Jarak dari Node G ke Node J: 4.30 km - Jl. Dago Giri
Jarak dari Node J ke Node L: 2.36 km - Jl. Ir. H. Djuanda
Total jarak rute: 8.73 km

6. Percobaan Keenam

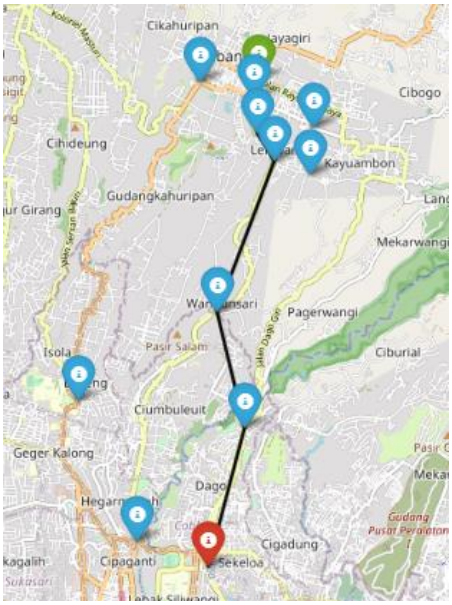


Gambar 3.6 Visual Map dengan Node Rute Percobaan Keenam

Jarak dari Node A ke Node C: 0.33 km - Jl. Panorama
Jarak dari Node C ke Node D: 1.09 km - Jl. Kayu Ambon
Jarak dari Node D ke Node G: 0.78 km - Jl. Buka Nagara
Jarak dari Node G ke Node J: 4.30 km - Jl. Dago Giri
Jarak dari Node J ke Node L: 2.36 km - Jl. Ir. H. Djuanda
Total jarak rute: 8.87 km

B. Hasil dari pencarian rute terdekat

Berikut ini adalah hasil dari pencarian rute terpendek yang didapat dari percobaan keempat :



Gambar 4 Visual Map dengan Node Rute Terpendek

Jalur terdekat:
A -> C -> E -> F -> H -> J -> L
Nama jalan yang dilalui:
1. Jl. Panorama
2. Jl. Punclut
3. Jl. Punclut
4. Jl. Punclut
5. Jl. Citra Green
6. Jl. Ir. H. Djuanda

Total jarak: 8.41 kilometer

Jarak tempuh sebanyak 8.41 KM merupakan jarak terpendek dari Rumah Alief (Lembang) ke Kampus UNIKOM (Dipatiukur) yang harus dilalui.

4. KESIMPULAN

Sebagai kesimpulan berdasarkan hasil dan pembahasan yang telah dijelaskan sebelumnya, dapat disimpulkan bahwa dalam penelitian ini telah berhasil membuat sebuah aplikasi pencarian rute terpendek menggunakan algoritma brute force. Dengan aplikasi ini, pengguna dapat mencari rute terpendek dari suatu titik ke titik yang lain. Aplikasi ini mengimplementasikan algoritma brute force dalam mencari rute terpendek.

Dalam proses pembuatan aplikasi ini, penulis telah melakukan beberapa tahap mulai dari pengumpulan data, analisis data, hingga implementasi algoritma brute force pada aplikasi. Penggunaan pendekatan kualitatif dengan metode deskriptif menjadi metode penelitian yang digunakan untuk mencapai tujuan dari penelitian ini.

Dalam proses analisis data, penulis menggunakan tabel data dan perhitungan manual dengan algoritma brute force untuk mencari rute terpendek dari suatu titik ke titik yang lain. Selanjutnya, penulis mengimplementasikan algoritma brute force dalam aplikasi dan melakukan uji coba untuk menguji keakuratan dan kevalidan aplikasi.

Dari hasil uji coba, aplikasi ini telah berhasil bekerja dengan baik dan menghasilkan rute terpendek dengan akurat. Aplikasi ini dapat membantu pengguna dalam mencari rute terpendek dari suatu titik ke titik yang lain dengan cepat dan mudah.

Namun, ada beberapa kelemahan pada aplikasi ini. Salah satunya adalah keterbatasan data yang digunakan dalam penelitian. Selain itu, penulis hanya mengimplementasikan algoritma brute force dalam aplikasi ini, sehingga masih terdapat ruang untuk mengembangkan aplikasi dengan mengimplementasikan algoritma lain yang lebih kompleks dan akurat.

Dengan demikian, diharapkan penelitian ini dapat memberikan kontribusi dan menjadi referensi bagi peneliti lainnya yang tertarik dalam mengembangkan aplikasi pencarian rute terpendek menggunakan algoritma brute force.

