DSA_PROJECT

Dijkstra TRAVEL Planner

Project Title:Dijkstra Travel Planner

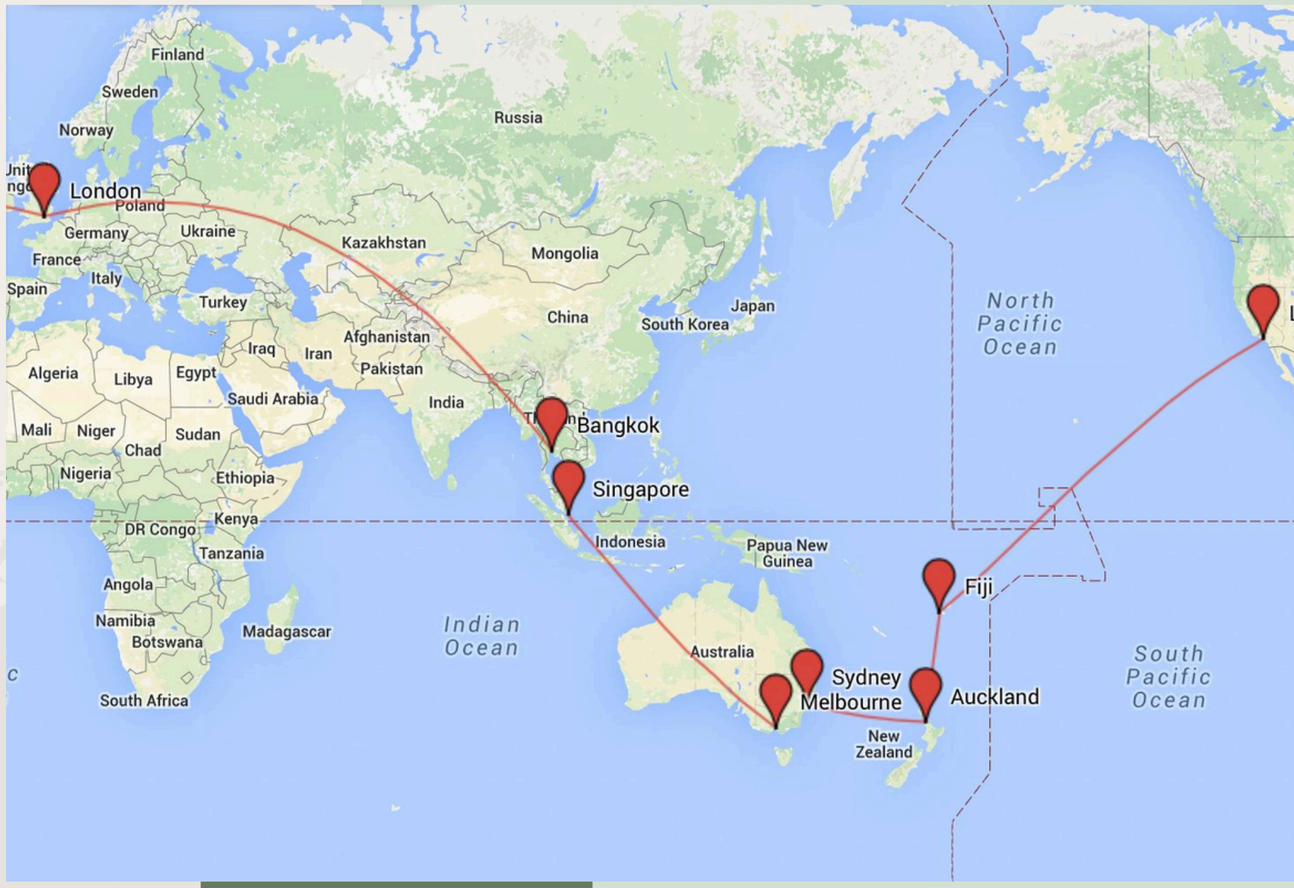
Course Details: CSL2020, DSA, Suchetana Chakraborty

Mentor TA: Arnav Sharma

Team Members: (Rohit_Mourya,B23ES1029)
(Vishwaksen,B23BB1009)

