Imajiner: Jurnal Matematika dan Pendidikan Matematika
ISSN (Online): 2685-3892
Vol. 7, No. 4, Juli 2025, Hal. 256-269
Available Online at journal.upgris.ac.id/index.php/imajiner

Aplikasi Pencarian Rute Optimal Antar Objek Wisata di Kabupaten Cilacap Berbasis Algoritma Floyd-Warshall

Nida Alifa Salsadina[1], Isnaini Rosyida[2]
[1,2]Universitas Negeri Semarang
[1]Email : nidaalifa1612@gmail.com
[2]Email : isnaini@mail.unnes.ac.id

ABSTRAK

Penelitian ini bertujuan untuk membangun jaringan pariwisata dengan mengidentifikasi rute optimal antar destinasi wisata berdasarkan data nama lokasi dan jarak antar lokasi. Metode yang digunakan adalah implementasi Algoritma Floyd Warshall, yang diuji coba melalui dua pendekatan. Pendekatan pertama dilakukan dengan menjalankan perhitungan menggunakan *source code* khusus berbasis *pyhton* sedangkan pendekatan kedua dilakukan melalui perancangan dan penggunaan aplikasi pencarian rute yang dirancang khusus dalam penelitian ini. Aplikasi pada penelitian ini dirancang menggunakan metode *waterfall* yang terdiri dari beberapa tahapan berurutan, yaitu analisis kebutuhan, desain sistem, implementasi, pengujian sistem serta pemeliharaan sistem. Hasil penelitian menunjukkan bahwa kedua pendekatan tersebut menghasilkan rute dan jarak total tempuh yang identik. Konsistensi hasil ini menegaskan efektivitas Algoritma Floyd Warshall dalam menentukan rute terpendek, sekaligus menunjukkan potensi aplikasi ini sebagai alat bantu perencanaan perjalanan yang efisien.

Kata Kunci: Floyd Warshall; Rute Optimal; Aplikasi.

ABSTRACT

This research aims to build a tourism network by identifying optimal routes between tourist destinations based on node name data and distance between nodes. The method used is the implementation of the Floyd Warshall Algorithm, which is tested through two approaches. The first approach is done by running calculations using a special pyhton-based source code while the second approach is done through the design and use of route finding applications specifically designed in this research. The application in this study was designed using the waterfall method which consists of several sequential stages, namely requirements analysis, system design, implementation, system testing and system maintenance. The results show that both approaches produce identical routes and total distance traveled. The consistency of these results confirms the effectiveness of the Floyd Warshall Algorithm in determining the shortest route, as well as showing the potential of this application as an efficient travel planning tool.
Keywords: Floyd warshall; Optimal Routes; Application.

PENDAHULUAN

Menurut United Nasional World Tourism Organization (2019), sektor pariwisata memegang peranan krusial dalam dinamika perekonomian suatu negara, berfungsi sebagai salah satu pilar utama dalam upaya meningkatkan kesejahteraan penduduk. Sektor ini diakui sebagai sektor unggulan yang memiliki daya tarik universal, mampu berkembang pesat, dan menjangkau berbagai lapisan masyarakat di seluruh dunia (Septemuryantoro, 2021). Lebih dari sekedar dampak ekonomi, kegiatan rekreasi yang menjadi inti pariwisata, juga berhasil menjadi solusi untuk meredakan stres dan kecemasan yang muncul akibat tuntutan kehidupan modern. Melakukannya secara teratur tidak hanya memberikan hiburan semata, tetapi juga membawa dampak positif yang penting bagi kesehatan fisik dan mental.

Berdasarkan data dari Badan Pusat Statistik (2022), Kabupaten Cilacap tercatat sebagai wilayah terluas pertama di provinsi tersebut dengan luas mencapai $2.124{,}50\ km^2$. Selain luas wilayah, Kabupaten Cilacap juga memiliki populasi yang besar. Data Badan Pusat

Statistik (2024), menunjukkan bahwa Kabupaten Cilacap menduduki peringkat kedua sebagai wilayah dengan jumlah penduduk terbanyak setelah Kabupaten Brebes, dengan jumlah penduduk mencapai 2.007.829 jiwa. Luas wilayah dan jumlah penduduk yang besar memungkinkan adanya potensi besar untuk pengembangan sektor pariwisata. Menurut informasi dari Dinas Pemuda Olahraga dan Pariwisata (2025), Kabupaten Cilacap menawarkan beragam daya tarik wisata yang mencakup keindahan alam, kekayaan budaya, dan destinasi budaya, seperti Pantai Teluk Penyu, Hutan Payau, Museum Soesilo Soedarman dan destinasi wisata lainnya.

Namun, di balik pesona Kabupaten Cilacap, wisatawan sering kali dihadapkan pada tantangan dalam merencanakan perjalanan. Saat ini, pencarian rute destinasi wisata umumnya dilakukan melalui layanan seperti *Google Maps,* yang hanya menyediakan rute dari satu lokasi awal ke lokasi tujuan. Namun, di era digital serba cepat ini, wisatawan membutuhkan solusi yang lebih fleksibel dan efisien. Salah satu kebutuhannya adalah menemukan rute yang tidak hanya menghubungkan lokasi awal dengan lokasi tujuan, tetapi juga melewati beberapa destinasi wisata perantara. Pendekatan ini memungkinkan mereka untuk mempersingkat waktu perjalanan sekaligus mengunjungi lebih banyak destinasi wisata dalam satu alur perjalanan yang terencana, berbeda dengan pencarian rute standar yang mungkin mengabaikan potensi kunjungan tambahan di sepanjang perjalanan. Dengan demikian, diperlukan pendekatan yang lebih sistematis untuk menyelesaikan permasalahan ini.

Penerapan algoritma pencarian lintasan terpendek, seperti algoritma Floyd-Warshall, dapat menjadi solusi yang tepat. Algoritma ini memungkinkan untuk menghitung dan membandingkan berbagai kemungkinan lintasan, dengan mempertimbangkan faktor-faktor seperti jarak, waktu tempuh, dan biaya. Sebagai salah satu varian dari pemrograman dinamis, algoritma Floyd-Warshall bekerja dengan memandang solusi akhir sebagai hasil dari keputusan – keputusan yang saling berhubungan. Dengan demikian, solusi – solusi tersebut akan terbentuk dari solusi yang ada pada tahap – tahap sebelumnya dan tidak menutup kemungkinan adanya beberapa solusi yang tepat (Ningrum & Andrasto, 2016). Dalam menjalankan prosesnya, algoritma Floyd-Warshall akan mengevaluasi semua kemungkinan lintasan di dalam graf, untuk setiap sisi yang menghubungkan semua simpul (Gunawan et al., 2020).

Selain itu, pembuatan aplikasi atau web yang terhubung dengan data destinasi wisata dan algoritma pencarian lintasan terpendek dapat menjadi solusi yang sangat membantu. Aplikasi atau web semacam ini dapat memberikan rekomendasi lintasan yang disesuaikan dengan pilihan wisatawan, seperti pilihan terhadap jenis destinasi wisata. Dengan demikian, pemanfaatan teknologi dan pendekatan yang lebih sistematis dapat membantu wisatawan dalam merencanakan perjalanan yang lebih tepat dan menyenangkan di Kabupaten Cilacap. Solusi ini tidak hanya meningkatkan pengalaman wisatawan, tetapi juga berpotensi untuk meningkatkan kunjungan wisatawan ke Kabupaten Cilacap, yang pada akhirnya akan berdampak positif pada perekonomian daerah.

Penelitian mengenai lintasan terpendek menggunakan Algoritma Floyd Warshall telah banyak diteliti, seperti yang dilakukan oleh Yustita et al (2018) yang melakukan penelitian terkait Algoritma Floyd Warshall menggunakan komputasi program MATLAB untuk penentuan rute terpendek model jaringan pariwisata Kabupaten Banyuwangi. selain menggunakan algoritma Floyd-Warshall, penentuan lintasan terpendek juga telah banyak diteliti menggunakan algoritma lain, seperti yang dilakukan oleh Puspita & Yanto (2024) meneliti tentang Algoritma Dijkstra untuk menentukan lintasan terpendek lokasi wisata budaya di Kota Padang. Selain penelitian tentang lintasan terpendek, penelitian lain terkait aplikasi juga telah banyak dipelajari dan dikembangkan, seperti yang dilakukan oleh Kristanto et al (2023) yang meneliti tentang perancangan aplikasi android untuk pemilihan tempat

wisata di Kota Kediri. Penelitian mengenai perancangan aplikasi menggunakan Algoritma Floyd Warshall sebelumnya telah dilakukan oleh Siregar et al (2021) yang meneliti tentang sistem informasi geografis lokasi pariwisata Kota Padang Sidempuan berbasis android menggunakan algoritma Floyd-Warshall.

Berdasarkan uraian di atas, peneliti tertarik untuk melakukan penelitian perancangan aplikasi pencarian rute optimal antar objek wisata di Kabupaten Cilacap menggunakan Algoritma Floyd Warshall. Penelitian ini bertujuan untuk membuat model jaringan graf untuk merepresentasikan jalur objek wisata di Kabupaten Cilacap, menerapkan Algoritma Floyd Warshall untuk pencarian rute optimal antar objek wisata di Kabupaten Cilacap, dan menampilkan visualisasi aplikasi pencarian rute optimal antar objek wisata di Kabupaten Cilacap menggunakan Algoritma Floyd Warshall.

METODE PENELITIAN

Penelitian yang digunakan merupakan penelitian kualitatif dan penelitian kuantitatif. Penelitian kualitatif merupakan penelitian yang menekankan pada kondisi objek ilmiah peneliti sebagai instrumen kunci, di mana data berupa jarak dan lintasan perjalanan yang terdapat pada *Google maps* yang diidentifikasi dan dikumpulkan menggunakan metode dokumentasi (Sugiyono, 2013). Selanjutnya, penelitian kuantitatif digunakan untuk mengolah data jarak antar objek wisata yang telah didumpullkan, di mana penelitian kuantitatif merupakan penelitian yang dinyatakan dengan menggunakan angka statistik dengan instrumen penelitian untuk menganalisis data (Sugiyono, 2013).

Metode yang digunakan pada penelitian ini adalah studi pustaka, pengumpulan data, cara pemecahan masalah dan penarikan kesimpulan. Studi pustaka menjelaskan tentang sumber relevan yang digunakan untuk mengumpulkan informasi yang diperlukan dalam penelitian. Studi pustaka dilakukan dengan memanfaatkan berbagai sumber seperti, buku, jurnal, dan artikel yang berhubungan erat dengan topik permasalahan yang diteliti. Pada akhirnya, sumber pustaka akan dijadikan landasan untuk menganalisis permasalahan.