5. DAFTAR PUSTAKA

Perbandingan Algoritma Brute-Force dan Algoritma A* untuk Mencari Rute Terpendek Antar Klinik Kecantikan di Kota Medan (usu.ac.id)

PENERAPAN ALGORITMA BRUTE FORCE UNTUK PENCARIAN JALUR ALTERNATIF DI KOTA BANDUNG - Repository (unikom.ac.id)

PENERAPAN ALGORITMA HEURISTIK UNTUK MENENTUKAN RUTE TERPENDEK PENDISTRIBUSIAN MINUMAN RINGAN (SOFTDRINK) PADA PT. MEDAN SUMBER ALAS SEMESTA MEDAN - Digital Repository Universitas Negeri Medan (unimed.ac.id)

https://stackoverflow.com/search?q=brute+force&s=0aeef8eb-fe21-49d1-b8c1-f7c973b39cd8

https://ichi.pro/id/visualisasi-peta-interaktif-dengan-folium-di-python-51218899373467

6. PROGRAM

a) Menampilkan Peta, Rute Keseluruhan, Node, dan Garis Busur

```python
import folium

# definisikan koordinat node
A = -6.81458,107.62306
B = -6.81495,107.61428
C = -6.81752,107.62242
D = -6.82171,107.63139
E = -6.82271,107.62295
F = -6.82675,107.62552
G = -6.82872,107.63081
H = -6.84915,107.61691
I = -6.86230,107.59622
J = -6.86638,107.62102
K = -6.88335,107.60498
L = -6.88701,107.61552

# buat objek peta
peta = folium.Map(location=A, zoom_start=12)

# tambahkan marker untuk setiap node
nodes = [A, B, C, D, E, F, G, H, I, J, K, L]
node_names = ['Node A',
              'Node B',
              'Node C',
              'Node D',
              'Node E',
              'Node F',
              'Node G',
              'Node H',
              'Node I',
              'Node J',
              'Node K',
              'Node L']

for i in range(len(nodes)):
    if i == 0: # node A
        folium.Marker(location=nodes[i], tooltip=node_names[i], icon=folium.Icon(color='green')).add_to(peta)
    elif i == len(nodes)-1: # node L
        folium.Marker(location=nodes[i], tooltip=node_names[i], icon=folium.Icon(color='red')).add_to(peta)
    elif i > 0 and i < 11: # node B-K
        folium.Marker(location=nodes[i], tooltip=node_names[i], icon=folium.Icon(color='blue')).add_to(peta)
    else:
        folium.Marker(location=nodes[i], tooltip=node_names[i]).add_to(peta)

# hubungkan setiap node dengan jalur
# definisikan urutan node dalam jalur
jalur = [A, C, B, I, K, L, J, G, D, C, E, F, H, K, I, E, F, G, J, H]

# tambahkan jalur ke peta
folium.PolyLine(locations=jalur, color='black').add_to(peta)

# tampilkan peta
peta
```

b) Menampilkan Node, Jarak Antar Node Beserta Nama Jalannya

```python
from geopy.distance import geodesic
import pandas as pd

# Koordinat
locations = {
    'A': (-6.81458, 107.62306),
    'B': (-6.81495, 107.61428),
    'C': (-6.81752, 107.62242),
    'D': (-6.82171, 107.63139),
    'E': (-6.82271, 107.62295),
    'F': (-6.82675, 107.62552),
    'G': (-6.82872, 107.63081),
    'H': (-6.84915, 107.61691),
    'I': (-6.86230, 107.59622),
    'J': (-6.86638, 107.62102),
    'K': (-6.88335, 107.60498),
    'L': (-6.88701, 107.61552)
}

# Definisi jalan antar titik
routes = [
    ('A', 'C', 'Jl. Panorama'),
    ('C', 'B', 'Jl. Grand Hotel'),
    ('B', 'I', 'Jl. Raya Lembang'),
    ('I', 'K', 'Jl. Dr. Setiabudi'),
    ('K', 'L', 'Jl. Siliwangi'),
    ('C', 'E', 'Jl. Punclut'),
    ('E', 'I', 'Jl. Cijengkol'),
    ('E', 'F', 'Jl. Punclut'),
    ('F', 'G', 'Jl. Pager Wangi'),
    ('F', 'H', 'Jl. Punclut'),
    ('H', 'K', 'Jl. Punclut'),
    ('H', 'J', 'Jl. Citra Green'),
    ('C', 'D', 'Jl. Kayu Ambon'),
    ('D', 'G', 'Jl. Buka Nagara'),
    ('G', 'J', 'Jl. Dago Giri'),
    ('J', 'L', 'Jl. Ir. H. Djuanda')
]

# Hitung jarak dan buat DataFrame
data = []
for route in routes:
    start, end, road_name = route
    distance = geodesic(locations[start], locations[end]).km
    data.append((f'{start}-{end}', distance, road_name))

df = pd.DataFrame(data, columns=['Node', 'Jarak (km)', 'Nama Jalan'])

# Memformat kolom 'Jarak (km)' menjadi 2 angka di belakang koma
df['Jarak (km)'] = df['Jarak (km)'].round(2)

# Tampilkan tabel
df
```

