COIMBATORE INSTITUTE OF TECHNOLOGY

M.SC DATA SCIENCE

21MDS43 – Design And Analysis of Algorithms

DHIVYA K P (71762232010)

KAVYA V V (71762232026)

PRITHIKA J (71762232039)

AMBULANCE ROUTE OPTIMIZATION

Problem Statement:

This project involves developing a Flask web application that helps users find the shortest path to a nearby hospital from their current location using the All-Pairs Shortest Path algorithm (Floyd-Warshall). The application will integrate mapping data to represent various locations and routes within a city. Users will input their current location, and the system will calculate the shortest path to the nearest hospital by leveraging the Floyd-Warshall algorithm, which efficiently computes shortest paths between all pairs of nodes in a weighted graph. The result will be displayed on a map interface, providing users with clear directions to the closest medical facility. This application combines web development, algorithm implementation, and geographic information system (GIS) capabilities to offer a practical and user-friendly navigation tool.

Aim:

The aim of this project is to develop a user-friendly web application that enables users to quickly and accurately determine the shortest path to the nearest hospital from their current location. By utilizing the All-Pairs Shortest Path algorithm (Floyd-Warshall), the application will provide efficient route calculations within a city map, thereby offering an essential tool for individuals in need of immediate medical assistance. This project seeks to combine advanced algorithmic techniques with practical web and mapping technologies to enhance public accessibility to critical healthcare services.

Solution:

Importing Necessary Libraries

The solution begins by importing essential libraries. Flask is used for creating the web application, handling routing, and processing HTTP requests. The request module is utilized for accessing query parameters from incoming HTTP requests, while jsonify helps format the responses as JSON. The render_template function is used to render HTML templates for the web interface. Although the numpy library is imported for numerical operations, it's not explicitly used in the provided code snippet.

Initializing the Flask Application

An instance of the Flask class is created to initialize the web application. This instance will be used to define routes and handle requests.

Setting Up the Graph and Hospital Data

A graph representing the city is set up as an adjacency matrix where each element indicates the distance between two nodes (locations). A value representing infinity is used to denote no direct path between nodes. A list of hospital nodes indicates which locations are hospitals. The number of vertices (nodes) in the graph is also determined.

Implementing the Floyd-Warshall Algorithm

The Floyd-Warshall algorithm is implemented to compute the shortest paths between all pairs of nodes in the graph. This algorithm uses dynamic programming to update the shortest paths by considering each node as an intermediate point. Initially, the distances are the same as the input graph, but they are updated iteratively to reflect the shortest possible paths between all pairs of nodes.

Precomputing Shortest Paths

The Floyd-Warshall function is executed to precompute the shortest paths between all pairs of nodes in the graph. This step ensures that when a user requests the shortest path to a hospital, the application can quickly retrieve the precomputed distance rather than recalculating it each time.

Home Page Route

A route is defined for the home page, which serves an HTML template. This template includes a form where users can input their current location (node) to find the nearest hospital.

Endpoint to Get the Nearest Hospital

A route is created to handle GET requests for finding the nearest hospital. The current location is obtained from the query parameters. The server then calculates the nearest hospital by checking the precomputed shortest paths. It iterates through the list of hospitals, comparing distances to determine the closest one. The result, including the nearest hospital node and the distance, is returned as a JSON response.

Rendering the User Interface

The HTML template includes a form where users can enter their current location. When the form is submitted, a JavaScript function sends an AJAX request to the server to find the nearest hospital. The server processes the request, determines the nearest hospital, and returns the result, which is then displayed on the web page.

Running the Application

Finally, the application is run in debug mode, allowing for real-time updates and easier debugging during development. This setup ensures that the server listens for incoming requests and serves the appropriate responses based on the defined routes and logic

Concepts used:

- Floyd Warshall Algorithm
- All Pair Shortest Path

Technology Stack:

Flask

Code:

