**CITY NAVIGATION AND EMERGENCY ROUTE PLANNING TOOL**

CPSC 535: Advanced Algorithms (Fall 2023)

Project 2 / GROUP 3

**Team Members:**

Anthony Martinez

James Kim

Anshika Khandelwal

Pravallika Bahadur

Tejashwa Tiwari

Rishitha Bathini

Param Venkat Vivek Kesireddy

**Roles:**

Algorithm Implementation - Tejashwa Tiwari and Anshika Khandelwal

Simulation of Blockages - James kim and Rishitha Bathini

User Interface (UI) and Visualization - Anthony Martinez and Pravallika Bahadur

Data Collection - Param Venkat Vivek Kesireddy

Project Coordinator - Pravallika Bahadur



**Instructor:** Dr Syed Hassan Shah

California State University, Fullerton

Date of Submission - 10/13/2023

# **ABSTRACT**

This project introduces a novel urban navigation tool that employs the Floyd-Warshall algorithm to calculate the shortest routes between key landmarks in advance. This tool's primary objective is to enhance emergency route planning by enabling rapid identification and usage of the quickest routes by emergency responders, even when certain roads are closed or inaccessible. In situations like natural disasters or city-wide emergencies, various routes might be blocked, hindering rapid response. By utilizing the Floyd-Warshall algorithm to create a matrix of the shortest paths, considering alternative routes, this tool speeds up emergency response times. It is especially well suited for this critical role since it can determine the shortest routes between all pairs of sites, ensuring effective and rapid deployment of emergency services under difficult circumstances.

**TABLE OF CONTENTS**

[1. ABSTRACT](#_zi4kq9l9yvj) 2

2. TABLE OF CONTENTS………………………………………………………...…………2

3. [INTRODUCTION](#_f84hj8o3n6wj) 3

4.DATA [COLLECTION, ALGORITHM IMPLEMENTATION, SIMULATION OF BLOCKAGES, AND UI & VISUALIZATION.](#_nd250wwla77h) 4

5[. TESTING](#_md0o7n4stcr2) 19

6. CHALLENGES FACED……………………………………………………………………0

7[. CONCLUSION](#_jehkq6thqmgd) 20

8[. REFERENCES.....................................................................................................................20](#_ym14chgn6apn)

# **INTRODUCTION**

For urban planning and emergency response systems, effective city navigation is crucial, particularly in times of crisis like natural disasters or widespread catastrophes. By improving navigation in cities, we can help to make them safer and more resilient during times of crisis.

The Floyd-Warshall algorithm is a powerful algorithm for finding the shortest paths between all pairs of vertices in a weighted graph. It is a fundamental algorithm in graph theory, and it has many applications in computer science, including network routing, transportation planning, and social network analysis. We can develop more effective emergency response tactics, perhaps saving lives and reducing the effects of catastrophes on our communities, by improving our understanding of this novel strategy.

Essentially, integrating urban planning, emergency response systems, and the Floyd-Warshall algorithm provides a hopeful path toward establishing cities that are safer and more adaptable. By persistently researching and applying these methods, we are shaping a future where our communities can endure hardships and emerge more robust in times of difficulty.

**Problem Statement**: In times of crisis, obstructed roads create obstacles for emergency responders. Creating an essential city navigation tool becomes imperative. This tool must utilize the Floyd-Warshall algorithm to calculate the shortest paths between landmarks, factoring in alternative routes. The key challenge lies in improving response times, allowing responders to navigate swiftly even when roads are blocked.

**Hypothesis:** Utilizing the Floyd-Warshall algorithm within a city navigation tool can greatly improve emergency response systems. By accurately computing the shortest routes between landmarks and exploring alternative paths, this inventive method has the capacity to markedly decrease emergency response durations. Consequently, it facilitates rapid and efficient navigation for responders during crises, resulting in saved lives and mitigating disaster effects on communities.