c) Percobaan Rute Ke-1

```python
import folium
from geopy.distance import geodesic
import pandas as pd

# Definisikan koordinat node
nodes = {
    'A': (-6.81458, 107.62306),
    'B': (-6.81495, 107.61428),
    'C': (-6.81752, 107.62242),
    'D': (-6.82171, 107.63139),
    'E': (-6.82271, 107.62295),
    'F': (-6.82675, 107.62552),
    'G': (-6.82872, 107.63081),
    'H': (-6.84915, 107.61691),
    'I': (-6.86230, 107.59622),
    'J': (-6.86638, 107.62102),
    'K': (-6.88335, 107.60498),
    'L': (-6.88701, 107.61552)
}

# Buat objek peta
peta = folium.Map(location=nodes['A'], zoom_start=12)

# Tambahkan marker untuk setiap node
for node, coordinates in nodes.items():
    icon_color = 'green' if node == 'A' else ('red' if node == 'L' else 'blue')
    folium.Marker(location=coordinates, tooltip=f'Node {node}', icon=folium.Icon(color=icon_color)).add_to(peta)

# Hubungkan setiap node dengan jalur
jalur = ['A', 'C', 'B', 'I', 'K', 'L']

# Konversi jalur ke koordinat
jalur_coordinates = [nodes[node] for node in jalur]

# Tambahkan jalur ke peta
for i in range(len(jalur) - 1):
    folium.PolyLine(locations=[jalur_coordinates[i], jalur_coordinates[i + 1]], color='black').add_to(peta)

# Tampilkan peta
peta
```

d) Menampilkan Jarak Antar Node dan Nama Jalan

```python
from geopy.distance import geodesic

# Definisikan urutan jalur
jalur = ['A', 'C', 'B', 'I', 'K', 'L']

# Definisikan koordinat node
nodes = {
    'A': (-6.81458, 107.62306),
    'B': (-6.81495, 107.61428),
    'C': (-6.81752, 107.62242),
    'D': (-6.82171, 107.63139),
    'E': (-6.82271, 107.62295),
    'F': (-6.82675, 107.62552),
    'G': (-6.82872, 107.63081),
    'H': (-6.84915, 107.61691),
    'I': (-6.86230, 107.59622),
    'J': (-6.86638, 107.62102),
    'K': (-6.88335, 107.60498),
    'L': (-6.88701, 107.61552)
}

# Definisikan kamus nama jalan berdasarkan node awal dan akhir
nama_jalan_map = {
    ('A', 'C'): 'Jl. Panorama',
    ('C', 'B'): 'Jl. Grand Hotel',
    ('B', 'I'): 'Jl. Raya Lembang',
    ('I', 'K'): 'Jl. Dr. Setiabudi',
    ('K', 'L'): 'Jl. Siliwangi',
    ('C', 'E'): 'Jl. Punclut',
    ('E', 'I'): 'Jl. Cijengkol',
    ('E', 'F'): 'Jl. Punclut',
    ('F', 'G'): 'Jl. Pager Wangi',
    ('F', 'H'): 'Jl. Punclut',
    ('H', 'K'): 'Jl. Punclut',
    ('H', 'J'): 'Jl. Citra Green',
    ('C', 'D'): 'Jl. Kayu Ambon',
    ('D', 'G'): 'Jl. Buka Nagara',
    ('G', 'J'): 'Jl. Dago Giri',
    ('J', 'L'): 'Jl. Ir. H. Djuanda'
}

# Tampilkan jarak antara setiap node dan nama jalan
total_distance = 0
for i in range(len(jalur) - 1):
    start = jalur[i]
    end = jalur[i + 1]

    if start in nodes and end in nodes:
        distance = geodesic(nodes[start], nodes[end]).kilometers
        total_distance += distance
        jalan = nama_jalan_map.get((start, end), f"Jl. {start}-{end}")
        print(f"Jarak dari Node {start} ke Node {end}: {distance:.2f} km - {jalan}")
    else:
        print(f"Tidak dapat menghitung jarak dari Node {start} ke Node {end}: Koordinat tidak tersedia")

# Tampilkan total jarak rute
print(f"Total jarak rute: {total_distance:.2f} km")
```

e) Percobaan Rute Ke-2

```python
import folium
from geopy.distance import geodesic
import pandas as pd

# Definisikan koordinat node
nodes = {
    'A': (-6.81458, 107.62306),
    'B': (-6.81495, 107.61428),
    'C': (-6.81752, 107.62242),
    'D': (-6.82171, 107.63139),
    'E': (-6.82271, 107.62295),
    'F': (-6.82675, 107.62552),
    'G': (-6.82872, 107.63081),
    'H': (-6.84915, 107.61691),
    'I': (-6.86230, 107.59622),
    'J': (-6.86638, 107.62102),
    'K': (-6.88335, 107.60498),
    'L': (-6.88701, 107.61552)
}

# Buat objek peta
peta = folium.Map(location=nodes['A'], zoom_start=12)

# Tambahkan marker untuk setiap node
for node, coordinates in nodes.items():
    icon_color = 'green' if node == 'A' else ('red' if node == 'L' else 'blue')
    folium.Marker(location=coordinates, tooltip=f'Node {node}', icon=folium.Icon(color=icon_color)).add_to(peta)

# Hubungkan setiap node dengan jalur
jalur = ['A', 'C', 'E', 'I', 'K', 'L']

# Konversi jalur ke koordinat
jalur_coordinates = [nodes[node] for node in jalur]

# Tambahkan jalur ke peta
for i in range(len(jalur) - 1):
    folium.PolyLine(locations=[jalur_coordinates[i], jalur_coordinates[i + 1]], color='black').add_to(peta)

# Tampilkan peta
peta
```