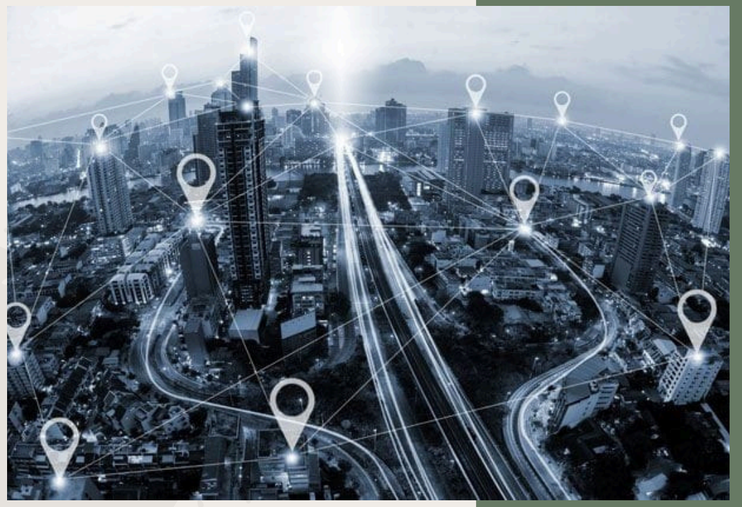
Understanding the Problem

The problem involves computing the optimal travel route between two cities, considering either the ***fastest*** or the ***cheapest*** mode of transportation. Given real-world data including geographical locations of cities and travel details such as cost and time, the challenge is to identify the most efficient route using algorithmic techniques. The **domain** is travel and logistics, and the **scope** includes trip planning applications, logistics optimization, and map-based services. This is an important problem as it underpins key functionalities in apps like Google Maps, Uber, and airline route planners. It's also **challenging** due to the need to manage large datasets, diverse transport options, and optimization of time/cost metrics. A data-driven approach using graph-based algorithms like Dijkstra's ensures a scalable and reliable solution.



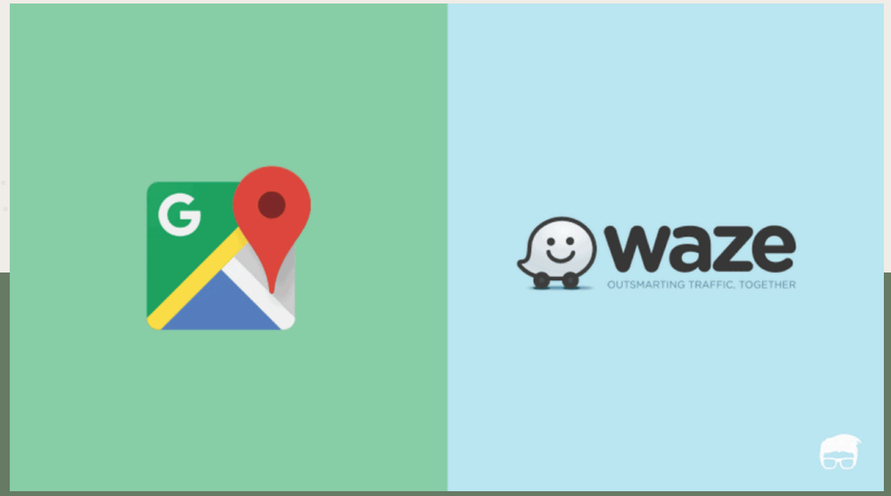
Domain, Relevance & Challenges

Our project lies in the domain of intelligent transportation systems and digital travel assistants. It is highly relevant due to the increased reliance on digital tools for trip planning and navigation. The problem is complex because it requires efficient data parsing, real-time pathfinding, and clear visualization. Unlike static routing, our system must dynamically choose paths based on changing user preferences (fastest or cheapest), making it a dynamic shortest path problem. Traditional brute-force methods are too slow for large networks, but a graph-driven solution using Dijkstra’s algorithm offers a powerful and efficient alternative.



