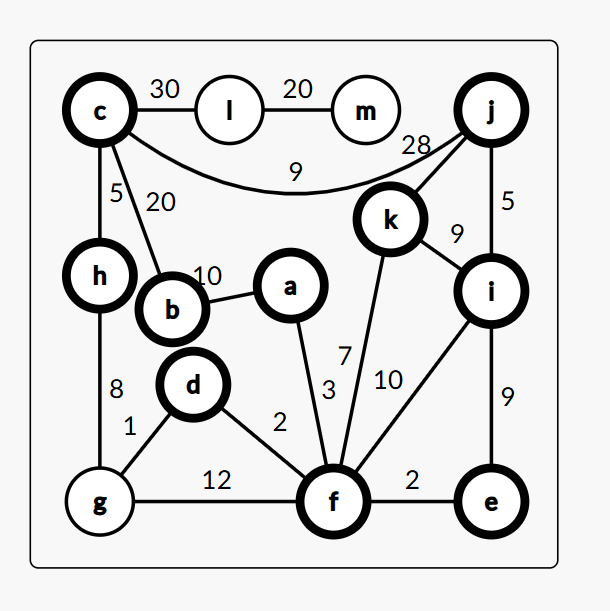
**Project Proposal: Shortest Path Finder Using Dijkstra's Algorithm**

**1. Project Title:** Shortest Path Finder Using Dijkstra's Algorithm

**2. Project Overview:** This project aims to develop a C-based program that calculates the shortest path between nodes in a weighted, undirected graph using Dijkstra's algorithm. The program reads graph data from an input file, processes the shortest path calculations, and outputs the results to an output file. This tool is useful for applications such as navigation systems (navigation systems use the algorithm to determine the shortest or fastest route between locations, such as in GPS-based mapping applications like Google Maps or vehicle navigation software ) , network routing, and urban planning.

****

**3. Objectives:**

* Implement Dijkstra’s algorithm to determine the shortest paths from a given source node.
* Read graph data dynamically from an input file to handle different scenarios.
* Represent nodes using custom naming conventions rather than fixed indices.
* Output the shortest distance and path to each node in a structured format.
* Ensure efficient memory usage and optimization for handling large graphs.

**4. Features:**

* Dynamic Node Handling: Users can define their own node names, and the system assigns them indices dynamically.
* File-Based Input and Output: The program reads edges and weights from a file and writes results to an output file.
* Path Tracing: The program reconstructs and displays the optimal paths from the source node to all other nodes.
* Error Handling: Proper handling of invalid inputs and missing files.
* Graph Representation: Uses an adjacency matrix for efficient distance updates.

**5. Implementation Details:**

* **Programming Language:** C
* **Data Structures Used:**
  + Adjacency Matrix for graph representation
  + Arrays for storing distances, visited nodes, and parent nodes
* **Algorithm:** Dijkstra’s Algorithm (Greedy approach)
* **File Handling:**
  + Input format: <node1> <node2> <weight>
  + Source node is specified at the end of the input file.
  + Output format: Node -> Distance -> Path

**6. Expected Outcomes:**

* A functional C program that efficiently finds the shortest path in a weighted graph.
* A structured and well-documented output file showing shortest paths from the source node.
* Improved understanding of graph algorithms and their practical applications.

**7. Potential Enhancements:**

* Implementing a graphical user interface (GUI) for easier input and visualization.
* Supporting directed graphs and negative weights using Bellman-Ford Algorithm.
* Improving efficiency by using a priority queue (Min-Heap) instead of an adjacency matrix.
* Allowing real-time input rather than relying on a pre-defined file.

**8. Conclusion:** This project provides a practical and efficient implementation of Dijkstra's Algorithm for shortest path computation. It can serve as an educational tool for understanding graph theory and algorithms or be extended for real-world applications such as network routing and urban traffic management.