f) Menampilkan Jarak Antar Node dan Nama Jalan (2)

```python
from geopy.distance import geodesic

# Definisikan urutan jalur
jalur = ['A', 'C', 'E', 'I', 'K', 'L']

# Definisikan koordinat node
nodes = {
    'A': (-6.81458, 107.62306),
    'B': (-6.81495, 107.61428),
    'C': (-6.81752, 107.62242),
    'D': (-6.82171, 107.63139),
    'E': (-6.82271, 107.62295),
    'F': (-6.82675, 107.62552),
    'G': (-6.82872, 107.63081),
    'H': (-6.84915, 107.61691),
    'I': (-6.86230, 107.59622),
    'J': (-6.86638, 107.62102),
    'K': (-6.88335, 107.60498),
    'L': (-6.88701, 107.61552)
}

# Definisikan kamus nama jalan berdasarkan node awal dan akhir
nama_jalan_map = {
    ('A', 'C'): 'Jl. Panorama',
    ('C', 'B'): 'Jl. Grand Hotel',
    ('B', 'I'): 'Jl. Raya Lembang',
    ('I', 'K'): 'Jl. Dr. Setiabudi',
    ('K', 'L'): 'Jl. Siliwangi',
    ('C', 'E'): 'Jl. Punclut',
    ('E', 'I'): 'Jl. Cijengkol',
    ('E', 'F'): 'Jl. Punclut',
    ('F', 'G'): 'Jl. Pager Wangi',
    ('F', 'H'): 'Jl. Punclut',
    ('H', 'K'): 'Jl. Punclut',
    ('H', 'J'): 'Jl. Citra Green',
    ('C', 'D'): 'Jl. Kayu Ambon',
    ('D', 'G'): 'Jl. Buka Nagara',
    ('G', 'J'): 'Jl. Dago Giri',
    ('J', 'L'): 'Jl. Ir. H. Djuanda'
}

# Tampilkan jarak antara setiap node dan nama jalan
total_distance = 0
for i in range(len(jalur) - 1):
    start = jalur[i]
    end = jalur[i + 1]

    if start in nodes and end in nodes:
        distance = geodesic(nodes[start], nodes[end]).kilometers
        total_distance += distance
        jalan = nama_jalan_map.get((start, end), f"Jl. {start}-{end}")
        print(f"Jarak dari Node {start} ke Node {end}: {distance:.2f} km - {jalan}")
    else:
        print(f"Tidak dapat menghitung jarak dari Node {start} ke Node {end}: Koordinat tidak tersedia")

# Tampilkan total jarak rute
print(f"Total jarak rute: {total_distance:.2f} km")
```

g) Percobaan Rute Ke-3

```python
import folium
from geopy.distance import geodesic
import pandas as pd

# Definisikan koordinat node
nodes = {
    'A': (-6.81458, 107.62306),
    'B': (-6.81495, 107.61428),
    'C': (-6.81752, 107.62242),
    'D': (-6.82171, 107.63139),
    'E': (-6.82271, 107.62295),
    'F': (-6.82675, 107.62552),
    'G': (-6.82872, 107.63081),
    'H': (-6.84915, 107.61691),
    'I': (-6.86230, 107.59622),
    'J': (-6.86638, 107.62102),
    'K': (-6.88335, 107.60498),
    'L': (-6.88701, 107.61552)
}

# Buat objek peta
peta = folium.Map(location=nodes['A'], zoom_start=12)

# Tambahkan marker untuk setiap node
for node, coordinates in nodes.items():
    icon_color = 'green' if node == 'A' else ('red' if node == 'L' else 'blue')
    folium.Marker(location=coordinates, tooltip=f'Node {node}', icon=folium.Icon(color=icon_color)).add_to(peta)

# Hubungkan setiap node dengan jalur
jalur = ['A', 'C', 'E', 'F', 'H', 'K', 'L']

# Konversi jalur ke koordinat
jalur_coordinates = [nodes[node] for node in jalur]

# Tambahkan jalur ke peta
for i in range(len(jalur) - 1):
    folium.PolyLine(locations=[jalur_coordinates[i], jalur_coordinates[i + 1]], color='black').add_to(peta)

# Tampilkan peta
peta
```