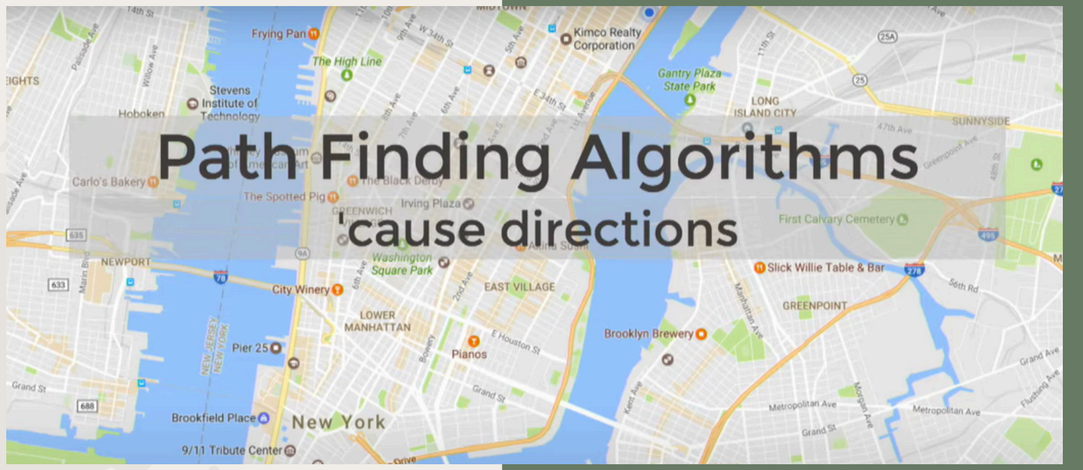
Existing Solutions & Gaps

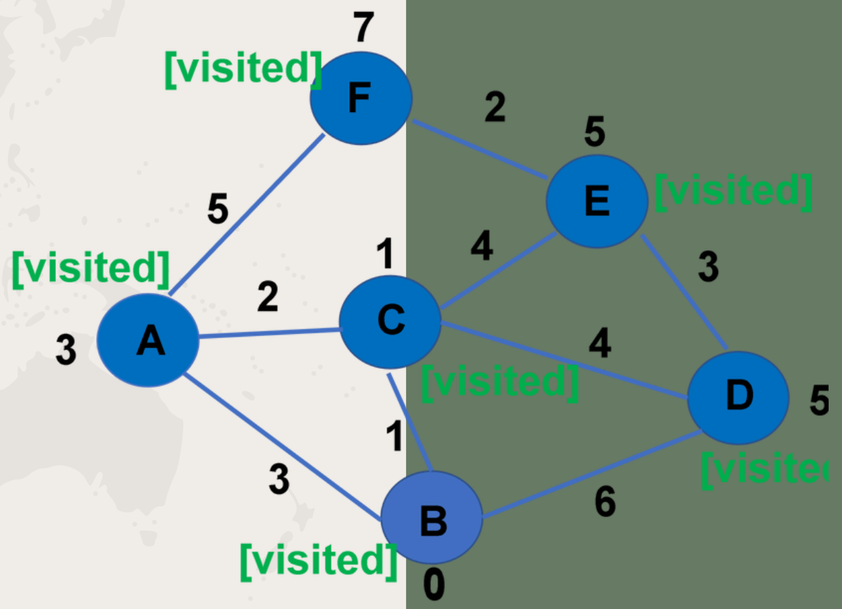
route-finding solutions are commonly implemented in navigation apps like **Google Maps and Waze**. These applications use advanced pathfinding algorithms like A*, Dijkstra's, and even machine learning for traffic prediction. However, many academic or lightweight travel tools lack such functionality. Existing open-source tools often lack customization (e.g., user-selectable metrics like cheapest vs. fastest). Research papers emphasize the efficiency of Dijkstra's algorithm on static graphs but also reveal limitations in integrating route visualization. Our project bridges this gap by integrating a classical algorithm with modern visualization (Google Maps) for user-friendly output, focusing specifically on the algorithmic and data structure aspect.



Proposed Idea

Our project lies in the domain of intelligent transportation systems and digital travel assistants. It is highly relevant due to the increased reliance on digital tools for trip planning and navigation. The problem is complex because it requires efficient **data parsing, real-time pathfinding, and clear visualization**. Unlike static routing, our system must dynamically choose paths based on changing user preferences (fastest or cheapest), making it a dynamic shortest path problem. Traditional brute-force methods are too slow for large networks, but a graph-driven solution using Dijkstra's algorithm offers a powerful and efficient alternative.





System Design & Execution Flow

The system is designed in modular C++ files. FileOperations handles parsing of CSV files. Location and Route classes model the graph's vertices and edges. GraphFunctions implements Dijkstra’s algorithm and supports utility functions like path extraction and weight calculation. Main.cpp coordinates execution. After computation, the route is visualized via output.html. This design ensures code reusability, clarity, and logical separation of responsibilities. The use of STL structures like priority_queue, stack, and vector helps maintain performance and simplicity.

cities.csv

- Contains country name, capital city, latitude, and longitude.
- Each entry is used to create a Location object representing a graph node.
- Coordinates are essential for map visualization and spatial analysis.