Pengumpulan data dilakukan dengan cara mengambil data sekunder menggunakan metode dokumentasi data yaitu, mengumpulkan dan mencatat informasi yang diperoleh dari sumber yang telah terdokumentasi sebelumnya (Sugiyono, 2022). Data yang dikumpulkan berupa nama objek wisata dan jalan penghubung antar lokasi beserta jarak antar lokasi objek wisata, terminal, stasiun serta bandara di Kabupaten Cilacap yang diperoleh dari *Google Maps*. Proses pengumpulan data ini bertujuan untuk memperoleh informasi akurat mengenai hubungan antar lokasi yang selanjutnya akan digunakan dalam menganalisis rute optimal menggunakan Algoritma Floyd Warshall.

Langkah – langkah yang digunakan untuk memecahkan masalah dalam penelitian ini dilakukan dengan cara memodelkan data lokasi yang telah dikumpulkan dengan simpul dan jalan penghubung antar lokasi disajikan sebagai sisi, dan jarak antar lokasi disajikan sebagai bobot sisi. Kemudian membuat model jaringan dalam bentuk graf dari matriks berbobot yang memiliki ukuran $n \times n$, dimana n merupakan banyaknya simpul dalam graf yang elemennya berisi bobot sisi atau jarak antar simpul, dengan

$$d_{i,j} = \begin{cases} bobot\ sisi & , \text{jika } i \neq j \in E \left(\text{simpul } v_i \text{ terhubung dengan simpul } v_j\right) \\ \infty & , \begin{pmatrix} \text{jika } i = j \text{ (simpul } v_i \text{ terhubung ke simpul } v_i \text{ itu sendiri)} \\ \text{jika } i \neq j \notin E \left(\text{simpul } v_i \text{ tidak terhubung dengan simpul } v_j\right) \end{pmatrix} \end{cases}$$

Pada proses ini, diperoleh matriks awal berupa W_0 dengan output $W = W_0 = \left[d_{(ij)}^{(0)}\right]$.

Setelah model jaringan dibentuk dalam bentuk graf merepresentasikan jarak antar simpul, langkah selanjutnya adalah menghitung rute optimal menggunakan Algoritma Floyd Warshall menggunakan bahasa pemrograman *python*. Prosesnya melibatkan iterasi untuk

membandingkan dan memperbarui jarak terpendek antar semua pasangan simpul, dengan mempertimbangkan setiap simpul sebagai perantara.

Langkah – langkah perhitungan tersebut adalah sebagai berikut:

1. Proses perhitungan awal pada Algoritma Floyd Warshall, dilakukan dengan membuat matriks berbobot awal dengan ukuran $n \times n$.
2. Kemudian melakukan iterasi yang dimulai dari iterasi ke – 0 sampai dengan iterasi ke – n, dengan n jumlah simpul, menggunakan rumus

$$d_{i,j}^{(k)} = \min\left\{d_{i,j}^{(k-1)}, d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}\right\}$$

untuk $k = 1,2, \dots, n$ dengan k menyatakan iterasi, di mana:

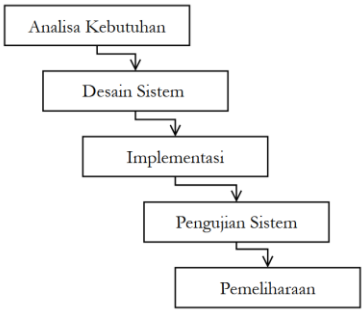
a. Apabila bobot $d_{i,j}^{(k-1)} > d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}$, maka bobot $d_{i,j}^{(k-1)}$ akan ditukar dengan bobot $d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}$.

b. Apabila bobot $d_{i,j}^{(k-1)} < d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}$, maka bobot $d_{i,j}^{(k-1)}$ tidak ditukar dengan bobot $d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}$, dengan kata lain bobot tersebut tetap.

Pada proses ini, diperoleh matriks berupa W_k dengan output $W = W_k = \left[d_{(ij)}^{(k)}\right]$.

3. Iterasi akan dilakukan sebanyak n jumlah simpul, dikarenakan penelitian ini menggunakan 57 simpul maka iterasi akan dilakukan sebanyak 57 kali. Hasil akhir dari iterasi pada Algoritma Floyd Warshall berupa matriks berbobot akhir yang berukuran $n \times n$ berupa W_n dengan output $W = W_n = \left[d_{(ij)}^{(n)}\right]$, di mana pada matriks W_n terdapat informasi mengenai rute optimal untuk setiap simpul.

Perancangan aplikasi pada penelitian ini akan dilakukan menggunakan metode *waterfall*. Metode *waterfall* adalah suatu model pengembangan perangkat lunak yang mengikuti pendekatan linier dan berurutan. Metode *waterfall* mengikuti aliran air terjun, di mana setiap tahap mengalir ke tahap berikutnya secara berurutan, setiap tahap harus diselesaikan sepenuhnya sebelum tahap berikutnya dimulai (Kristanto et al., 2023). Tahapan dalam metode *waterfall* adalah sebagai berikut:



Gambar 1. Metode *Waterfall*

HASIL DAN PEMBAHASAN

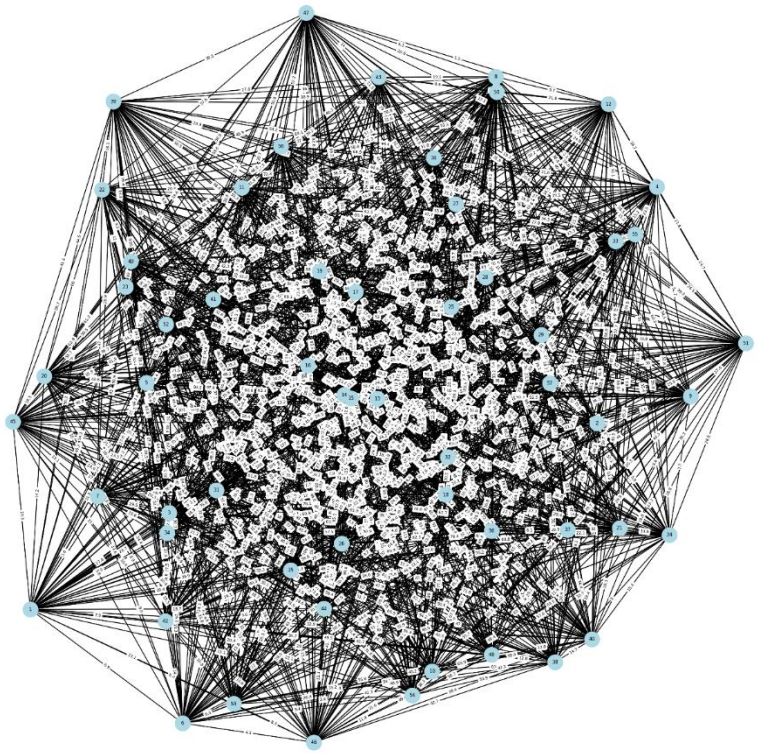
Penelitian ini berlokasi di Kabupaten Cilacap, yang secara informal terbagi menjadi empat wilayah berdasarkan karakteristik geografisnya. Cilacap Barat didominasi dataran tinggi dan perbukitan, berbatasan langsung dengan Jawa Barat. Cilacap Timur merupakan dataran rendah pesisir. Di bagian utara, Cilacap Utara menjadi penghubung antara dataran rendah dan perbukitan di Jawa Tengah. Sementara itu, Cilacap Selatan dikenal dengan wilayah pesisir dan pulau-pulau kecil, termasuk Nusakambangan.

Dalam penelitian ini, nama destinasi wisata, bandara, stasiun, dan terminal akan menjadi simpul, sementara jarak antar simpul berfungsi sebagai bobot sisi. Jarak ini menunjukkan kedekatan lokasi dalam jaringan yang menjadi dasar analisis rute optimal. Dengan begitu, struktur data penelitian ini membentuk graf berbobot yang memungkinkan penerapan algoritma untuk mencari rute optimal antar simpul.

Tabel 1. Data Nama Simpul

Simpul	Nama Simpul	Simpul	Nama Simpul
1	Bandar Udara Tunggul Wulung	30	Konservasi Penyu Cilacap
2	Stasiun Sidareja	31	Alun – alun Kroya
3	Stasiun Kroya	32	Museum Susilo Soedarman
4	Stasiun Maos	33	Taman Sari Rasa
5	Stasiun Gandrungmangu	34	Wisata Edukasi Kampung Kreatif Karisma
6	Stasiun Gumilir	35	Kawista Emji
7	Stasiun Jeruklegi	36	Curug Taman Desa Ciruyung
8	Stasiun Cilacap	37	Air Terjun Pengantin
9	Terminal Sidareja	38	Taman Wisata Dinosaurus
10	Terminal Kroya	39	Alun - Alun Gandrungmangu
11	Terminal Karangpucung	40	Puncak Bukit Urip
12	Terminal Bangga Mbangun Desa	41	Curug Kedungborang
13	Puncak Gelewer Dayeuhluhur	42	Curug Mandala
14	Cidayeuh Canyon	43	Havana Hills
15	Wisata Air Curug Ngebul	44	Puncak Wadas Tumpang
16	Air Terjun Curug Bandung	45	Pantai Menganti
17	Curug Panto Cinagara	46	Wisata Hutan Payau
18	Alun – Alun Majenang	47	Stadion Wijaya Kusuma
19	Wisata Gunung Payung Cilacap	48	Taman Pesona Rengganis
20	Wisata Petik Durian Pesahangan	49	Rumah Makan Mbah Suro
21	Pemandian Air Panas Cipari	50	Waduk Kubangkangkung
22	Curug Cipari	51	Wisata Bukit Pelangi
23	Nirwana Penyarang Land	52	Kampoeng Kepiting
24	Kemit Forest Education	53	Alun - alun Cilacap
25	Pantai Jetis	54	Benteng Pendem Cilacap
26	Pantai Karangpakis	55	Pantai Teluk Penyu
27	Pantai Widarapayung	56	Dermaga Ujung gagak
28	Pantai Ketapang Indah	57	Pelabuhan Wijaya Kusuma
29	Wisata Mangrove Bunton		

Pemodelan jaringan graf dalam penelitian ini melibatkan penggambaran simpul berupa destinasi wisata, bandara, stasiun, dan terminal. Setiap objek pada Tabel 1 terhubung ke 56 objek lainnya, sehingga terbentuk matriks berbobot berukuran 57×57 seperti yang disajikan pada Gambar 3, dengan bobot menyatakan jarak antar lokasi. Pemodelan ini membantu visualisasi hubungan kompleks antar simpul, memudahkan analisis efisiensi konektivitas, perencanaan rute, dan identifikasi simpul kunci. Visualisasi model jaringan dapat dilihat pada Gambar 2.