h) Menampilkan Jarak Antar Node dan Nama Jalan (3)

```python
from geopy.distance import geodesic

# Definisikan urutan jalur
jalur = ['A', 'C', 'E', 'F', 'H', 'K', 'L']

# Definisikan koordinat node
nodes = {
    'A': (-6.81458, 107.62306),
    'B': (-6.81495, 107.61428),
    'C': (-6.81752, 107.62242),
    'D': (-6.82171, 107.63139),
    'E': (-6.82271, 107.62295),
    'F': (-6.82675, 107.62552),
    'G': (-6.82872, 107.63081),
    'H': (-6.84915, 107.61691),
    'I': (-6.86230, 107.59622),
    'J': (-6.86638, 107.62102),
    'K': (-6.88335, 107.60498),
    'L': (-6.88701, 107.61552)
}

# Definisikan kamus nama jalan berdasarkan node awal dan akhir
nama_jalan_map = {
    ('A', 'C'): 'Jl. Panorama',
    ('C', 'B'): 'Jl. Grand Hotel',
    ('B', 'I'): 'Jl. Raya Lembang',
    ('I', 'K'): 'Jl. Dr. Setiabudi',
    ('K', 'L'): 'Jl. Siliwangi',
    ('C', 'E'): 'Jl. Punclut',
    ('E', 'I'): 'Jl. Cijengkol',
    ('E', 'F'): 'Jl. Punclut',
    ('F', 'G'): 'Jl. Pager Wangi',
    ('F', 'H'): 'Jl. Punclut',
    ('H', 'K'): 'Jl. Punclut',
    ('H', 'J'): 'Jl. Citra Green',
    ('C', 'D'): 'Jl. Kayu Ambon',
    ('D', 'G'): 'Jl. Buka Nagara',
    ('G', 'J'): 'Jl. Dago Giri',
    ('J', 'L'): 'Jl. Ir. H. Djuanda'
}

# Tampilkan jarak antara setiap node dan nama jalan
total_distance = 0
for i in range(len(jalur) - 1):
    start = jalur[i]
    end = jalur[i + 1]

    if start in nodes and end in nodes:
        distance = geodesic(nodes[start], nodes[end]).kilometers
        total_distance += distance
        jalan = nama_jalan_map.get((start, end), f"Jl. {start}-{end}")
        print(f"Jarak dari Node {start} ke Node {end}: {distance:.2f} km - {jalan}")
    else:
        print(f"Tidak dapat menghitung jarak dari Node {start} ke Node {end}: Koordinat tidak tersedia")

# Tampilkan total jarak rute
print(f"Total jarak rute: {total_distance:.2f} km")
```

i) Percobaan Rute Ke-4

```python
import folium
from geopy.distance import geodesic
import pandas as pd

# Definisikan koordinat node
nodes = {
    'A': (-6.81458, 107.62306),
    'B': (-6.81495, 107.61428),
    'C': (-6.81752, 107.62242),
    'D': (-6.82171, 107.63139),
    'E': (-6.82271, 107.62295),
    'F': (-6.82675, 107.62552),
    'G': (-6.82872, 107.63081),
    'H': (-6.84915, 107.61691),
    'I': (-6.86230, 107.59622),
    'J': (-6.86638, 107.62102),
    'K': (-6.88335, 107.60498),
    'L': (-6.88701, 107.61552)
}

# Buat objek peta
peta = folium.Map(location=nodes['A'], zoom_start=12)

# Tambahkan marker untuk setiap node
for node, coordinates in nodes.items():
    icon_color = 'green' if node == 'A' else ('red' if node == 'L' else 'blue')
    folium.Marker(location=coordinates, tooltip=f'Node {node}', icon=folium.Icon(color=icon_color)).add_to(peta)

# Hubungkan setiap node dengan jalur
jalur = ['A', 'C', 'E', 'F', 'H', 'J', 'L']

# Konversi jalur ke koordinat
jalur_coordinates = [nodes[node] for node in jalur]

# Tambahkan jalur ke peta
for i in range(len(jalur) - 1):
    folium.PolyLine(locations=[jalur_coordinates[i], jalur_coordinates[i + 1]], color='black').add_to(peta)

# Tampilkan peta
peta
```

