Uber Nearest Vehicle Technique

November 5, 2023

Harsh Patidar (2022MCB1265) ,
Jigisha Arora (2022MCB1267)

Instructor:
Dr. Anil Shukla

Teaching Assistant:
Mr Law Kumar

Summary: This project is a C++ implementation of a city transportation arrangement system that leverages graph theory and shortest path algorithms to efficiently connect users with the nearest available cab/auto-rickshaw service. The city is represented as a weighted graph(map of the city is given below), with nodes corresponding to locations and edges indicating distances between them. Users input their current location as any node/vertex of the graph and desired destination, and the system employs Floyd-Warshall's algorithm to find the shortest path in the graph. It maintains a list of available cabs at different nodes, determining the closest one to the user. The system calculates the travel distance and estimates the cost based on a fixed rate. It then outputs the nearest cab, travel distance, and cost to the user, facilitating efficient and cost-effective urban transportation. Please note that this is a simplified model, and a complete system would require accurate geographical data and additional features for a real-world implementation. Remember,this code will work only on this city, if you want to change the city then bring a new map and feed its edges in this code.

1. Introduction

The City Transportation System project is an innovative application of graph theory and algorithmic optimization to provide a user-friendly solution for urban travelling arrangements. In urban environments, efficient transportation is a vital component of daily life, and this project aims to streamline the process of finding the nearest available cab (in this case, auto-rickshaws) for users. By leveraging graph representations of the city's road network and employing advanced algorithms such as Johnson's and Floyd-Warshall, this system efficiently calculates the shortest travel routes and estimates the cost for users. This project simplifies the often complex task of navigating through a city by connecting users with the closest available cabs, offering a more convenient and cost-effective commuting experience. The ability to find the nearest auto-rickshaw and calculate travel costs can significantly enhance the quality of urban transportation for residents and visitors alike.

2. Purpose and Scope

The primary purpose of this project is to provide a user-friendly solution for urban transportation, specifically connecting users with the nearest available auto-rickshaws. By employing advanced algorithms, the system reduces travel time and costs for users, contributing to a more efficient and cost-effective commuting experience. This project focuses on a simplified model of arrangements of urban transportation, with a fixed cab rate assumption. While this is a basic representation, it demonstrates the core concepts of finding the nearest cab and estimating the cost based on distance.

3. Data Structures and Algorithms Involved

The Floyd-Warshall algorithm: The Floyd-Warshall algorithm is a dynamic programming algorithm used to find the shortest paths between all pairs of vertices in a weighted, directed graph. It works by iteratively updating the shortest path distances between all pairs of vertices until the shortest paths are determined. Here's a note on the working of the Floyd-Warshall algorithm:

Mapping: Mapping plays a significant role in organizing and managing data. Mapping is utilized in the project to associate auto-rickshaws with their respective node locations, facilitating efficient tracking and coordination within the system. This enables the precise identification and management of auto-rickshaws as they navigate through the network of nodes, streamlining their movements and services.

Graph: The graph data structure is fundamental to your project, serving as the canvas upon which the Floyd-Warshall's algorithm operates. Graphs are a versatile representation of various relationships and connections, consisting of vertices (places in city) and edges (roads connecting edges). In this project, graph is used to model and visualize the city and provide a framework for applying the algorithm efficiently. The choice of graph representation is adjacency matrix (although this can be done using an adjacency list).

Priority Queue: A priority queue is another key data structure employed in our project. It supports efficient access to the element with the highest priority(nearest vehicle have more priority) . In the context of this project, priority queues are used arrange the auto-rickshaw in the ascending order of distance. and if the user don't want to go with the nearest rickshaw. He/She has the option to go with the next nearest auto-rickshaw.

4. Algorithm working and analysis

4.1. How Floyd-Warshall's Algorithm works :-

Here's a brief note on the working of Floyd-Warshall's algorithm:

4.1.1 Initialization:

The algorithm starts by initializing a 2D array dist of size VxV (where V is the number of vertices in the graph) to represent the shortest path distances between all pairs of vertices. Initially, dist[i][j] is set to the weight of the edge from vertex i to vertex j if there's an edge, or it is set to infinity if there's no direct edge. Also, dist[i][i] is set to 0 for all vertices since the distance from a vertex to itself is always zero.

4.1.2 Iterative Updates:

The core of the algorithm involves a series of iterative updates of the dist matrix. It repeatedly considers all vertices k as potential intermediate vertices in the paths from vertex i to vertex j. For each pair of vertices i and j, the algorithm checks whether there's a shorter path from i to j through vertex k compared to the current distance dist[i][j].

4.1.3 Repeated Iterations:

The algorithm repeats the update step for all vertices as intermediate vertices (k) in a nested loop for each pair of vertices (i, j). This process is repeated until no further improvements can be made to the dist matrix. When no changes are made in an iteration, the algorithm terminates.

4.1.4 Detection of Negative Cycles:

One important feature of the Floyd-Warshall algorithm is that it can detect the presence of negative cycles in the graph. If, after the algorithm terminates, there are vertices for which dist[i][i] is still less than zero, it indicates the existence of a negative cycle in the graph.

4.1.5 Reconstructing Shortest Paths:

Once the algorithm has computed the shortest path distances, you can also reconstruct the actual shortest paths themselves by tracing back through the dist matrix to find the intermediate vertices used to reach the shortest paths.

4.2. Advantages of using Floyd-Warshall's algorithm:

4.2.1 The Floyd-Warshall algorithm is versatile and can handle graphs with both positive and negative edge weights.

4.2.2 It determines the shortest paths between all pairs of vertices in a single execution.

4.2.3 It can detect the presence of negative cycles.

4.3. Limitations of using Floyd-Warshall's algorithm:

4.3.1 The algorithm has a time complexity of O(V(cube)), making it less efficient for large graphs.

4.3.2 It is not as efficient as other algorithms, such as Dijkstra's algorithm, for finding the shortest path from one source vertex to all other vertices when the problem only requires a single-source shortest path.