Gambar 2. Jaringan Pariwisata

*) Data jarak antar lokasi pada jaringan di atas dimuat pada *source code* khusus berbasis *python* yang terdapat pada Gambar 3.

Berdasarkan uraian di atas, penelitian ini memodelkan jaringan rute destinasi wisata di Kabupaten Cilacap dalam bentuk graf berbobot yang terdiri dari 57 titik lokasi seperti destinasi wisata, bandara, stasiun, dan terminal, dengan jarak antar lokasi sebagai bobot penghubungnya. Kabupaten Cilacap dibagi menjadi empat wilayah yaitu, barat, timur, utara, dan selatan yang mempunyai karakter geografis yang berbeda – beda. Pemodelan ini dibuat untuk melihat hubungan antar lokasi. Dengan struktur seperti ini, Algoritma Floyd Warshall dapat diterapkan untuk menghitung rute optimal destinasi wisata di Kabupaten Cilacap. Gambar 2 menampilkan visualisasi jaringan destinasi wisata di Kabupaten Cilacap.

Guna memudahkan pengguna dalam memahami dan menerapkan Algoritma Floyd Warshall, dikembangkan sebuah *source code* khusus berbasis *Python. Python* dapat digunakan untuk berbagai tujuan, termasuk pengembangan web sisi server, pembuatan perangkat lunak, otomatisasi skrip sistem, dan penyelesaian masalah matematika yang rumit (Ma'arif, 2020). Perancangan *source code* ini berfokus pada kemudahan penggunaan, memungkinkan peneliti atau pengguna untuk secara efisien menentukan rute optimal antar lokasi yang akan dicari.