routes.csv

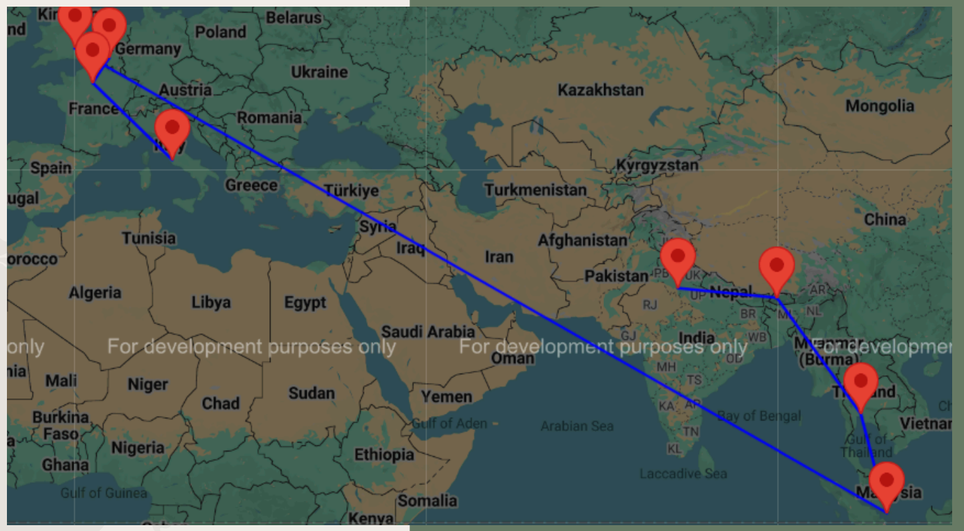
- Includes origin city, destination city, transport type, travel time, and travel cost.
- Each entry creates a Route object, representing a weighted edge in the graph.
- The transport type can be bus, train, or plane.
- Edge weights depend on user preference: fastest (time) or cheapest (cost).

Results & Demonstration

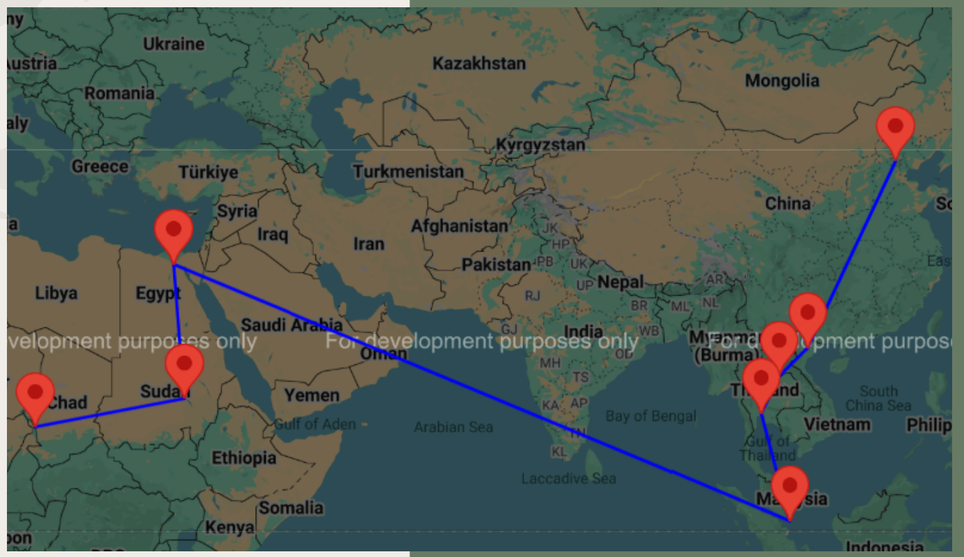
Upon successful compilation and input, the user is prompted to enter origin, destination, and travel preference. The system computes the optimal route and generates a dynamic HTML file. The route is shown on an interactive Google Map with clickable markers displaying city names and routes with details like travel time, cost, and mode. Example result:India to Italy . This visual representation confirms that the computed route respects the user’s preference and validates the underlying algorithm.

GITHUB:https://github.com/rohitkumarmourya-maker/DSA_Project.git

India to Italy (fastest)



Chad to China (cheapest)



Summary & Insights

Through this project, we successfully implemented Dijkstra's algorithm in a real-world use case, combining it with **C++ data structures and Google Maps visualization. We learned to handle file parsing, object-oriented design, and graph algorithms at scale**. The system effectively demonstrates how core algorithmic concepts can be integrated with modern UI tools. This project forms a solid base for future enhancements like adding real-time traffic, weather data, or supporting round trips.

Individual Contributions

Content:

- [**Vishwaksen**]: Core implementation of Dijkstra's algorithm, HTML generation logic, output design Route and Location class logic..
- [**Rohit Mourya**]: Core implementation of Dijkstra's algorithm, File parsing and data model setup, Route and Location class logic.

Acknowledgements

We would like to acknowledge the guidance of our instructor **Suchetana Chakraborty** and our TA **Arnav Sharma** for their support and feedback. We also thank the developers of the C++ STL for robust and efficient data structures, and Google for providing access to the Google Maps JavaScript API. We took care to avoid plagiarism by writing original code, citing inspirations properly, and developing this project from scratch.

Thank you

Presented by:
[Rohit_Mourya] (B23ES1029)
[Vishwaksen] (B23BB1009)