j) Menampilkan Jarak Antar Node dan Nama Jalan (4)

```python
from geopy.distance import geodesic

# Definisikan urutan jalur
jalur = ['A', 'C', 'E', 'F', 'H', 'J', 'L']

# Definisikan koordinat node
nodes = {
    'A': (-6.81458, 107.62306),
    'B': (-6.81495, 107.61428),
    'C': (-6.81752, 107.62242),
    'D': (-6.82171, 107.63139),
    'E': (-6.82271, 107.62295),
    'F': (-6.82675, 107.62552),
    'G': (-6.82872, 107.63081),
    'H': (-6.84915, 107.61691),
    'I': (-6.86230, 107.59622),
    'J': (-6.86638, 107.62102),
    'K': (-6.88335, 107.60498),
    'L': (-6.88701, 107.61552)
}

# Definisikan kamus nama jalan berdasarkan node awal dan akhir
nama_jalan_map = {
    ('A', 'C'): 'Jl. Panorama',
    ('C', 'B'): 'Jl. Grand Hotel',
    ('B', 'I'): 'Jl. Raya Lembang',
    ('I', 'K'): 'Jl. Dr. Setiabudi',
    ('K', 'L'): 'Jl. Siliwangi',
    ('C', 'E'): 'Jl. Punclut',
    ('E', 'I'): 'Jl. Cijengkol',
    ('E', 'F'): 'Jl. Punclut',
    ('F', 'G'): 'Jl. Pager Wangi',
    ('F', 'H'): 'Jl. Punclut',
    ('H', 'K'): 'Jl. Punclut',
    ('H', 'J'): 'Jl. Citra Green',
    ('C', 'D'): 'Jl. Kayu Ambon',
    ('D', 'G'): 'Jl. Buka Nagara',
    ('G', 'J'): 'Jl. Dago Giri',
    ('J', 'L'): 'Jl. Ir. H. Djuanda'
}

# Tampilkan jarak antara setiap node dan nama jalan
total_distance = 0
for i in range(len(jalur) - 1):
    start = jalur[i]
    end = jalur[i + 1]

    if start in nodes and end in nodes:
        distance = geodesic(nodes[start], nodes[end]).kilometers
        total_distance += distance
        jalan = nama_jalan_map.get((start, end), f"Jl. {start}-{end}")
        print(f"Jarak dari Node {start} ke Node {end}: {distance:.2f} km - {jalan}")
    else:
        print(f"Tidak dapat menghitung jarak dari Node {start} ke Node {end}: Koordinat tidak tersedia")

# Tampilkan total jarak rute
print(f"Total jarak rute: {total_distance:.2f} km")
```

k) Percobaan Rute Ke-5

```python
import folium
from geopy.distance import geodesic
import pandas as pd

# Definisikan koordinat node
nodes = {
    'A': (-6.81458, 107.62306),
    'B': (-6.81495, 107.61428),
    'C': (-6.81752, 107.62242),
    'D': (-6.82171, 107.63139),
    'E': (-6.82271, 107.62295),
    'F': (-6.82675, 107.62552),
    'G': (-6.82872, 107.63081),
    'H': (-6.84915, 107.61691),
    'I': (-6.86230, 107.59622),
    'J': (-6.86638, 107.62102),
    'K': (-6.88335, 107.60498),
    'L': (-6.88701, 107.61552)
}

# Buat objek peta
peta = folium.Map(location=nodes['A'], zoom_start=12)

# Tambahkan marker untuk setiap node
for node, coordinates in nodes.items():
    icon_color = 'green' if node == 'A' else ('red' if node == 'L' else 'blue')
    folium.Marker(location=coordinates, tooltip=f'Node {node}', icon=folium.Icon(color=icon_color)).add_to(peta)

# Hubungkan setiap node dengan jalur
jalur = ['A', 'C', 'E', 'F', 'G', 'J', 'L']

# Konversi jalur ke koordinat
jalur_coordinates = [nodes[node] for node in jalur]

# Tambahkan jalur ke peta
for i in range(len(jalur) - 1):
    folium.PolyLine(locations=[jalur_coordinates[i], jalur_coordinates[i + 1]], color='black').add_to(peta)

# Tampilkan peta
peta
```

l) Menampilkan Jarak Antar Node dan Nama Jalan (5)

```python
from geopy.distance import geodesic

# Definisikan urutan jalur
jalur = ['A', 'C', 'E', 'F', 'G', 'J', 'L']

# Definisikan koordinat node
nodes = {
    'A': (-6.81458, 107.62306),
    'B': (-6.81495, 107.61428),
    'C': (-6.81752, 107.62242),
    'D': (-6.82171, 107.63139),
    'E': (-6.82271, 107.62295),
    'F': (-6.82675, 107.62552),
    'G': (-6.82872, 107.63081),
    'H': (-6.84915, 107.61691),
    'I': (-6.86230, 107.59622),
    'J': (-6.86638, 107.62102),
    'K': (-6.88335, 107.60498),
    'L': (-6.88701, 107.61552)
}

# Definisikan kamus nama jalan berdasarkan node awal dan akhir
nama_jalan_map = {
    ('A', 'C'): 'Jl. Panorama',
    ('C', 'B'): 'Jl. Grand Hotel',
    ('B', 'I'): 'Jl. Raya Lembang',
    ('I', 'K'): 'Jl. Dr. Setiabudi',
    ('K', 'L'): 'Jl. Siliwangi',
    ('C', 'E'): 'Jl. Punclut',
    ('E', 'I'): 'Jl. Cijengkol',
    ('E', 'F'): 'Jl. Punclut',
    ('F', 'G'): 'Jl. Pager Wangi',
    ('F', 'H'): 'Jl. Punclut',
    ('H', 'K'): 'Jl. Punclut',
    ('H', 'J'): 'Jl. Citra Green',
    ('C', 'D'): 'Jl. Kayu Ambon',
    ('D', 'G'): 'Jl. Buka Nagara',
    ('G', 'J'): 'Jl. Dago Giri',
    ('J', 'L'): 'Jl. Ir. H. Djuanda'
}

# Tampilkan jarak antara setiap node dan nama jalan
total_distance = 0
for i in range(len(jalur) - 1):
    start = jalur[i]
    end = jalur[i + 1]

    if start in nodes and end in nodes:
        distance = geodesic(nodes[start], nodes[end]).kilometers
        total_distance += distance
        jalan = nama_jalan_map.get((start, end), f"Jl. {start}-{end}")
        print(f"Jarak dari Node {start} ke Node {end}: {distance:.2f} km - {jalan}")
    else:
        print(f"Tidak dapat menghitung jarak dari Node {start} ke Node {end}: Koordinat tidak tersedia")

# Tampilkan total jarak rute
print(f"Total jarak rute: {total_distance:.2f} km")
```