Dengan memanfaatkan bahasa pemrograman komputasi *Python,* proses perhitungan rute optimal yang kompleks dapat dilakukan jauh lebih cepat dan akurat dibandingkan metode perhitungan manual yang sangat rentan terhadap kesalahan dan memakan waktu. Implementasi ini secara signifikan meningkatkan efisiensi operasional dan analisis data rute dalam penelitian. Berikut ini ditampilkan sebuah *source code* khusus berbasis *Python* untuk menghitung perhitungan setiap iterasi pada penerapan Algoritma Floyd Warshall yang menampilkan jarak dan rute optimal antar lokasi.

```python
def floyd_warshall_with_labels(graph, labels):
    # Jumlah node dalam graf
    n = len(graph)

    # Inisialisasi matriks jarak dan next_node
    distances = [[graph[i][j] for j in range(n)] for i in range(n)]
    next_node = [[None if graph[i][j] == float('inf') else j for j in range(n)] for i in range(n)]

    print("Matriks Jarak Awal (distances):")
    for i, row in enumerate(distances):
        print(f"{labels[i]}: {row}")
    print("\n")

    # Floyd-Warshall Algorithm
    for k in range(n):
        print(f"Iterasi k = {k + 1}")
        for i in range(n):
            for j in range(n):
                if distances[i][k] + distances[k][j] < distances[i][j]:
                    distances[i][j] = distances[i][k] + distances[k][j]
                    next_node[i][j] = next_node[i][k]

        # Tampilkan matriks jarak dan jalur setelah iterasi
        print("Matriks Jarak (distances):")
        for i, row in enumerate(distances):
            print(f"{labels[i]}: {row}")
        print("\n")

    return distances, next_node
```

```python
def reconstruct_path_with_labels(next_node, start, end, labels):
    if next_node[start][end] is None:
        return None  # Tidak ada jalur
    path = [labels[start]]
    while start != end:
        start = next_node[start][end]
        path.append(labels[start])
    return path

# Contoh Penggunaan
graph = [
    [float('inf'), 39.3, 27.8, 13.5, 31.3, 6.9, 3.6, 14.1, 41.1, 28.4, 43.4, 9.3, 93.6, 105, 91.8, 76.6, 79, 64.7, 75.2, 14.2, 49.7, 50.2, 46.2, 42.4, 44.2, 39.9, 30.8, 35.1,
     22.1, 23.4, 26.7, 25, 28.7, 16.9, 19.4, 52.3, 45.7, 36.7, 32.1, 32.9, 32.3, 7.9, 5.2, 18.7, 10.3, 6.8, 8.2, 43.4, 54.2, 13.2, 13.2, 24.3, 13.1, 15.1, 13.3, 44.4, 15.5],
    [39.3, float('inf'), 65.2, 50.9, 8.1, 45.4, 36.1, 52.6, 1.8, 65.8, 16.2, 47.8, 52.9, 65.6, 52.6, 40.2, 45.1, 30.8, 41.3, 49.1, 10.3, 10.8, 7, 5.9, 82.2, 78, 68.9, 73.1,
     60, 61.4, 64.1, 59.7, 58.6, 54.3, 56.8, 25.2, 24.5, 8.1, 7.3, 18, 17.4, 45.3, 35, 46.8, 48.8, 45.3, 46.7, 48, 12.2, 26.1, 25.9, 39.2, 52.1, 54.4, 52.7, 28.2, 54.1],
    [27.8, 65.2, float('inf'), 14.5, 64.4, 26.2, 29.5, 33.8, 67, 1.9, 54.8, 28.8, 114, 126, 113, 88, 90.4, 76.1, 86.6, 21.2, 75.6, 76.1, 67, 62.5, 20.5, 17.4, 9.5, 12.5,
     16.2, 13.6, 1.1, 7.7, 7.2, 17.9, 11.9, 63.7, 57.1, 62, 58.7, 59.6, 58.9, 27.5, 31.1, 25.6, 24.6, 31, 27.8, 70.1, 77.7, 39.1, 39.1, 51, 32.6, 35.5, 33.8, 71.5, 35.4],
    [13.5, 50.9, 14.5, float('inf'), 42.8, 15.9, 15.2, 23.5, 52.7, 15.1, 45.1, 18.7, 105, 116, 103, 78.3, 80.8, 66.5, 76.9, 6.8, 61.3, 61.8, 57.3, 52.8, 32.4, 28.2, 19.1, 23.3,
     10.4, 11.6, 13.4, 12.6, 15.2, 3.4, 5.9, 54.1, 47.5, 48.3, 44.4, 45.2, 44.6, 13.2, 16.7, 11.3, 14.4, 20.7, 17.5, 55.7, 66.5, 24.8, 24.7, 36, 22.4, 25.3, 23.5, 57.1, 25.1],
    [31.3, 8.1, 64.4, 42.8, float('inf'), 37.3, 37.6, 44.6, 9.9, 57.7, 17.2, 39.7, 62.3, 73.7, 60.6, 48.2, 53, 38.7, 49.1, 43.6, 18.3, 18.8, 14.1, 9.6, 74.2, 69.9, 60.8, 65.1,
     52.1, 53.4, 56, 54.4, 58, 46.2, 48.7, 26.2, 25.5, 5.5, 0.85, 10.9, 10.2, 37.3, 26.9, 45.3, 40.7, 37.3, 38.6, 11.7, 20.2, 18, 17.8, 31.1, 44.1, 46.3, 44.6, 20.1, 46],
    [6.9, 45.4, 26.2, 15.9, 37.3, float('inf'), 9.6, 7.6, 47.1, 26.8, 49.3, 2.7, 99.7, 111, 98, 82.7, 85.2, 70.9, 81.3, 17.6, 55.9, 56.4, 52.4, 48.5, 40.7, 36.5, 27.4, 31.6,
     18.7, 19.9, 24.3, 24.2, 29.2, 19.3, 21.8, 58.4, 51.9, 42.7, 38.2, 39.1, 38.4, 14, 11.1, 22, 5.6, 4.3, 1.7, 49.4, 60.1, 19.2, 19.1, 30.3, 6.5, 9.4, 6.8, 50.8, 9.2],
    [3.6, 36.1, 29.5, 15.2, 27.6, 9.6, float('inf'), 16.9, 37.9, 30.1, 40.2, 12.1, 90.3, 102, 88.6, 73.4, 75.8, 61.5, 72, 16, 46.5, 47, 43, 39.2, 46.5, 42.3, 33.2, 37.4,
     24.5, 25.7, 28.4, 26.8, 30.4, 18.6, 21.1, 49.1, 42.5, 33.5, 28.9, 29.7, 29.1, 8.9, 1.9, 20.2, 13.1, 9.6, 11, 40.2, 50.9, 10, 9.9, 21.1, 15.5, 18.7, 17, 41.6, 18.4],
    [14.1, 52.6, 33.8, 23.5, 44.6, 7.6, 16.9, float('inf'), 55, 34.4, 57.3, 5.7, 107, 119, 106, 90.5, 93, 78.6, 89.1, 25.6, 63.7, 64.1, 60.1, 56.3, 48.3, 44.1, 35, 39.2,
     26.2, 27.5, 32.6, 31.8, 36.8, 26.9, 29.4, 66.2, 59.7, 50.6, 46, 46.9, 46.2, 21.8, 19.1, 30, 11.6, 9.7, 6.1, 57.4, 64.9, 27.1, 26.5, 38.3, 1.3, 2.7, 2.1, 58.8, 2],
    [41.1, 1.8, 67, 52.7, 9.9, 47.1, 37.9, 55, float('inf'), 67.6, 18, 49.6, 52.5, 63.8, 50.7, 38.4, 49.1, 34.3, 45.9, 50.9, 8.6, 9.1, 8.7, 7.7, 84, 79.8, 76.6, 74.9, 62,
     63.2, 65.9, 61.5, 60.3, 56.1, 58.6, 27, 26.3, 9.9, 9.1, 19.8, 19.1, 47.1, 36.8, 48.6, 50.6, 47.1, 48.5, 3, 10.3, 27.9, 27.6, 41, 53.9, 56.4, 54.5, 29.9, 55.8],
    [28.4, 65.8, 1.9, 15.1, 57.7, 26.8, 30.1, 34.4, 67.6, float('inf'), 55.8, 29.4, 115, 127, 114, 89, 91.4, 77.1, 87.6, 21.8, 76.2, 76.7, 68, 63.5, 20.8, 16.2, 7.8, 11.3,
     16.8, 12.2, 1.7, 8.7, 8.2, 18.6, 12.5, 64.7, 58.1, 63, 58.6, 59.4, 58.8, 29, 31.7, 26.2, 25.2, 31.6, 28.4, 69.9, 78.7, 39.7, 39.7, 50.9, 33.2, 36.3, 34.4, 72.1, 36],
```

```
    [43.4, 16.2, 54.8, 45.1, 17.2, 49.3, 40.2, 57.3, 18, 55.8, float('inf'), 51.8, 59.6, 70.9, 57.9, 33.3, 35.7, 21.4, 31.9, 38.3, 26.5, 26.9, 15.1, 13.1, 77.3, 73.1, 64,
     68.2, 55.3, 56.5, 55, 48.8, 47.7, 48.5, 51.5, 9, 8.3, 12.6, 16.5, 26.1, 25.4, 37.3, 39.2, 36, 52.9, 49.4, 50.8, 20.9, 28.3, 35.3, 35, 48.5, 55.6, 57.6, 55.9, 37.3, 58.1],
    [9.3, 47.8, 29, 18.7, 39.7, 2.7, 12.1, 5.7, 49.6, 29.4, 51.8, float('inf'), 102, 114, 100, 82.5, 87.6, 73.3, 83.3, 20.4, 58.4, 58.8, 54.8, 51, 43.6, 39.3, 30.2, 34.5,
     21.3, 22.8, 27.9, 27.1, 32.1, 22.2, 24.4, 60.9, 54.3, 45.3, 40.7, 41.6, 40.9, 16.5, 13.8, 25.3, 7.4, 4.8, 1.5, 52.1, 46.7, 21.8, 21.8, 32.9, 3.8, 6.3, 4.4, 53.5, 7.3],
    [93.6, 52.9, 114, 105, 62.3, 99.7, 90.3, 107, 52.5, 115, 59.6, 102, float('inf'), 21.1, 21, 22.8, 44.4, 34.7, 43.5, 97.8, 45.9, 46.4, 55.8, 60.2, 136, 132, 123, 127,
     114, 116, 115, 108, 107, 108, 111, 66, 65.3, 62.3, 60.2, 72.2, 71.6, 96.8, 89.2, 95.5, 103, 99.6, 101, 55.1, 43.6, 80.4, 80.1, 93.6, 106, 109, 107, 82.4, 108],
    [105, 65.6, 126, 116, 73.7, 111, 102, 119, 63.8, 127, 70.9, 114, 21.1, float('inf'), 27, 35, 64.3, 46.5, 63.4, 109, 57.2, 57.7, 67.1, 71.5, 148, 144, 123, 139,
     126, 127, 126, 120, 119, 119, 122, 77.3, 76.6, 73.7, 72.2, 83.6, 82.9, 108, 101, 107, 114, 111, 112, 66.4, 55, 91.7, 91.4, 105, 117, 119, 117, 93.7, 120],
    [91.8, 52.6, 113, 103, 60.6, 98, 88.6, 106, 50.7, 114, 57.9, 100, 21, 27, float('inf'), 14.3, 51.2, 30.5, 44.4, 90.2, 44.1, 44.6, 54.1, 58.4, 129, 125, 116, 120,
     107, 108, 107, 101, 99.6, 100, 103, 58.4, 57.7, 60.6, 59.8, 70.5, 69.9, 89.2, 87.5, 87.9, 101, 97.8, 99.2, 53.7, 44.9, 78.6, 78.4, 91.8, 105, 107, 105, 80.7, 107],
    [76.6, 40.2, 88, 78.3, 48.2, 82.7, 73.4, 90.5, 38.4, 89, 33.3, 82.5, 22.8, 35, 14.3, float('inf'), 26.6, 11.9, 25.8, 71.5, 31.8, 32.3, 38.9, 46.3, 111, 106, 97.2, 101,
     86.6, 87.9, 88.2, 82, 80.9, 81.7, 84.7, 39.7, 39, 45.9, 47.5, 58.2, 57.5, 70.5, 72.4, 69.2, 86.1, 82.6, 84, 41.4, 32.6, 66.3, 66, 79.5, 88.7, 91.9, 89.4, 70.6, 91.3],
    [79, 45.1, 90.4, 80.8, 53, 85.2, 75.8, 93, 49.1, 91.4, 35.7, 87.6, 44.4, 64.3, 51.2, 26.6, float('inf'), 14.8, 28.2, 74, 42.5, 43, 41.4, 43.3, 113, 109, 99.7, 103, 90.8,