App.py

```
from flask import Flask, request, jsonify, render_template


app = Flask(__name__)


# Sample graph represented as an adjacency matrix
# 0 represents no path between nodes, use a high value to represent "infinity" where there is no direct path
INF = float('inf')
graph = [
    [0, 10, INF, INF, INF, 5],
    [10, 0, 3, INF, INF, 2],
    [INF, 3, 0, 1, INF, INF],
    [INF, INF, 1, 0, 4, INF],
    [INF, INF, INF, 4, 0, 6],
    [5, 2, INF, INF, 6, 0]
```

```
]

# List of hospital locations
hospitals = [0, 4]

# Coordinates for each node (latitude, longitude)
coordinates = [
    (40.712776, -74.005974),  # Node 0
    (34.052235, -118.243683),  # Node 1
    (41.878113, -87.629799),  # Node 2
    (29.760427, -95.369804),   # Node 3
    (39.739236, -104.990251),  # Node 4
    (32.715736, -117.161087)   # Node 5
]

# Number of vertices in the graph
V = len(graph)

# Function to run the Floyd-Warshall algorithm
def floyd_warshall(graph):
    dist = list(map(lambda i: list(map(lambda j: j, i)), graph))

    for k in range(V):
        for i in range(V):
            for j in range(V):
                dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])

    return dist
```

```
# Precompute shortest paths between all pairs of nodes
shortest_paths = floyd_warshall(graph)

# Home page
@app.route('/')
def home():
    return render_template('index.html')

# Endpoint to get the nearest hospital
@app.route('/nearest_hospital', methods=['GET'])
def nearest_hospital():
    try:
        # Get current location from query parameters
        current_location = int(request.args.get('current_location'))

        if current_location < 0 or current_location >= V:
            return jsonify({'error': 'Invalid current location'}), 400

        # Find the nearest hospital
        min_distance = INF
        nearest_hospital = -1
        for hospital in hospitals:
            if shortest_paths[current_location][hospital] < min_distance:
                min_distance = shortest_paths[current_location][hospital]
                nearest_hospital = hospital

        if nearest_hospital == -1:
            return jsonify({'error': 'No hospital found'}), 404
```

```
        return jsonify({
            'nearest_hospital': nearest_hospital,
            'distance': min_distance,
            'hospital_coordinates': coordinates[nearest_hospital],
            'current_coordinates': coordinates[current_location]
        })

    except Exception as e:
        return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True)
```

template

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Find Nearest Hospital</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
        body {
            padding-top: 50px;
        }
        .container {
            max-width: 600px;
```

```
    }
  </style>
</head>
<body>
  <div class="container">
    <h1 class="text-center">Find Nearest Hospital</h1>
    <form id="locationForm" class="mt-5">
      <div class="form-group">
        <label for="currentLocation">Current Location (Node):</label>
        <input type="number" class="form-control" id="currentLocation"
placeholder="Enter current location node" required>
      </div>
      <button type="submit" class="btn btn-primary btn-block">Find Nearest
Hospital</button>
    </form>
    <div id="result" class="mt-4"></div>
  </div>

  <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
  <script>
    $(document).ready(function() {
      $('#locationForm').on('submit', function(e) {
        e.preventDefault();
        const currentLocation = $('#currentLocation').val();

        $.ajax({
          url: '/nearest_hospital',
          type: 'GET',
          data: { current_location: currentLocation },
          success: function(response) {
```

```
                $('#result').html(
                    `<div class="alert alert-success">
                        Nearest Hospital: Node ${response.nearest_hospital} <br>
                        Distance: ${response.distance}
                    </div>`
                );
            },
            error: function(xhr) {
                $('#result').html(
                    `<div class="alert alert-danger">
                        Error: ${xhr.responseJSON.error}
                    </div>`
                );
            }
        });
    });
  </script>
</body>
</html>
```

Style.css

```
body {
  padding-top: 50px;
  background-color: #f8f9fa;
  font-family: Arial, sans-serif;
}
```

```
.container {
    max-width: 600px;
    background: #ffffff;
    padding: 30px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h1 {
    color: #007bff;
}

.form-group label {
    font-weight: bold;
}

.btn-primary {
    background-color: #007bff;
    border-color: #007bff;
}

.btn-primary:hover {
    background-color: #0056b3;
    border-color: #004085;
}

.alert {
    margin-top: 20px;
```

```
}
#result .alert-success {
    background-color: #d4edda;
    color: #155724;
    border-color: #c3e6cb;
}
#result .alert-danger {
    background-color: #f8d7da;
    color: #721c24;
    border-color: #f5c6cb;
}
```

OUTPUT :