5. Complexity Analysis:

5.1. Time Complexity:

The time complexity of the Floyd-Warshall algorithm is O(V(cube)), where V is the number of vertices in the graph. The triple nested loop iterating through all vertices, combined with the V iterations, results in O(V(cube)) time complexity.(for more details refer the code).

5.2. Space Complexity:

The space complexity of the Floyd-Warshall algorithm is O(V(square)), which represents the space required to store the distance matrix for the shortest path distances.
2D Array dist: To represent the shortest path distances between all pairs of vertices, the algorithm uses a 2D array dist of size V*V, where V is the number of vertices. This matrix requires O(V*V) space.

6. City Map:

The map of the city is given at the back. In this city of 20 nodes, there are 5 auto-rickshaws(not fixed) at different positions(not-fixed).
And the user will be calling these rickshaws using this.
Here, if the user is not comfortable to go with with the driver.He/She have the option for second nearest auto-rickshaw.

7. Bibliography:

7.1. Introduction to Algorithms

Introduction to Algorithms by Thomas H. Cormen (Author), Charles E. Leiserson (Author), Ronald L. Rivest (Author), Clifford Stein (Author) Page no:-693 (3rd Edition).

7.2. javatpoint

www.javatpoint.com

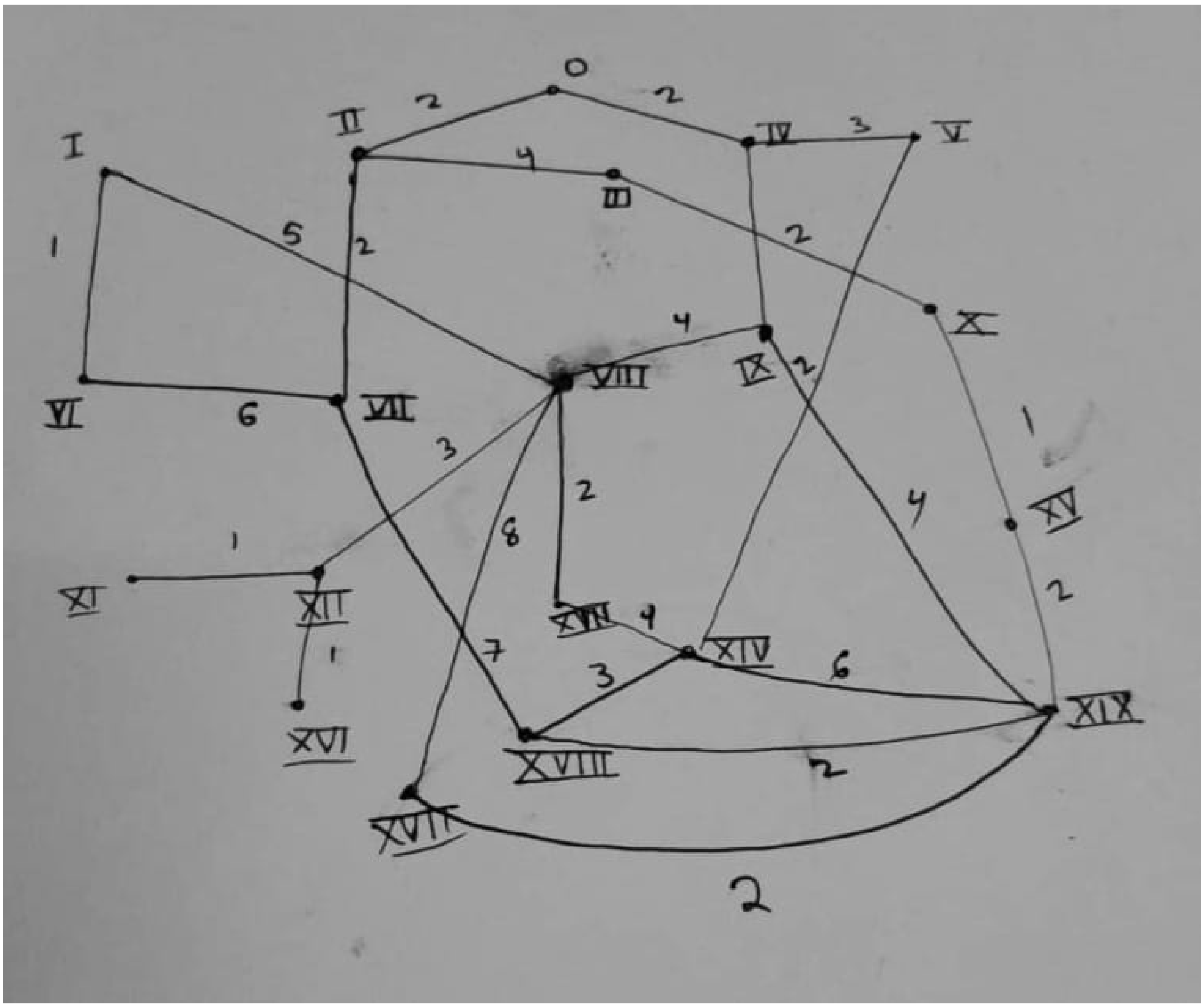

Figure 1: Map of the city

8. Conclusion

The "City Transportation System" project utilizes advanced algorithms like Floyd-Warshall and Johnson's to simplify urban travel by connecting users with the nearest auto-rickshaws, improving commuting efficiency and cost-effectiveness. The project's core focus is on finding the nearest auto-rickshaw and estimating costs, offering a user-friendly solution for urban transportation, enhancing residents' and visitors' commuting experience.