     92.2, 90.7, 84.5, 83.4, 84.2, 87.2, 42.1, 41.4, 48.3, 52.3, 61.1, 55.1, 73, 74.9, 71.7, 88.5, 85.1, 86.5, 52.1, 43.2, 71.1, 70.8, 84.2, 91.3, 93.3, 91.6, 73.1, 93.8],
    [39.3, 30.8, 76.1, 66.5, 38.7, 70.9, 61.5, 78.6, 34.3, 77.1, 21.4, 73.3, 34.7, 46.5, 30.5, 11.9, 14.8, float('inf'), 13.9, 59.6, 27.7, 28.2, 24.2, 28.4, 98.6, 94.4, 85.3,
     89.5, 76.6, 77.8, 76.3, 70.2, 69, 69.9, 72.4, 27.8, 27.1, 34, 37.9, 47.5, 41.7, 58.6, 60.5, 57.3, 74.2, 70.7, 72.1, 37.3, 28.4, 56.7, 56.4, 69.9, 77.5, 80, 78.1, 58.7, 79.5],
    [75.2, 41.3, 86.6, 76.9, 49.1, 81.3, 72, 89.1, 45.9, 87.6, 31.9, 83.3, 43.5, 63.4, 44.4, 25.8, 28.2, 13.9, float('inf'), 70.1, 38.3, 38.8, 34.7, 38.9, 109, 105, 95.8, 100,
     87.1, 88.3, 86.8, 80.7, 79.5, 80.4, 83.3, 38.3, 37.6, 43.8, 48.4, 57.3, 52.9, 69.1, 71, 67.8, 84.7, 81.2, 82.6, 51.2, 42.3, 67.2, 66.9, 80.4, 88, 90.5, 88.6, 69.2, 89.9],
    [14.2, 49.1, 21.2, 6.8, 43.6, 17.6, 16, 25.6, 50.9, 21.8, 38.3, 20.4, 97.8, 109, 90.2, 71.5, 74, 59.6, 70.1, float('inf'), 59.5, 60, 50.5, 46, 39, 34.8, 25.7, 29.9, 17,
     18.2, 20.1, 18.4, 20.4, 10.3, 12.8, 47.2, 40.6, 45.5, 44.5, 45.3, 44.7, 10.6, 17.5, 6.6, 16.5, 18.9, 19.6, 53.9, 61.2, 25.6, 25.5, 36.7, 24.4, 27.3, 25.6, 57.2, 27.2],
    [49.7, 10.3, 75.6, 61.3, 18.3, 55.9, 46.5, 63.7, 8.6, 76.2, 26.5, 58.4, 45.9, 57.2, 44.1, 31.8, 42.5, 27.7, 38.3, 59.5, float('inf'), 0.5, 11.5, 13.9, 92.5, 88.3, 79.2,
     83.4, 70.3, 71.7, 74.4, 70, 68.8, 64.6, 67.1, 35.5, 32, 18.4, 17.6, 28.3, 27.6, 55.6, 45.4, 57.3, 59.2, 55.7, 57.1, 11.6, 8.7, 36.5, 36.3, 49.7, 62.6, 65, 63.1, 38.6, 64.5],
    [50.2, 10.8, 76.1, 61.8, 18.8, 56.4, 47, 64.1, 9.1, 76.7, 26.9, 58.8, 46.4, 57.7, 44.6, 32.3, 43, 28.2, 38.8, 60, 0.5, float('inf'), 12, 14.4, 93.1, 88.9, 79.8, 84, 71.1,
     72.3, 75, 70.6, 69.5, 65.2, 67.7, 33.2, 32.5, 18.8, 18.1, 28.8, 28.1, 56.2, 45.9, 57.7, 59.7, 56.2, 57.6, 12.1, 9.2, 37, 36.8, 50.1, 62.5, 64.8, 63.1, 39.1, 65],
    [46.2, 7, 67, 57.3, 14.1, 52.4, 43, 60.1, 8.7, 68, 15.1, 54.8, 55.8, 67.1, 54.1, 38.9, 41.4, 24.2, 34.7, 50.5, 11.5, 12, float('inf'), 4.5, 89.1, 84.9, 75.8, 80, 67.1,
     68.3, 67.3, 61.1, 60, 60.8, 63.7, 21.5, 20.8, 9.5, 13.4, 22.9, 22.3, 49.5, 41.9, 48.2, 55.7, 52.2, 53.6, 11.7, 19, 32.1, 31.9, 46.1, 58.5, 60.8, 59.1, 34.2, 61],
    [42.4, 5.9, 62.5, 52.8, 9.6, 48.5, 39.2, 56.3, 7.7, 63.5, 13.1, 51, 60.2, 71.5, 58.4, 46.3, 43.3, 28.4, 38.9, 46, 13.9, 14.4, 4.5, float('inf'), 85, 80.8, 71.7, 75.9,
     63, 64.2, 62.7, 56.6, 55.4, 56.3, 59.2, 22.1, 21.4, 5, 8.8, 18.4, 17.8, 45, 38, 43.7, 51.9, 48.4, 49.8, 10.7, 18, 29.2, 28.9, 42.3, 54.6, 57.6, 55.8, 31.2, 57.1],
    [44.2, 82.2, 20.5, 32.4, 74.2, 40.7, 46.5, 48.3, 84, 20.8, 77.3, 43.6, 136, 148, 129, 111, 113, 98.6, 109, 39, 92.5, 93.1, 89.1, 85, float('inf'), 6, 15.7, 11.1, 28.5,
     23.3, 21.1, 28.1, 28.5, 35.8, 31.9, 86.5, 78.4, 79.6, 75, 75.8, 75.2, 44.5, 48.1, 43.8, 39.2, 45.5, 42.3, 86.3, 97.1, 56.1, 56.1, 65.2, 47.2, 50, 48.3, 87.7, 49.9],
    [39.9, 78, 17.4, 28.2, 69.9, 36.5, 42.3, 44.1, 79.8, 16.2, 73.1, 39.3, 132, 144, 125, 106, 109, 94.4, 105, 34.8, 88.3, 88.9, 84.9, 80.8, 6, float('inf'), 11.5, 6.9,
     24.3, 19.1, 18, 24.9, 24.5, 31.6, 28.7, 82.3, 74.4, 75.4, 70.8, 71.6, 71, 40.3, 43.9, 39.6, 35, 41.3, 38.1, 82.1, 92.9, 51.9, 51.9, 63, 43, 45.3, 43.5, 83.5, 45.7],
```

```
[30.8, 68.9, 9.5, 19.1, 60.8, 27.4, 33.2, 35, 76.6, 7.8, 64, 30.2, 123, 123, 116, 97.2, 99.7, 85.3, 95.8, 25.7, 79.2, 79.8, 75.8, 71.7, 15.7, 11.5, float('inf'), 6.4,
 15.2, 10, 9.1, 18, 17.8, 22.5, 19.5, 73.2, 66.6, 66.3, 61.7, 62.5, 61.9, 31.2, 34.5, 30.2, 25.9, 32.2, 29, 73, 83.8, 42.8, 42.8, 54, 33.9, 36.7, 34.3, 74.4, 36.6],
[35.1, 73.1, 12.5, 23.3, 65.1, 31.6, 37.4, 39.2, 74.9, 11.3, 68.2, 34.5, 127, 139, 120, 101, 103, 89.5, 100, 29.9, 83.4, 84, 80, 75.9, 11.1, 6.9, 6.4, float('inf'),
 19.4, 14.3, 13.1, 20, 19.6, 26.7, 23.8, 77.5, 69.5, 70.5, 65.9, 66.8, 66.1, 35.4, 39, 34.7, 30.1, 35.7, 33.2, 77.2, 88, 47, 47, 58.3, 38.1, 41, 39.2, 78.6, 40.8],
[22.1, 60, 16.2, 10.4, 52.1, 18.7, 24.5, 26.2, 62, 16.8, 55.3, 21.3, 114, 126, 107, 86.6, 90.8, 76.6, 87.1, 17, 70.3, 71.1, 67.1, 63, 28.5, 24.3, 15.2, 19.4, float('inf'),
 7.7, 15.1, 14.2, 19.3, 13.6, 11.6, 64.3, 57.7, 57.6, 52.8, 53.6, 53, 22.5, 25.8, 21.5, 16.9, 23.2, 20.1, 64.1, 72.3, 34.1, 33.8, 45, 24.9, 28.2, 25.5, 65.5, 27.7],
[23.4, 61.4, 13.6, 11.6, 53.4, 19.9, 25.7, 27.5, 63.2, 12.2, 56.5, 22.8, 116, 127, 108, 87.9, 92.2, 77.8, 88.3, 18.2, 71.7, 72.3, 68.3, 64.2, 23.3, 19.1, 18, 14.3, 7.7,
 float('inf'),  12.5, 15.7, 19.9, 15, 13, 65.7, 59.2, 58.8, 54.2, 55, 54.4, 23.7, 27.3, 22.7, 18.4, 24.7, 21.5, 65.5, 76.3, 35.3, 35.3, 46.4, 26.4, 28.7, 27, 66.9, 29.1],
[26.7, 64.1, 1.1, 13.4, 56, 24.3, 28.4, 32.6, 65.9, 1.7, 55, 27.9, 115, 126, 107, 88.2, 90.7, 76.3, 86.8, 20.1, 74.4, 75, 67.3, 62.7, 21.1, 18, 9.1, 13.1, 15.1, 12.5,
 float('inf'), 7.9, 7.5, 16.9, 10.8, 63.9, 57.4, 61.5, 56.9, 57.7, 57.1, 27.3, 30, 24.5, 23.5, 28.9, 26.7, 68.2, 78, 38, 40.1, 31.5, 33.8, 32.1, 69.6, 34.3],
[25, 59.7, 7.7, 12.6, 54.4, 24.2, 26.8, 31.8, 61.5, 8.7, 48.8, 27.1, 108, 120, 101, 82, 84.5, 70.2, 80.7, 18.4, 70, 70.6, 61.1, 56.6, 28.1, 24.9, 18, 20, 14.2, 15.7,
 7.9, float('inf'), 5.8, 12.4, 5.6, 57.8, 51.2, 56.1, 55.2, 56.1, 55.4, 24.8, 28.3, 22.9, 22.7, 28.9, 25.8, 64.5, 72.8, 36.3, 36.3, 47.6, 30.7, 33.8, 31, 68, 33.4],
[28.7, 58.6, 7.2, 15.2, 58, 29.2, 30.4, 36.8, 60.3, 8.2, 47.7, 32.1, 107, 119, 99.6, 80.9, 83.4, 69, 79.5, 20.4, 68.8, 69.5, 60, 55.4, 28.5, 24.5, 17.8, 19.6, 19.3,
 19.9, 7.5, 5.8, float('inf'), 16.1, 11.3, 56.6, 50.1, 55, 58.9, 59.7, 55.9, 29.2, 31.9, 23.4, 27.7, 33.3, 32.7, 63.3, 70.7, 40, 39.9, 51.2, 37.6, 40.7, 38.2, 71.6, 40.3],
[16.9, 54.3, 17.9, 3.4, 46.2, 19.3, 18.6, 26.9, 56.1, 18.6, 48.5, 22.2, 108, 119, 100, 81.7, 84.2, 69.9, 80.4, 10.3, 64.6, 65.2, 60.8, 56.3, 35.8, 31.6, 22.5, 26.7, 13.6,
 15, 16.9, 12.4, 16.1, float('inf'), 6.8, 57.5, 50.9, 51.7, 47.1, 47.9, 47.3, 16.6, 20.1, 14.7, 17, 22.9, 20.9, 58.4, 66.4, 28.2, 28.1, 39.4, 25.8, 28.9, 26.4, 59.8, 28.5],
[19.4, 56.8, 11.9, 5.9, 48.7, 21.8, 21.1, 29.4, 58.6, 12.5, 51.5, 24.4, 111, 122, 103, 84.7, 87.2, 72.4, 83.3, 12.8, 67.1, 67.7, 63.7, 59.2, 31.9, 28.7, 19.5, 23.8, 11.6,
 13, 10.8, 5.6, 11.3, 6.8, float('inf'), 60.4, 53.8, 54.2, 49.6, 50.4, 49.8, 19.1, 22.6, 17.2, 20.3, 25.4, 23.4, 60.9, 68.9, 30.7, 30.7, 41.8, 28.3, 31.4, 29.3, 62.3, 31],
[52.3, 25.2, 63.7, 54.1, 26.2, 58.4, 49.1, 66.2, 27, 64.7, 9, 60.9, 66, 77.3, 58.4, 39.7, 42.1, 27.8, 38.3, 47.2, 35.5, 33.2, 21.5, 22.1, 86.5, 82.3, 73.2, 77.5, 64.3, 65.7,
 63.9, 57.8, 56.6, 57.5, 60.4, float('inf'), 6.1, 21.6, 25.5, 35.1, 30.1, 46.2, 48.1, 44.9, 61.8, 58.3, 59.7, 30, 37.3, 44.3, 44, 57.5, 64.6, 67.5, 65.2, 46.3, 67],
[45.7, 24.5, 57.1, 47.5, 25.5, 51.9, 42.5, 59.7, 26.3, 58.1, 8.3, 54.3, 65.3, 76.6, 57.7, 39, 41.4, 27.1, 37.6, 40.6, 32, 32.5, 20.8, 21.4, 78.4, 74.4, 66.6, 69.5, 57.7,
 59.2, 57.4, 51.2, 50.1, 50.9, 53.8, 6.1, float('inf'), 20.9, 24.8, 34.4, 24, 39.6, 41.5, 38.3, 55.2, 51.7, 53.1, 29.2, 36.6, 43.6, 43.3, 56.7, 58, 60.3, 58.6, 45.6, 60.5],
[36.7, 8.1, 62, 48.3, 5.5, 42.7, 33.5, 50.6, 9.9, 63, 12.6, 45.3, 62.3, 73.7, 60.6, 45.9, 48.3, 34, 43.8, 45.5, 18.4, 18.8, 9.5, 5, 79.6, 75.4, 66.3, 70.5, 57.6, 58.8,
 61.5, 56.1, 55, 51.7, 54.2, 21.6, 20.9, float('inf'), 4.7, 14.3, 13.7, 42.7, 32.4, 43.3, 46.2, 42.7, 44.1, 12.8, 20.2, 23.5, 23.2, 36.6, 48.9, 50.9, 49.2, 25.5, 51.7],
[32.1, 7.3, 58.7, 44.4, 0.85, 38.2, 28.9, 46, 9.1, 58.6, 16.5, 40.7, 60.2, 72.2, 59.8, 47.5, 52.3, 37.9, 48.4, 44.5, 17.6, 18.1, 13.4, 8.8, 75, 70.8, 61.7, 65.9, 52.8,
