ROAD NAVIGATION SYSTEM AND NETWORKING

Submitted by:

- Anish -106124017
- Dheeraj Krishna -106124033
- K Gnandeep-106124057

Submitted to:

- Dr. Kamalika Bhattacharjee
- Computer science and engineering

Course: Combinatrics and graph theory

Institution: National institute of technology, Trichy

Date of Submission: 23-10-2025

Introduction

Road navigation systems are a critical component of modern transportation, urban planning, and logistics management. They provide optimal routes, monitor connectivity, and allow planning of efficient road networks. With increasing traffic and urbanization, designing a navigation system that can efficiently compute reachability, shortest paths, and minimum-cost road networks is vital.

In this project, we implement a **Road Navigation System** using **graph algorithms**:

- **BFS Algorithm :** Check if a destination is reachable from a source.
- **Dijkstra's Algorithm** : for shortest path computation
- **Prim's Algorithm** : for constructing a minimum spanning road network

The system models **cities or intersections as nodes** and **roads as weighted edges**, with the weight representing distance or travel cost. This allows to

visualize(understand) with the real-world navigation scenarios for testing and analysis.

Objectives

The main objectives of this project are:

1. Represent a road network as a **weighted undirected graph(**.
2. We will implement BFS to test reachability between cities.
3. We will be applying **Dijkstra's algorithm** to find the **shortest path and distance** between two locations.
4. We will be applying **Prim's algorithm** to construct a **minimum-cost road network** connecting all cities.
5. Provide clear, structured outputs for all operations.
6. We will be designing a **menu-driven system** for easy user interaction and multiple test cases.
7. It will be based on travel cost or distancer travelled.

Graph Representation

A graph $G(V, E)$is written as function of V ,E in code. We will input the number of cities(nodes, V) and number of edges (distance/cost, E) and create a road network:

- **Vertices (V)**: Cities or
- **Edges (E)**: Roads with distance or cost as weight

Graph representations:

- **Adjacency Matrix**: 2D array storing edge weights
- **Adjacency List**: Efficient for sparse graphs

Breadth-First Search (BFS)

BFS is a **graph traversal algorithm** that traverses nodes **level by level**. It uses a **queue** to visit all adjacent vertices before moving to the next level.

Applications in this project:

- Test if all cities are reachable(from given source).
- Determine paths in unweighted scenarios .

Algorithm Steps:

1. Initialize a queue with the source vertex.
2. Mark the source as visited.
3. While the queue is not empty:
    - Dequeue a vertex, visit it
    - Enqueue all unvisited adjacent vertices

Time Complexity: $O(V + E)$

Dijkstra’s Algorithm

Dijkstra’s Algorithm finds the **shortest path** from a **source** to all other vertices in a **weighted graph with non-negative edges**.

Algorithm Steps:

1. Assign distance = 0 to source; ∞ to all others.
2. Insert the source into a priority queue.
3. While the queue is not empty:
    - Extract the vertex with the minimum distance
    - Update distances of all adjacent vertices if a shorter path is found

Applications:

- Finding shortest routes between cities
- Travel distance and cost optimization

Time Complexity: $O((V+E)\log V)$using a priority queue

Prim’s Algorithm

Prim’s Algorithm constructs a **Minimum Spanning Tree (MST)** — a subset of edges connecting all vertices with **minimum total weight**.

Algorithm Steps:

1. Start with an arbitrary vertex.
2. Add the minimum weight edge connecting MST to a new vertex.
3. Repeat until all vertices are included

Applications:

- Designing low-cost road networks
- Planning telecommunication or power grids

Time Complexity: $O(E\log V)$

Algorithms and Pseudocode

BFS Pseudocode

```
BFS(Graph, source):
  create a queue Q
  mark source as visited and enqueue it
  while Q is not empty:
    node = Q.dequeue()
  for each neighbor of node:
      if neighbor not visited:
        mark visited
    Q.enqueue(neighbor)
```

Dijkstra Pseudocode

```
Dijkstra(Graph, source):
  initialize distances to all vertices as INFINITY
distance[source] = 0
create a priority queue PQ and insert source
while PQ is not empty:
  u = vertex with minimum distance in PQ
  for each neighbor v of u:
    if distance[u] + weight(u,v) < distance[v]:
      distance[v] = distance[u] + weight(u,v)
      update PQ
```

Prim Pseudocode

```
Prim(Graph):
  choose any vertex as start
  initialize MST set
  while MST does not include all vertices:
    select edge with minimum weight connecting MST to new vertex
    add edge and vertex to MST
```

Implementation Details

- **Data Structures:** adjacency list, queue, min-heap
- **Menu Options:**
    1. Reachability Test (BFS)
    2. Shortest Path (Dijkstra)
    3. Minimum Spanning Road Network (Prim)

Input Format:

- Number of cities
- Road connections with distances
- Source and destination for Dijkstra

Examples :

Input Example:

Cities: 5

Roads: A-B(2), A-C(4), B-C(1), B-D(7), C-E(3), D-E(2)

Source: A

Destination: D

Output:

- **BFS Reachability:** A, B, C, D, E
- **Dijkstra Shortest Path:** A -> B -> C -> E -> D, Distance = 8
- **Prim MST:** B-C, C-E, A-B, E-D, Total Cost = 8

Results and Discussion

Function	**Algorithm**	**Purpose**	**Output**
Connectivity Test	BFS	Reachable cities	List of all reachable nodes
Shortest Path	Dijkstra	Min distance	Path + Distance
Minimum Road Network	Prim	Min cost network	MST edges + Total cost

Observations:

- BFS quickly identifies reachability
- Dijkstra provides accurate shortest paths
- Prim efficiently constructs a minimum-cost network

Applications

- GPS Navigation apps (Google Maps, Waze)
- Transport network planning
- Telecommunication and power grid design
- Logistics and delivery optimization
- Emergency and disaster management

Challenges and Future Scope

Challenges:

- Dynamic traffic conditions

- Real-time updates
- Large-scale city networks

Future Scope:

- A* algorithm integration for heuristic pathfinding
- GUI-based visualization
- Live traffic data integration
- Optimization for large datasets

Conclusion

The project successfully demonstrates the application of **BFS, Dijkstra, and Prim algorithms** in a Road Navigation System. It effectively tests reachability, finds shortest paths, and constructs minimum-cost road networks. These algorithms form the foundation for real-world navigation, urban planning, and network design.

References

1. Thomas H. Cormen et al., *Introduction to Algorithms*, MIT Press
2. Mark Allen Weiss, *Data Structures and Algorithm Analysis in C++*
3. GeeksforGeeks – Graph Algorithms: https://www.geeksforgeeks.org/graph-data-structure-and-algorithms