Project’s Report: Map Navigator

By Mahmoud Refaie, Noor Emam, and Ahmed Farid

**Department of Computer Science and Engineering, AUC**

1. **Introduction**

While viewing the list of projects, we decided to do the Map Navigation application as we believed it would be challenging and interesting. Additionally, we believe this particular project is beneficial to us, experience-wise, as it exposes us to real-world applications of our knowledge. We perceived the project as a perfect application to the course material that we had studied throughout the whole semester. The Map Navigator that we have implemented aims to find the shortest path between two cities. This data structure is being used in many well-known applications, such as Google Maps. Our group had worked hard to implement the most efficient code to such an application, also, we designed the code to be as clear and concise as we could so that group members would understand it and edit it in case there was no communication available at the time. Throughout this paper, a detailed report about the project will be provided, including the methodology, our analysis, and some of the obstacles we faced.

1. **Problem Definition**

This application aims to find the shortest possible path between two cities using some algorithms that are being used nowadays in some maps such as Google maps and Apple maps. The code provided serves as an implementation for analyzing a network of interconnected cities, employing graph theory principles and algorithms to ascertain optimal routes and edge weights. Upon reading city data from an external file, the program constructs a dynamic adjacency matrix to represent the relationships between cities and the associated edge weights. Noteworthy functions within the codebase include mechanisms for determining the presence of a city on the map, inserting cities with their corresponding indices, and populating the adjacency matrix with edge weights. The main function establishes a map to manage city names and indices, reads data from a designated file, initializes the adjacency matrix, and subsequently prompts user input to identify the shortest path between user-specified starting and ending cities. This implementation underscores the practical application of graph theory concepts, providing a framework for the analysis of city networks. By utilizing adjacency matrices and incorporating user input, the program offers a versatile tool for navigating complex urban infrastructures and optimizing travel routes. The underlying algorithmic structure provides a foundation for further exploration and extension, showcasing the significance of graph-based methodologies in solving real-world problems related to city connectivity and route optimization.

1. **Methodology**

The methodology adopted for this study entails the strategic implementation of Dijkstra's algorithm to tackle the intricate problem of identifying the shortest route between entities within a geographical context. To commence, pertinent geographical data is systematically collected from an external source, and carefully organized into a structured format. Two different external files were tested, Russian and Indian cities files. The subsequent processing phase involves the utilization of suitable data structures to effectively manage entities and encapsulate their interconnections, incorporating associated weights that represent distances. User interaction forms a crucial aspect of the methodology, as the algorithm prompts users to input the starting and ending points for which the shortest path is to be determined. The crux of the algorithm lies in the execution of Dijkstra's algorithm, seamlessly embedded within the code. This algorithm systematically assesses and calculates the optimal route by iteratively evaluating the distances between interconnected entities, thus ensuring the identification of the most efficient path. In addition to the algorithmic intricacies, the methodology emphasizes the implementation of robust mechanisms to handle potential errors and exceptions that may arise during code execution. This proactive approach contributes to the reliability and accuracy of the results obtained. Furthermore, the methodology places due importance on the documentation of the codebase. Clear and comprehensive documentation not only facilitates a better understanding of the code for future reference but also enhances the overall usability of the solution. This meticulous approach ensures a systematic and thorough exploration of the problem space, culminating in a sophisticated and effective solution for geographical pathfinding, with the utilization of Dijkstra's algorithm as a pivotal component in the computational framework.

**6. Data Specifications**

The input data utilized in this study originates from an external source and is structured to represent geographical information. The dataset, encapsulated in a file named "CitiesData.txt," comprises pairs of entities, such as cities, along with corresponding weights that denote the distances between them. The format of each line includes the names of two entities and the numerical value representing the distance or weight associated with their connection. The data is organized systematically to facilitate the processing and application of pathfinding algorithms for determining the shortest route between specified entities within a geographical context.

**6. Experimental Results**

Upon processing the input data using the implemented methodology and Dijkstra's algorithm, the experimental results are presented in a format that conveys the calculated shortest paths between specified starting and ending points. These results can be expressed textually, providing insights into the identified optimal routes, or presented in tabular form to offer a structured overview. Additionally, graphical representations, such as visualizations of the geographical network and the paths traversed, may be employed to enhance the interpretability of the experimental outcomes.

**7. Analysis and Critique**

The analysis and critique section delves into an evaluation of both the output results and the methodology/algorithms employed. The assessment considers factors such as the efficiency of the algorithm in identifying optimal paths, the accuracy of the calculated distances, and the robustness of the code in handling different scenarios. The complexity of Dijkstra’s algorithm is noticed to be O(V^2), V representing the number of cities. Critiques may encompass areas for potential improvement, the impact of any assumptions made during the study, and reflections on the suitability of the chosen algorithm for the specific geographical pathfinding problem under consideration.

**8. Conclusion**

In conclusion, this study has explored the problem of geographical pathfinding, presenting a methodology that leverages Dijkstra's algorithm to calculate the shortest paths between entities based on input data from "CitiesData.txt." The experimental results provide valuable insights into the efficiency and effectiveness of the implemented solution. The analysis and critique offer a reflective examination of the methodology, highlighting strengths, potential limitations, and areas for further refinement. This study contributes to the field of geographical pathfinding, providing a foundation for future research and practical applications in various domains.

**9. References**

* Bari, A. (2018, February 9). *3.6 dijkstra algorithm - single source shortest path - greedy method*. YouTube. <https://www.youtube.com/watch?v=XB4MIexjvY0>
* GeeksforGeeks. (2023, November 23). *Find shortest paths from source to all vertices using Dijkstra’s algorithm*. GeeksforGeeks. https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
* Dr. Gonied’s Slides.