 54.2, 56.9, 55.2, 58.9, 47.1, 49.6, 25.5, 24.8, 4.7, float('inf'), 10.4, 9.8, 38.1, 27.8, 46.1, 41.6, 38.1, 39.5, 12.1, 19.4, 18.9, 18.6, 32, 44.4, 46.3, 44.6, 20.9, 46.8],
[32.9, 18, 59.6, 45.2, 10.9, 39.1, 29.7, 46.9, 19.8, 59.4, 26.1, 41.6, 72.2, 83.6, 70.5, 58.2, 61.1, 47.5, 57.3, 45.3, 28.3, 28.8, 22.9, 18.4, 75.8, 71.6, 62.5, 66.8, 53.6,
 55, 57.7, 56.1, 59.7, 47.9, 50.4, 35.1, 34.4, 14.3, 10.4, float('inf'), 4.9, 39, 28.6, 47, 42.4, 38.9, 40.3, 22.8, 30.1, 19.7, 18.8, 32.9, 45.2, 47.2, 45.5, 26.2, 47.7],
[32.3, 17.4, 58.9, 44.6, 10.2, 38.4, 29.1, 46.2, 19.1, 58.8, 25.4, 40.9, 71.6, 82.9, 69.9, 57.5, 55.1, 41.7, 52.9, 44.7, 27.6, 28.1, 22.3, 17.8, 75.2, 71, 61.9, 66.1, 53,
 54.4, 57.1, 55.4, 55.9, 47.3, 49.8, 30.1, 24, 13.7, 9.8, 4.9, float('inf'), 38.3, 26.9, 44.2, 41.8, 38.3, 39.7, 22.2, 29.5, 19.1, 18.2, 32.2, 44.6, 46.6, 44.8, 25.6, 47],
[7.9, 45.3, 27.5, 13.2, 37.3, 14, 8.9, 21.8, 47.1, 29, 37.3, 16.5, 96.8, 108, 89.2, 70.5, 73, 58.6, 69.1, 10.6, 55.6, 56.2, 49.5, 45, 44.5, 40.3, 31.2, 35.4, 22.5, 23.7,
 27.3, 24.8, 29.2, 16.6, 19.1, 46.2, 39.6, 42.7, 38.1, 39, 38.3, float('inf'), 11.2, 5.6, 15, 13.9, 15.3, 49.5, 57.4, 19.2, 19.2, 30.5, 19.8, 22.1, 20.4, 50.9, 22.6],
```

```
[5.2, 35, 31.1, 16.7, 26.9, 11.1, 1.9, 19.1, 36.8, 31.7, 39.2, 13.8, 89.2, 101, 87.5, 72.4, 74.9, 60.5, 71, 17.5, 45.4, 45.9, 41.9, 38, 48.1, 43.9, 34.5, 39, 25.8, 27.3,
 30, 28.3, 31.9, 20.1, 22.6, 48.1, 41.5, 32.4, 27.8, 28.6, 26.9, 11.2, float('inf'), 20.6, 16.1, 12.6, 14, 38.8, 46.8, 8.6, 8.5, 19.7, 18.5, 21.5, 19.3, 40.2, 19.9],
[18.7, 46.8, 25.6, 11.3, 45.3, 11.1, 20.2, 30, 48.6, 26.2, 36, 25.3, 95.5, 107, 87.9, 69.2, 74.9, 57.3, 67.8, 6.6, 57.3, 57.7, 48.2, 43.7, 43.8, 39.6, 30.2, 34.7, 21.5,
 22.7, 24.5, 22.9, 23.4, 14.7, 17.2, 44.9, 38.1, 43.3, 46.1, 47, 44.2, 5.6, 20.6, float('inf'), 20.5, 23.3, 24, 51.6, 58.9, 27.2, 27.2, 38.4, 28.9, 31.8, 30.1, 58.9, 31.7],
[10.3, 48.8, 24.6, 14.4, 40.7, 5.6, 13.1, 11.6, 50.6, 25.2, 52.9, 7.4, 103, 114, 101, 86.1, 88.5, 74.2, 84.7, 16.5, 59.2, 59.7, 55.7, 51.9, 39.2, 35, 25.9, 30.1, 16.9,
 18.4, 23.5, 22.7, 27.7, 17, 20.3, 61.8, 55.2, 46.2, 41.6, 42.4, 41.8, 15, 16.1, 20.5, float('inf'), 10.4, 6.2, 54.9, 62.9, 24.7, 24.6, 35.9, 10.4, 11.5, 9.8, 56.3, 13.2],
[6.8, 45.3, 31, 20.7, 37.3, 4.3, 9.6, 9.7, 47.1, 31.6, 49.4, 4.8, 99.6, 111, 97.8, 82.6, 85.1, 70.7, 81.2, 18.9, 55.7, 56.2, 52.2, 48.4, 45.5, 41.3, 32.2, 35.7, 23.2, 24.7,
 28.9, 28.9, 33.3, 22.9, 25.4, 58.3, 51.7, 42.7, 38.1, 38.9, 38.3, 13.9, 12.6, 23.3, 10.4, float('inf'), 5.4, 49.4, 57.4, 19.2, 19.2, 30.4, 8.5, 11.6, 9.7, 50.8, 10.5],
[8.2, 46.7, 27.8, 17.5, 38.6, 1.7, 11, 6.1, 48.5, 28.4, 50.8, 1.5, 101, 112, 99.2, 84, 86.5, 72.1, 82.6, 19.6, 57.1, 57.6, 53.6, 49.8, 42.3, 38.1, 29, 33.2, 20.1,
 21.5, 26.7, 25.8, 32.7, 20.9, 23.4, 59.7, 53.1, 44.1, 39.5, 40.3, 39.7, 15.3, 14, 24, 6.2, 5.4, float('inf'), 50.8, 58.8, 20.6, 20.6, 31.7, 5, 6.9, 5.2, 52.2, 7.8],
[43.4, 4.8, 70.1, 55.7, 11.7, 49.4, 40.2, 57.4, 3, 69.9, 20.9, 52.1, 55.1, 66.4, 53.7, 41.4, 52.1, 37.3, 51.2, 53.9, 11.6, 12.1, 11.7, 10.7, 86.3, 82.1, 73, 77.2, 64.1,
 65.5, 68.2, 64.5, 63.3, 58.4, 60.9, 30, 29.2, 12.8, 12.1, 22.8, 22.2, 49.5, 38.8, 51.6, 54.9, 49.4, 50.8, float('inf'), 11.5, 30.2, 29.9, 43.3, 55.6, 58.7, 56.8, 32.2, 58.2],
[54.2, 12.2, 77.7, 66.5, 20.2, 60.1, 50.9, 64.9, 10.3, 78.7, 28.3, 46.7, 43.6, 55, 44.9, 32.6, 43.2, 28.4, 42.3, 61.2, 8.7, 9.2, 19, 18, 97.1, 92.9, 83.8, 88, 72.3, 76.3,
 78, 72.8, 70.7, 66.4, 68.9, 37.3, 36.6, 20.2, 19.4, 30.1, 29.5, 57.4, 46.8, 58.9, 62.9, 57.4, 58.8, 11.5, float('inf'), 38.2, 38, 51.4, 64.2, 69.4, 67.5, 40.3, 68.9],
[13.2, 26.1, 39.1, 24.8, 18, 19.2, 10, 27.1, 27.9, 39.7, 35.3, 21.8, 80.4, 91.7, 78.6, 66.3, 71.1, 56.7, 67.2, 25.6, 36.5, 37, 32.1, 29.2, 56.1, 51.9, 42.8, 47, 34.1,
 35.3, 38, 36.3, 40, 28.2, 30.7, 44.3, 43.6, 23.5, 18.9, 19.7, 19.1, 19.2, 8.6, 27.2, 24.7, 19.2, 20.6, 30.2, 38.2, float('inf'), 11.2, 16.6, 25.4, 28.3, 26.6, 31.6, 38.2],
[13.2, 25.9, 39.1, 24.7, 17.8, 19.1, 9.9, 26.5, 27.6, 39.7, 35, 21.8, 80.1, 91.4, 78.4, 66, 70.8, 56.4, 66.9, 25.5, 36.3, 36.8, 31.9, 28.9, 56.1, 51.9, 42.8, 47, 33.8,
 35.3, 38, 36.3, 39.9, 28.1, 30.7, 44, 43.3, 23.2, 18.6, 18.8, 18.2, 19.2, 8.5, 27.2, 24.6, 19.2, 20.6, 29.9, 38, 11.2, float('inf'), 22.7, 25.1, 28.3, 26.5, 31.4, 38],
[24.3, 39.2, 51, 36, 31.1, 30.3, 21.1, 38.3, 41, 50.9, 48.5, 32.9, 93.6, 105, 91.8, 79.5, 84.2, 69.9, 80.4, 36.7, 49.7, 50.1, 46.1, 42.3, 65.2, 63, 54, 58.3, 45, 46.4,
 49.1, 47.6, 51.2, 39.4, 41.8, 57.5, 56.7, 36.6, 32, 32.9, 32.2, 30.5, 19.7, 38.4, 35.9, 30.4, 31.7, 43.3, 51.4, 16.6, 22.7, float('inf'), 37.4, 39.7, 38, 44.7, 51.4],
[13.1, 52.1, 32.6, 22.4, 44.1, 6.5, 15.5, 1.3, 53.9, 33.2, 55.6, 3.8, 106, 117, 105, 88.7, 91.3, 77.5, 88, 24.4, 62.6, 62.5, 58.5, 54.6, 47.2, 43, 33.9, 38.1, 24.9, 26.4,
 31.5, 30.7, 37.6, 25.8, 28.3, 64.6, 58, 48.9, 44.4, 45.2, 44.6, 19.8, 18.5, 28.9, 10.4, 8.5, 5, 55.6, 64.2, 25.4, 25.1, 37.4, float('inf'), 3.4, 2.2, 57.7, 64.2],
[15.1, 54.4, 35.5, 25.3, 46.3, 9.4, 18.7, 2.7, 56.4, 36.3, 57.6, 6.3, 109, 119, 107, 91.9, 93.3, 80, 90.5, 27.3, 65, 64.8, 60.8, 57.6, 50, 45.3, 36.7, 41, 28.2, 28.7,
 33.8, 33.8, 40.7, 28.9, 31.4, 67.5, 60.3, 50.9, 46.3, 47.2, 46.6, 22.1, 21.5, 31.8, 11.5, 11.6, 6.9, 58.7, 69.4, 28.3, 28.3, 39.7, 3.4, float('inf'), 1.7, 59.9, 69.4],
[13.3, 52.7, 33.8, 23.5, 44.6, 6.8, 17, 2.1, 54.5, 34.4, 55.9, 4.4, 107, 117, 105, 89.4, 91.6, 78.1, 88.6, 25.6, 63.1, 63.1, 59.1, 55.8, 48.3, 43.5, 34.3, 39.2, 25.5,
 27, 32.1, 31, 38.2, 26.4, 29.3, 65.2, 58.6, 49.2, 44.6, 45.5, 44.8, 20.4, 19.3, 30.1, 9.8, 9.7, 5.2, 56.8, 67.5, 26.6, 26.5, 38, 2.2, 1.7, float('inf'), 58.2, 3.7],
[44.4, 28.2, 71.5, 57.1, 20.1, 50.8, 41.6, 58.8, 29.9, 72.1, 37.3, 53.5, 82.4, 93.7, 80.7, 70.6, 73.1, 58.7, 69.2, 57.2, 38.6, 39.1, 34.2, 31.2, 87.7, 83.5, 74.4, 78.6, 65.5,
 66.9, 69.6, 68, 71.6, 59.8, 62.3, 46.3, 45.6, 25.5, 20.9, 26.2, 25.6, 50.9, 40.2, 58.9, 56.3, 50.8, 52.2, 32.2, 40.3, 31.6, 31.4, 44.7, 57.7, 59.9, 58.2, float('inf'), 59.6],
[15.5, 54.1, 35.4, 25.1, 46, 9.2, 18.4, 2, 55.8, 36, 58.1, 7.3, 108, 120, 107, 91.3, 93.8, 79.5, 89.9, 27.2, 64.5, 65, 61, 57.1, 49.9, 45.7, 36.6, 40.8, 27.7, 29.1, 34.3,
 33.4, 40.3, 28.5, 31, 67, 60.5, 51.7, 46.8, 47.7, 47, 22.6, 19.9, 31.7, 13.2, 10.5, 7.8, 58.2, 68.9, 27.9, 27.9, 39.1, 2.9, 3.5, 3.7, 59.6, float('inf')]
]
```

```
# Label untuk node
labels = ['1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19','20','21','22','23','24','25','26','27','28','29',
          '30','31','32','33','34','35','36','37','38','39','40','41','42','43','44','45','46','47','48','49','50','51','52','53','54','55','56','57']

# Jalankan algoritma Floyd-Warshall
distances, next_node = floyd_warshall_with_labels(graph, labels)

# Rekonstruksi jalur terpendek dari node 16 ke node 57
start_label, end_label = '16','57'
start, end = labels.index(start_label), labels.index(end_label)
path = reconstruct_path_with_labels(next_node, start, end, labels)

# Tampilkan hasil akhir
print("Hasil Akhir:")
print("Matriks Jarak Terpendek (distances):")
for i, row in enumerate(distances):
    print(f"{labels[i]}: {row}")
print("\nJalur Terpendek:")
print(f"Dari {start_label} ke {end_label}: {path}")

# Menampilkan jarak terpendek
if path is not None:
    shortest_distance = distances[labels.index(start_label)][labels.index(end_label)]
    print(f"Jarak Terpendek dari {start_label} ke {end_label}: {shortest_distance}")
else:
    print(f"Tidak ada jalur dari {start_label} ke {end_label}.")
```