# **DATA COLLECTION, ALGORITHM IMPLEMENTATION, SIMULATION OF BLOCKAGES, AND UI & VISUALIZATION**

**DATA COLLECTION**

We used Overpass Turbo, which can run overpass API queries and analyze the resulting OSM data interactively on a map. It is a web-based data filtering tool for OpenStreetMap.

With Overpass Turbo, you can run Overpass API queries and analyse the resulting OSM data interactively on a map. There is an integrated wizard, which makes creating queries super easy. The process outlined below, starting with determining the boundary box lat/lon coordinates, serves as a foundational blueprint. This method can be theoretically replicated for any region by following the initial step and integrating the outcomes into our data ingestion framework. Through this seamless integration, the framework generates two essential CSV files.

The first file encompasses a comprehensive collection of nodes, each intricately labelled to signify the presence of specific road features such as stop signs, traffic signals, or the absence thereof. Each node is meticulously linked to its corresponding geographical coordinates. The second file contains a network of edges, intricately interwoven with references to two nodes, accompanied by vital information including street names and weightage. The weight of each edge is meticulously calculated based on the distance between the two nodes it connects.

In the process, a meticulous filtering mechanism is employed. It ensures that only edges where both connecting nodes are present in the node CSV file are considered. Additionally, nodes lacking at least one connecting edge are filtered out, ensuring that the resulting data is both accurate and comprehensive.