m) Percobaan Rute Ke-6

```python
import folium
from geopy.distance import geodesic
import pandas as pd

# Definisikan koordinat node
nodes = {
    'A': (-6.81458, 107.62306),
    'B': (-6.81495, 107.61428),
    'C': (-6.81752, 107.62242),
    'D': (-6.82171, 107.63139),
    'E': (-6.82271, 107.62295),
    'F': (-6.82675, 107.62552),
    'G': (-6.82872, 107.63081),
    'H': (-6.84915, 107.61691),
    'I': (-6.86230, 107.59622),
    'J': (-6.86638, 107.62102),
    'K': (-6.88335, 107.60498),
    'L': (-6.88701, 107.61552)
}

# Buat objek peta
peta = folium.Map(location=nodes['A'], zoom_start=12)

# Tambahkan marker untuk setiap node
for node, coordinates in nodes.items():
    icon_color = 'green' if node == 'A' else ('red' if node == 'L' else 'blue')
    folium.Marker(location=coordinates, tooltip=f'Node {node}', icon=folium.Icon(color=icon_color)).add_to(peta)

# Hubungkan setiap node dengan jalur
jalur = ['A', 'C', 'D', 'G', 'J', 'L']

# Konversi jalur ke koordinat
jalur_coordinates = [nodes[node] for node in jalur]

# Tambahkan jalur ke peta
for i in range(len(jalur) - 1):
    folium.PolyLine(locations=[jalur_coordinates[i], jalur_coordinates[i + 1]], color='black').add_to(peta)

# Tampilkan peta
peta
```

n) Menampilkan Jarak Antar Node dan Nama Jalan (6)

```python
from geopy.distance import geodesic

# Definisikan urutan jalur
jalur = ['A', 'C', 'D', 'G', 'J', 'L']

# Definisikan koordinat node
nodes = {
    'A': (-6.81458, 107.62306),
    'B': (-6.81495, 107.61428),
    'C': (-6.81752, 107.62242),
    'D': (-6.82171, 107.63139),
    'E': (-6.82271, 107.62295),
    'F': (-6.82675, 107.62552),
    'G': (-6.82872, 107.63081),
    'H': (-6.84915, 107.61691),
    'I': (-6.86230, 107.59622),
    'J': (-6.86638, 107.62102),
    'K': (-6.88335, 107.60498),
    'L': (-6.88701, 107.61552)
}

# Definisikan kamus nama jalan berdasarkan node awal dan akhir
nama_jalan_map = {
    ('A', 'C'): 'Jl. Panorama',
    ('C', 'B'): 'Jl. Grand Hotel',
    ('B', 'I'): 'Jl. Raya Lembang',
    ('I', 'K'): 'Jl. Dr. Setiabudi',
    ('K', 'L'): 'Jl. Siliwangi',
    ('C', 'E'): 'Jl. Punclut',
    ('E', 'I'): 'Jl. Cijengkol',
    ('E', 'F'): 'Jl. Punclut',
    ('F', 'G'): 'Jl. Pager Wangi',
    ('F', 'H'): 'Jl. Punclut',
    ('H', 'K'): 'Jl. Punclut',
    ('H', 'J'): 'Jl. Citra Green',
    ('C', 'D'): 'Jl. Kayu Ambon',
    ('D', 'G'): 'Jl. Buka Nagara',
    ('G', 'J'): 'Jl. Dago Giri',
    ('J', 'L'): 'Jl. Ir. H. Djuanda'
}

# Tampilkan jarak antara setiap node dan nama jalan
total_distance = 0
for i in range(len(jalur) - 1):
    start = jalur[i]
    end = jalur[i + 1]

    if start in nodes and end in nodes:
        distance = geodesic(nodes[start], nodes[end]).kilometers
        total_distance += distance
        jalan = nama_jalan_map.get((start, end), f"Jl. {start}-{end}")
        print(f"Jarak dari Node {start} ke Node {end}: {distance:.2f} km - {jalan}")
    else:
        print(f"Tidak dapat menghitung jarak dari Node {start} ke Node {end}: Koordinat tidak tersedia")

# Tampilkan total jarak rute
print(f"Total jarak rute: {total_distance:.2f} km")
```