Gambar 3. *Source Code Python*

Dalam *Python*, fungsi untuk mengimplementasikan Algoritma Floyd Warshall biasanya menerima sebuah matriks jarak awal yang merepresentasikan hubungan dan bobot antar simpul dalam graf. Fungsi ini kemudian akan mengolah matriks tersebut melalui proses iterasi yang akan dilakukan sebanyak n jumlah simpul. Inti dari fungsi ini adalah membandingkan jalur langsung antara i dan j dengan jalur yang melalui simpul perantara $(i \rightarrow k \rightarrow j)$, lalu memilih jarak terpendek. Penting menggunakan nilai tak hingga $(float('inf'))$ untuk merepresentasikan tidak adanya hubungan langsung atau hubungan dengan dirinya sendiri. Setelah semua iterasi selesai, fungsi ini akan mengembalikan matriks jarak terpendek antar semua pasangan simpul. Fungsi ini sangat berguna karena secara otomatis dapat menangani bobot sisi negatif (selama tidak ada siklus negatif) dan menyediakan solusi lengkap untuk semua jalur terpendek dalam graf.

Sebagai kelanjutan, untuk menggunakan *source code python* Algoritma Floyd Warshall, langkah – langkahnya dimulai dengan memasukkan data jarak antar lokasi dalam bentuk

matriks awal. Matriks ini berisi nilai jarak numerik untuk lokasi yang terhubung langsung dan nilai tak hingga $(float('inf'))$ untuk lokasi yang tidak terhubung atau terhubung dengan dirinya sendiri serta memasukkan kode simpul lokasinya. Selanjutnya, masukkan kode simpul lokasi yang akan dicari. Setelah itu, jalankan *source code* untuk menampilkan hasil perhitungan algoritma. Berdasarkan data matriks awal yang sudah diinput dan dihitung menggunakan iterasi Algoritma Floyd Warshall, *source code* akan menghasilkan informasi rute berupa kode simpul dan jumlah jarak tempuh antar lokasi awal hingga lokasi akhir dalam satuan kilometer. Berikut contoh tampilan hasil dari *source code python* :

```
Jalur Terpendek:
Dari 16 ke 57: ['16', '18', '1', '57']
Jarak Terpendek dari 16 ke 57: 66.7
```

Gambar 4. Hasil *Source Code Python*

Dari contoh hasil *source code python* pada Gambar 4 diperoleh jalur terpendek dari simpul 16 (Air Terjun Curug Bandung) menuju simpul 57 (Pelabuhan Wijayapura) yaitu melalui rute Air Terjun Curug Bandung (16) → Alun – Alun Majenang (18) → Bandar Udara Tunggul Wulung (1) → Pelabuhan Wijayapura (57) dengan informasi total jarak yaitu 66.70 km.

Berdasarkan uraian di atas, proses pencarian rute optimal dapat diimplementasikan melalui perhitungan *source code* khusus berbasis *python* seperti yang ditampilkan pada Gambar 3. Pemilihan komputasi pemrograman *python* dilandasi oleh kemampuannya dalam memproses perhitungan kompleks pada Algoritma Floyd Warshall secara cepat dan akurat, tidak seperti perhitungan manual yang rentan kesalahan. Fungsi utama dalam *source code* ini yaitu menerima matriks berbobot awal, lalu mengolahnya melalui proses iterasi untuk membandingkan rute langsung dengan rute melalui simpul perantara untuk memilih rute terpendek. Hasil dari *source code* khusus berbasis *python* adalah informasi rute berupa kode simpul dan jumlah jarak tempuh antar lokasi awal hingga lokasi akhir dalam satuan kilometer seperti yang ditampilkan pada Gambar 4.

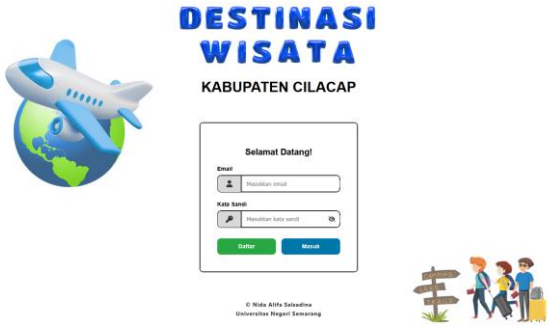
Selanjutnya akan dibuat rancangan aplikasi pada penelitian ini menggunakan metode *waterfall,* berikut tahapan – tahapan perancangan aplikasi pencarian rute optimal antar objek wisata di Kabupaten Cilacap menggunakan Algoritma Floyd Warshall:

Analisis Kebutuhan

Pada tahap awal, peneliti akan mengidentifikasi masalah melalui dua jenis analisis, yaitu analisis kebutuhan dan analisis spesifikasi. Analisis kebutuhan melibatkan studi literatur dan observasi untuk mengumpulkan data dan isu penting yang akan menjadi dasar penentuan fitur sistem yang dirancang. Sementara itu, analisis spesifikasi berfokus pada identifikasi perangkat lunak dan perangkat keras yang diperlukan untuk membangun sistem.

Desain Sistem

Pada tahap desain sistem, peneliti akan menyusun rancangan dan desain sistem yang akan dirancang, termasuk tampilan *inteface.* Tujuannya adalah menciptakan tampilan yang jelas sebagai panduan implementasi, memastikan semua fitur dan fungsi yang dirancang sesuai kebutuhan. Berikut visualisasi tampilan desain yang akan digunakan pada aplikasi pencarian rute optimal antar objek wisata di Kabupaten Cilacap menggunakan perhitungan Algoritma Floyd Warshall.



Gambar 5. Tampilan *Login*

Pada Gambar 5 berisi kolom untuk memasukkan email dan kata sandi akun yang sudah di daftarkan sebelumnya, lalu klik "Masuk" untuk lanjut ke tampilan Beranda. Apabila belum memiliki akun, silahkan klik tombol "Daftar".

Gambar 6. Tampilan *Registrasi*

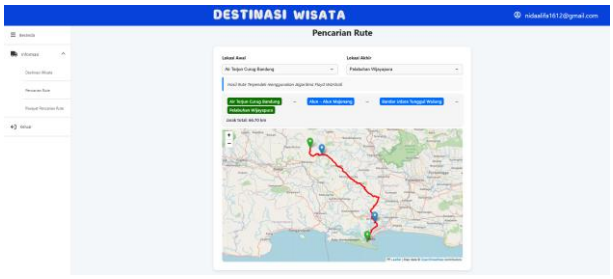
Pada Gambar 6 berisi kolom untuk memasukkan email yang akan didaftarkan dan membuat kata sandi yang akan didaftarkan serta mengkonfirmasi kata sandi untuk memastikan bahwa kata sandi yang akan didaftarkan sudah sesuai, lalu klik "Daftar" untuk mendaftarkan akun. Apabila sudah selesai membuat akun, silahkan klik "Masuk" untuk memastikan bahwa akun yang dibuat sudah terdaftar dan dapat digunakan.

Gambar 7. Tampilan Beranda

Pada Gambar 7 berisi informasi mengenai aplikasi pencarian rute optimal antar destinasi wisata di Kabupaten Cilacap menggunakan Algoritma Floyd Warshall, di mana juga terdapat menu berupa destinasi wisata, pencarian rute dan riwayat pencarian rute serta terdapat tombol untuk keluar aplikasi.

Gambar 8. Tampilan Destinasi Wisata

Pada Gambar 8 menampilkan daftar nama destinasi yang digunakan pada penelitian ini, yang meliputi nama destinasi, bandara, stasiun dan terminal yang terdapat di Kabupaten Cilacap.



Gambar 9. Tampilan Pencarian Rute

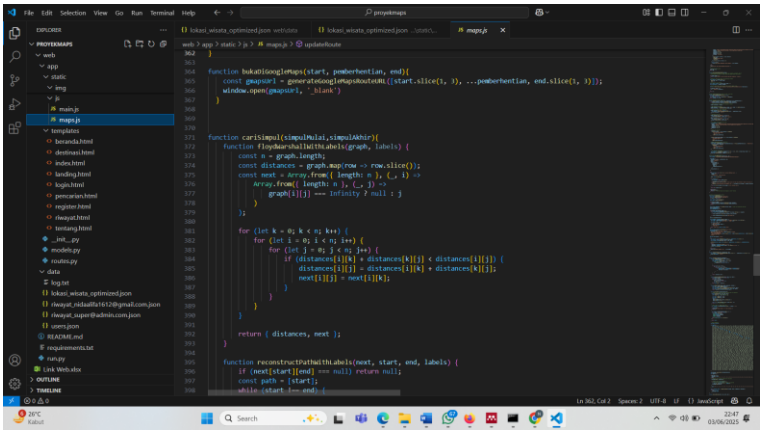
Pada Gambar 9 menampilkan daftar destinasi wisata dalam kolom lokasi awal dan lokasi akhir. Setelah memasukkan lokasi awal dan lokasi akhir, rute dan jarak tempuh yang telah dihitung menggunakan Algoritma Floyd Warshall akan ditampilkan.

Gambar 10. Tampilan Riwayat Pencarian

Pada Gambar 10 menampilkan daftar rute yang telah dicari sebelumnya melalui fitur pencarian rute. Informasi mengenai riwayat pencarian rute berupa lokasi awal, lokasi akhir, waktu mengakses pencarian rute serta jarak yang ditempuh.

Implementasi

Pada tahap implementasi, peneliti akan mengubah desain sistem menjadi *source code* program yang berfungsi. Proses ini melibatkan penulisan kode berdasarkan spesifikasi dan rancangan sebelumnya, dengan tujuan membangun fungsionalitas sistem secara keseluruhan dan menjadikannya aplikasi yang siap digunakan. Berikut merupakan *source code* Algoritma Floyd Warshall yang digunakan pada aplikasi pencarian rute optimal antar objek destinasi wisata di Kabupaten Cilacap.