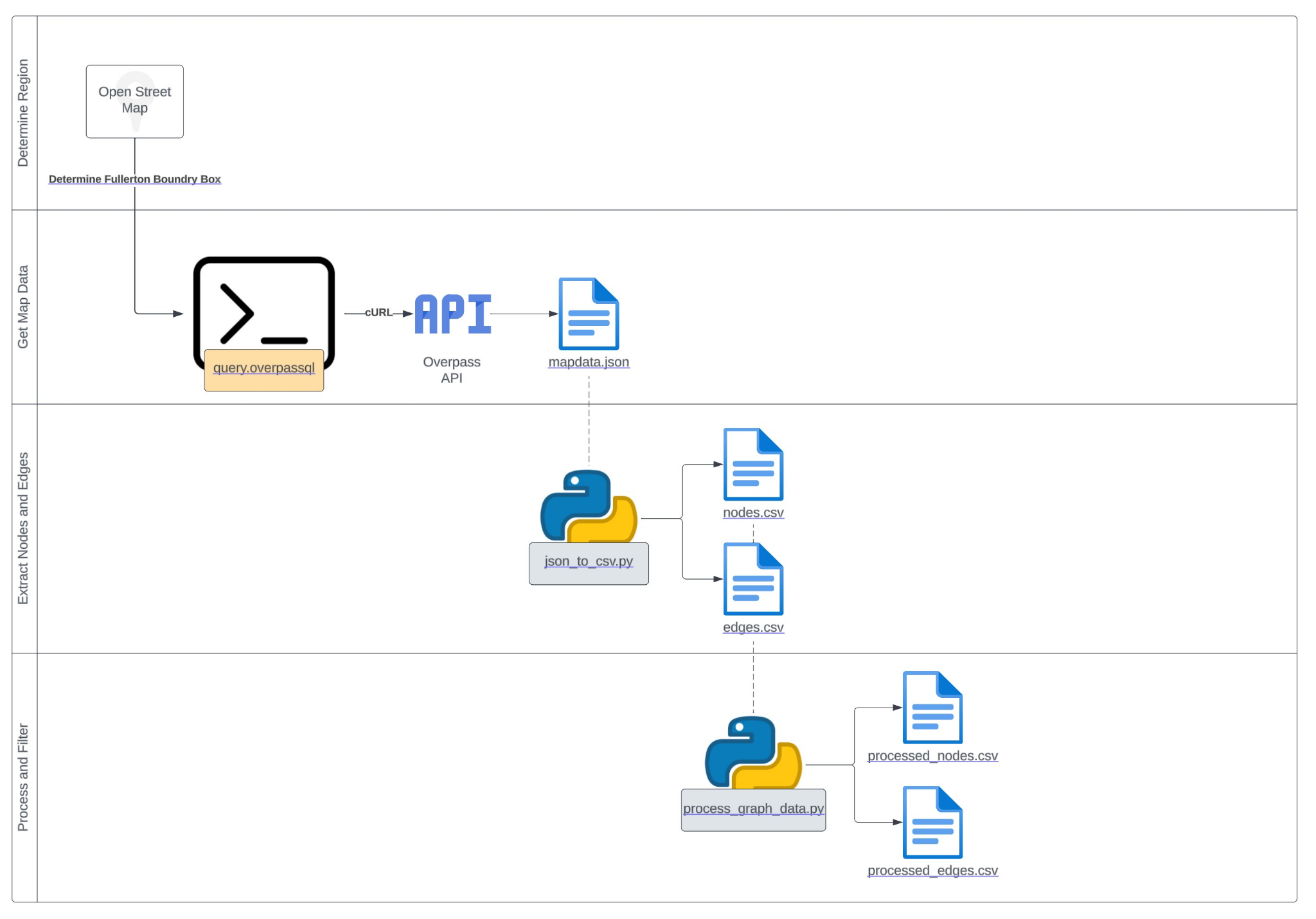


Fig. 1: Overview of Data Collection Part

**ALGORITHM IMPLEMENTATION**

The Floyd-Warshall algorithm is a dynamic programming technique used to identify the shortest pathways in a weighted network between every pair of vertices. Negative edge weights are supported as long as there aren't any negative weight cycles, and it works for both directed and undirected graphs.

An overview of the Floyd-Warshall algorithm is given below:

Create a 2D array dist with size VxV, where V is the graph's vertex count.

Set dist[i][j] to infinity if there is no direct edge from vertex i to vertex j, and initialize dist[i][j] to the weight of the edge from vertex i to vertex j for each pair of vertices (i, j). Additionally, set dist[i][i] for all vertices i to 0.

Consider all pairs of vertices (i, j) for each vertex k from 1 to V, and update dist[i][j] if there is a shorter path from vertex i to vertex j through vertex k. Dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]) is the updating rule.

The dist array will then contain the shortest path distances between all pairs of vertices after repeating step 3 for all vertices k.

**Pseudo code:**

function FloydWarshall(graph):

V = number of vertices in the graph

dist = a VxV array of distances, initialised as described above

for k from 1 to V:

for i from 1 to V:

for j from 1 to V:

dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])

return dist

**Time complexity:** The Floyd-Warshall algorithm has an O(V3) time complexity, where V is the number of graph vertices. Because of this, it is less effective than other algorithms, such as Dijkstra's algorithm, at determining the shortest path between a single pair of vertices, but it has the advantage of being able to determine the shortest paths between all pairs of vertices in a single run, making it appropriate for dense graphs or when you need to know the shortest paths between all pairs of nodes in a graph.

It calculates the shortest routes between landmarks, ensuring swift response despite blocked roads and changing city layouts. The tool adapts dynamically, instantly finding alternative paths for responders, which is crucial for saving lives in crises. Integrating real-time data improves decision-making, guiding responders through the most efficient routes. Overall, this algorithm boosts adaptability, optimises resource allocation, and enhances response efficiency. It becomes integral to building robust emergency response systems for cities, ensuring resilience and effectiveness in times of need.

**SIMULATION OF BLOCKAGES**

Ensuring Backend Stateless Design

One of the foundational principles of this project is to maintain a stateless backend. When users make API calls to block random edges and calculate the shortest path, this action does not affect future API calls. This stateless design ensures that each simulation or action is isolated, free from any side effects, and provides a clean slate for subsequent users.