o) Rute Terpendek

```python
import folium
from geopy.distance import geodesic

# Definisikan urutan jalur
jalur_list = [
    ['A', 'C', 'B', 'I', 'K', 'L'],
    ['A', 'C', 'E', 'I', 'K', 'L'],
    ['A', 'C', 'E', 'F', 'H', 'K', 'L'],
    ['A', 'C', 'E', 'F', 'H', 'J', 'L'],
    ['A', 'C', 'E', 'F', 'G', 'J', 'L'],
    ['A', 'C', 'D', 'G', 'J', 'L']
]

# Definisikan koordinat node
nodes = {
    'A': (-6.81458, 107.62306),
    'B': (-6.81495, 107.61428),
    'C': (-6.81752, 107.62242),
    'D': (-6.82171, 107.63139),
    'E': (-6.82271, 107.62295),
    'F': (-6.82675, 107.62552),
    'G': (-6.82872, 107.63081),
    'H': (-6.84915, 107.61691),
    'I': (-6.86230, 107.59622),
    'J': (-6.86638, 107.62102),
    'K': (-6.88335, 107.60498),
    'L': (-6.88701, 107.61552)
}

# Definisikan kamus nama jalan berdasarkan node awal dan akhir
nama_jalan_map = {
    ('A', 'C'): 'Jl. Panorama',
    ('C', 'B'): 'Jl. Grand Hotel',
    ('B', 'I'): 'Jl. Raya Lembang',
    ('I', 'K'): 'Jl. Dr. Setiabudi',
    ('K', 'L'): 'Jl. Siliwangi',
    ('C', 'E'): 'Jl. Punclut',
    ('E', 'I'): 'Jl. Cijengkol',
    ('E', 'F'): 'Jl. Punclut',
    ('F', 'G'): 'Jl. Pager Wangi',
    ('F', 'H'): 'Jl. Punclut',
    ('H', 'K'): 'Jl. Punclut',
    ('H', 'J'): 'Jl. Citra Green',
    ('C', 'D'): 'Jl. Kayu Ambon',
    ('D', 'G'): 'Jl. Buka Nagara',
    ('G', 'J'): 'Jl. Dago Giri',
    ('J', 'L'): 'Jl. Ir. H. Djuanda'
}

import folium
from geopy.distance import geodesic

# ... (previous code for defining nodes, jalur_list, and nama_jalan_map)

# Fungsi untuk menghitung total jarak suatu jalur
def calculate_total_distance(jalur):
    total_distance = 0
    for i in range(len(jalur) - 1):
        start = jalur[i]
        end = jalur[i + 1]
        if start in nodes and end in nodes:
            distance = geodesic(nodes[start], nodes[end]).kilometers
            total_distance += distance
    return total_distance

# Cari jalur terpendek dan nilai terkecil
shortest_distance = float('inf')
shortest_path = []

for jalur in jalur_list:
    current_distance = calculate_total_distance(jalur)
    if current_distance < shortest_distance:
        shortest_distance = current_distance
        shortest_path = jalur

# Buat objek peta
peta = folium.Map(location=nodes['A'], zoom_start=12)

# Tambahkan marker untuk setiap node
for node, coordinates in nodes.items():
    icon_color = 'green' if node == 'A' else ('red' if node == 'L' else 'blue')
    folium.Marker(location=coordinates, tooltip=f'Node {node}', icon=folium.Icon(color=icon_color)).add_to(peta)

# Hubungkan setiap node dengan jalur dan tampilkan nama jalan
for i in range(len(shortest_path) - 1):
    start_node = shortest_path[i]
    end_node = shortest_path[i + 1]
    start_coordinates = nodes[start_node]
    end_coordinates = nodes[end_node]
    jalan = nama_jalan_map.get((start_node, end_node), f'Jl. {start_node}-{end_node}')
    folium.PolyLine(locations=[start_coordinates, end_coordinates], color='black', tooltip=jalan).add_to(peta)

print("Jalur terdekat:")
print(" -> ".join(shortest_path))
print("Total jarak: {:.2f} kilometer".format(shortest_distance))
print("Nama jalan yang dilalui:")
for i in range(len(shortest_path) - 1):
    start_node = shortest_path[i]
    end_node = shortest_path[i + 1]
    jalan = nama_jalan_map.get((start_node, end_node), f'Jl. {start_node}-{end_node}')
    print(f"{i + 1}. {jalan}")
# Tampilkan peta
peta

```