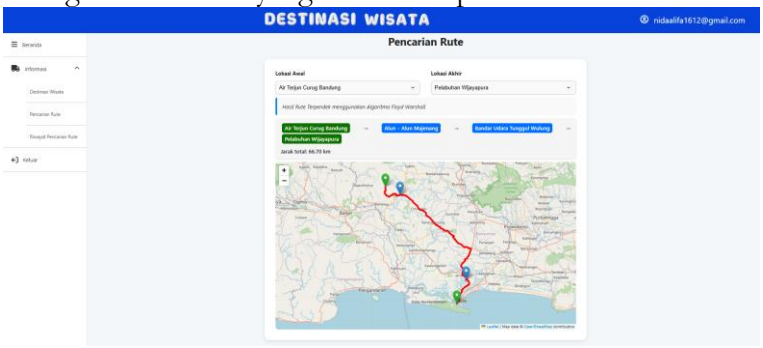
Gambar 11. *Source Code* Aplikasi

Proses implementasi melibatkan penulisan *source code* menggunakan *python,* HTML, PHP, CSS, dan *JavaScript* untuk membangun *interface* pengguna aplikasi. HTML adalah kerangka utama yang digunakan untuk menampilkan halaman web, sedangkan CSS adalah

bahasa pemrograman yang digunakan untuk mengatur tampilan dan tata letak elemen – elemen dalam web yang ditulis dengan HTML (Hidayatullah & Tubagus, 2024). JavaScript merupakan suatu bahasa script yang banyak digunakan dalam dunia teknologi terutama internet (A. O. Sari et al., 2019). Apabila HTML dan CSS hanya mengatur struktur dan tampilan, maka JavaScript untuk menambahkan fungsi yang memungkinkan halaman web merespon aksi pengguna seperti mengklik tombol. Semua pengembangan *source code* dilakukan pada *VS Code*. Tujuannya adalah menciptakan fungsionalitas sistem secara keseluruhan, mengubah rancangan menjadi aplikasi yang siap pakai.

Pengujian Sistem

Pada tahap ini peneliti akan melakukan pengujian dari sistem program yang sudah dibuat untuk memastikan sistem berfungsi sesuai dengan kebutuhan yang telah ditetapkan. Pada tahap pengujian sistem, peneliti akan melakukan serangkaian tes terhadap program yang telah dikembangkan. Tujuannya adalah untuk memastikan bahwa sistem berfungsi dengan benar dan sesuai dengan semua kebutuhan yang telah ditetapkan di awal penelitianBerikut visualisasi tampilan desain yang sudah dilakukan pengujian sistem di mana sistem sudah berfungsi sesuai dengan kebutuhan yang telah ditetapkan.



Gambar 12. Pencarian Rute

Pada Gambar 12 menampilkan visualisasi pada menu pencarian rute. Pengguna dapat memilih lokasi awal dan lokasi akhir dari *dropdown* yang tersedia, kemudian sistem akan menampilkan rute optimal antara dua titik tersebut menggunakan perhitungan Algoritma Floyd Warshall. Pada Gambar 12, rute dari Air Terjun Curug Bandung ke Pelabuhan Wijayapura ditampilkan secara visual di peta, lengkap dengan simpul – simpul yang dilalui yaitu Air Terjun Curug Bandung → Alun – Alun Majenang → Bandar Udara Tunggul Wulung → Pelabuhan Wijayapura dengan informasi total jarak yaitu 66.70 km.

Pemeliharaan

Pada tahap pemeliharaan, peneliti akan mengevaluasi dan menganalisis sistem secara berkelanjutan pasca-implementasi. Ini meliputi pemantauan kinerja, identifikasi masalah, serta pengumpulan umpan balik pengguna. Jika ditemukan kekurangan atau ada kebutuhan baru, peneliti akan melakukan perbaikan dan pembaruan untuk memastikan sistem tetap berfungsi dengan optimal dan relevan seiring waktu.

Berdasarkan uraian di atas, penelitian ini berhasil merancang dan mengimplementasikan aplikasi pencarian rute optimal antar destinasi wisata di Kabupaten Cilacap menggunakan metode *waterfall.* Tahapan dimulai dari analisis kebutuhan untuk menentukan fitur dan spesifikasi sistem, dilanjut dengan desain *interface* yang mencakup tampilan *login* pada Gambar 5*, registrasi* pada Gambar 6*,* beranda pada Gambar 7, *list* destinasi pada Gambar 8, pencarian rute pada Gambar 9, dan riwayat pencarian rute pada Gambar 10. Desain ini menjadi panduan penting untuk memastikan bahwa semua fitur selaras dengan kebutuhan. Desain yang sudah dirancang akan diubah menjadi kode program fungsional pada tahap implementasi. Kode program tersebut dirancang menggunakan *python* sebagai

bahasa *backend* inti, sementara HTML, CSS, dan *JavaScript* untuk membangun *interface* aplikasi, dengan *VS Code* sebagai lingkungan pengembangan utama.

Setelah aplikasi dirancang, tahap pengujian sistem akan dilakukan untuk memverifikasi fungsionalitas aplikasi. Pada Gambar 12 menunjukkan hasil pengujian di menu pencarian rute, di mana pengguna dapat memilih lokasi awal dan akhir dari *dropdown*. Sistem berhasil menampilkan rute optimal dan visualisasinya di peta, lengkap dengan rute yang dilalui dan total jarak tempuh yang sudah dihitung menggunakan Algoritma Floyd Warshall. Contoh rute dari Air Terjun Curug Bandung ke Pelabuhan Wijayapura dengan rute yang dilalui yaitu, Air Terjun Curug Bandung → Alun – Alun Majenang → Bandar Udara Tunggul Wulung → Pelabuhan Wijayapura diperoleh jarak sejauh 66.70 km, ditampilkan secara akurat. Keberhasilan ini menegaskan bahwa sistem berfungsi sesuai dengan kebutuhan yang telah ditetapkan. Terakhir, tahap pemeliharaan dilakukan untuk memastikan sistem tetap relevan melalui evaluasi berkala dan perbaikan berdasarkan umpan balik pengguna.

PENUTUP

Penelitian ini menghasilkan jaringan wisata berbasis graf yang dibuat dari data yang telah dikumpulkan, berupa nama – nama simpul dan jarak penghubung antar simpul. Simpul – simpul tersebut merepresentasikan lokasi – lokasi penting seperti destinasi wisata, bandara, stasiun, dan terminal di Kabupaten Cilacap. Berdasarkan perhitungan yang dilakukan, baik melalui komputasi pemrograman *python* maupun aplikasi, diperoleh contoh rute optimal dari Air Terjun Curug Bandung menuju Pelabuhan Wijayapura dengan jalur yang dilalui, yaitu Air Terjun Curug Bandung → Alun – Alun Majenang → Bandar Udara Tunggul Wulung → Pelabuhan Wijayapura, dengan total jarak tempuh sebesar 66,70 km. Hal ini menunjukkan bahwa aplikasi yang dirancang mampu merepresentasikan hasil perhitungan algoritma dengan tepat serta berfungsi sebagai alat bantu dalam perencanaan perjalanan wisata.

Untuk pengembangan lebih lanjut, disarankan agar aplikasi pencarian rute optimal ini diperluas dengan penambahan simpul data destinasi wisata yang lebih lengkap serta mencakup wilayah yang lebih luas, tidak hanya terbatas pada Kabupaten Cilacap. Selain itu, aplikasi dapat dilengkapi dengan fitur – fitur tambahan seperti estimasi waktu tempuh, estimasi biaya perjalanan, informasi kondisi jalan, informasi mengenai destinasi seperti jam operasional dan tiket masuk, serta integrasi dengan sistem navigasi atau peta digital *real – time* agar lebih interaktif dan bermanfaat bagi pengguna.

REFERENSI

Badan Pusat Statistik. (2022). Luas Wilayah Menurut Kabupaten/Kota, 2019-2021. BPS Provinsi Jawa Tengah. https://jateng.bps.go.id/id/statistics-table/2/NjEzIzI=/luas-wilayah-menurut-kabupaten-kota.html, diakses 7 Maret 2025.

Badan Pusat Statistik. (2024). Jumlah Penduduk Menurut Kabupaten/Kota di Jawa Tengah (Jiwa), 2022-2023. https://jateng.bps.go.id/id/statistics-table/2/NzY2IzI=/jumlah-penduduk-menurut-kabupaten-kota-di-jawa-tengah.html, diakses 7 Maret 2025.

Dinas Pemuda Olahraga dan Pariwisata. (2025). Daya Tarik Wisata Alam dan Buatan. https://disporapar.cilacapkab.go.id/destinasi-pariwisata/, diakses 7 Maret 2025.

Gunawan, F., Fatma, Y., & Mukhtar, H. (2020). Aplikasi Pencarian Rute Terpendek Tempat Wisata Di Kota Pekanbaru Menggunakan Floyd Warshall. Jurnal Fasilkom, 10(1), 54–60.

Hidayatullah, P., & Tubagus, R. (2024). Pemrograman Web.

Kristanto, H. T., Mustakim, A. S., Huda, M. Y. K., & Helilintar, R. (2023). Perancangan Aplikasi Android Untuk Pemilihan Tempat Wisata Di Kota Kediri. Prosiding Seminar Nasional Teknologi Dan Sains, 2, 27–34.

Ma'arif, A. (2020). Buku Ajar Pemrograman Lanjut Bahasa Pemrograman Python Oleh : Alfian Ma ' Arif. Universitas Ahmad Dahlan, 62.

Ningrum, F. W., & Andrasto, T. (2016). Penerapan Algoritma Floyd-Warshall dalam Menentukan Rute Terpendek pada Pemodelan Jaringan Pariwisata di Kota Semarang. Jurnal Teknik Elektro, 8(1), 21–24.

Puspita, S., & Yanto, G. (2024). Jalur terpendek lokasi wisata budaya di kota padang menggunakan algoritma dijkstra. JISTech (Journal of Islamic Science and Technology), 9(1), 9–15.

Sari, A. O., Abdilah, A., & Sunarti. (2019). Web Programming. In Introduction to Bioinformatics and Clinical Scientific Computing.

Septemuryantoro, S. A. (2021). Potensi Desa Wisata sebagai Alternatif Destinasi Wisata New Normal. Media Wisata, 19(2), 186–197.

Siregar, N. N., Suendri, & Triase. (2021). Penerapan Algoritma Floyd Warshall Pada Sistem Informasi Geografis Lokasi Pariwisata Kota Padang Sidempuan Berbasis Android. JISTech (Journal of Islamic Science and Technology), 6(2), 50–60.

Sugiyono. (2013). Metodologi Penelitian Kuantitatif, Kualitatif dan R & D.

Sugiyono. (2022). Metode Penelitian Kualitatif (Untuk penelitian yang bersifat: eksploratif, enterpretif, interaktif dan konstruktif).

Yustita, A. D., Hardiyanti, S. A., & Yuniwati, I. (2018). Algoritma Floyd-Warshall untuk Penentuan Rute Terpendek Model Jaringan Pariwisata Kabupaten Banyuwangi. JMPM: Jurnal Matematika Dan Pendidikan Matematika, 3(2), 137–146.