Enhancing API Flexibility

To make the system more user-friendly, we have opted for a flexible API design. Rather than creating distinct APIs for calculating the shortest paths with or without blocked edges, the system is designed to accept variables. When a client makes a request with variables specifying blocked paths or edges, the backend handles these variables accordingly. This flexibility empowers users to tailor their simulations and route calculations, ultimately leading to a more versatile and responsive system.

By adhering to these design principles and integrating the Floyd-Warshall algorithm, our project strives to provide a city navigation tool that can revolutionize urban planning and emergency response, making our communities safer and more resilient in the face of challenges.

**UI & VISUALIZATION**

HTTP Server: The system operates an HTTP server, which serves as the primary interface for users and handles requests related to location data and route calculations. The server is designed to be easily accessible through a web browser.

CSV Data Integration: To provide location data and route information, the system utilizes data from CSV files. These files contain information about various landmarks, road networks, and geographic coordinates.

Endpoint Handlers: The HTTP server features specialized endpoint handlers for two main types of user requests:

Location Requests: Users can query the system for information about specific locations, landmarks, or geographic points of interest.

Route Requests: Users initiate route planning by selecting a starting point and an ending point on a map.

User Interaction:

Selecting Starting and Ending Points: When a user interacts with the map interface, they click on an arbitrary point to specify their chosen starting location. Subsequently, they click on another point to define the ending location for their route.

Closest Nodes Identification: The Express.js endpoint, upon receiving the user's selected points, determines the closest nodes on the underlying road network to these user-specified locations. These nodes serve as the starting and ending points for the route calculation.

Route Calculation Algorithm: The system employs the Floyd-Warshall algorithm to calculate the shortest routes between the selected nodes, taking into account the entire road network. This ensures that the tool identifies the most efficient path even in complex urban environments with multiple potential routes.

Intermediate Nodes: As part of the route calculation process, the system identifies and returns intermediate nodes. These nodes represent key waypoints along the selected route and are essential for providing detailed directions.

Visualisation of Routes: The system visually presents the calculated route by drawing polylines connecting each node. This gives users a clear visual representation of the path they should follow to reach their destination.

Graph Representation: Additionally, the system plots a graph that displays the nodes and edges corresponding to the selected route. This graph representation helps users grasp the structure of the route and the relationships between different waypoints.

The user interface of this system is designed to be intuitive and interactive, allowing users to easily select starting and ending points on a map. It then seamlessly integrates with the backend algorithm, the Floyd-Warshall algorithm, to provide efficient and visually appealing route suggestions. The combination of user-friendly design and advanced route calculation ensures that users can quickly plan their journeys, including emergency response routes, with confidence.

# **CHALLENGES FACED**

Developing a city navigation tool using the Floyd-Warshall algorithm for emergency route planning presented us with several challenges, such as -

* The first and foremost challenge was data collection. We tried using various APIs and also did the group manually. We finally had to stick to Overpass Turbo to proceed with the project.
* It is essential to guarantee that the implemented algorithm functions properly. To detect and correct any algorithmic mistakes, we went through thorough testing and debugging processes.
* User testing was necessary to gather feedback and make improvements based on user preferences and pain points in order to create a product that people find useful and simple to use.
* Integrating the frontend and backend with real time data to represent the map with shortest path was indeed a challenge.

# [**https://leafletjs.com/examples.html**](https://leafletjs.com/examples.html)

# **CONCLUSION**

In summary, integrating the Floyd-Warshall algorithm into urban navigation tools transforms emergency response. By calculating the best routes, adjusting to dynamic situations, and incorporating live data, it guarantees rapid and efficient actions, preserving lives during emergencies. Its flexibility and effectiveness are crucial for developing robust and responsive urban emergency systems.

**REFERENCES**

1. <https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm>
2. <https://overpass-turbo.eu/>
3. <https://leafletjs.com/examples.html>
4. <https://www.programiz.com/dsa/floyd-warshall-algorithm>
5. <https://mui.com/material-ui/getting-started/templates/>
6. <https://www.npmjs.com/package/concurrently>