**Title: Implementing Shortest Path Routing Using Dijkstra's Algorithm**

**Objective:** The objective of this project is to implement and understand the shortest path routing protocol using Dijkstra's algorithm in C++. This project demonstrates how the link-state approach efficiently finds the shortest paths from a source node to all other nodes in a network, a core concept in network routing.

**Problem Statement:** Implement a shortest path routing protocol using the link-state approach, specifically Dijkstra's algorithm, to calculate the shortest paths from a source node to all other nodes in a weighted, directed graph. The program should output the shortest distance to each reachable node and indicate nodes that are unreachable.

**CODE:**

#include <iostream>

#include <vector>

#include <queue>

#include <climits>

using namespace std;

typedef pair<int, int> Edge;

void dijkstra(int n, vector<vector<Edge>>& adj, int src) {

vector<int> dist(n, INT\_MAX);

dist[src] = 0;

priority\_queue<Edge, vector<Edge>, greater<Edge>> pq;

pq.push({0, src});

while (!pq.empty()) {

int u = pq.top().second;

int d = pq.top().first;

pq.pop();

if (d > dist[u]) continue;

for (auto &[weight, v] : adj[u]) {

int newDist = dist[u] + weight;

if (newDist < dist[v]) {

dist[v] = newDist;

pq.push({dist[v], v});

}

}

}

cout << "Shortest distances from source node " << src << ":\n";

for (int i = 0; i < n; i++) {

cout << "Node " << i << " : ";

if (dist[i] == INT\_MAX) {

cout << "Unreachable" << endl;

} else {

cout << dist[i] << endl;

}

}

}

int main() {

int n = 5;

vector<vector<Edge>> adj(n);

adj[0].push\_back({10, 1});

adj[0].push\_back({3, 2});

adj[1].push\_back({1, 2});

adj[1].push\_back({2, 3});

adj[2].push\_back({4, 3});

adj[3].push\_back({2, 4});

adj[2].push\_back({8, 4});

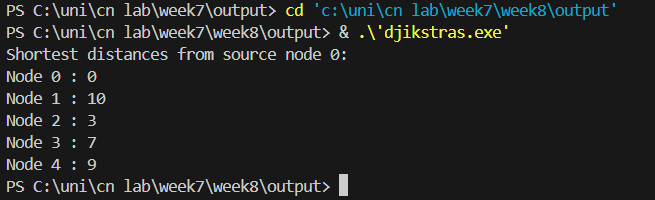
int src = 0;

dijkstra(n, adj, src);

return 0;

}

**OUTPUT :**

****

**Problems faced:** One issue encountered during development was handling nodes that were unreachable from the source node, which required special handling to avoid incorrect distance calculations. Another challenge was ensuring efficient handling of large graphs by using a priority queue to optimize node selection in Dijkstra's algorithm.

**Conclusion:** Through this project, I gained a deeper understanding of Dijkstra's algorithm and its application in network routing protocols. I learned to implement the link-state approach to routing, optimize code for efficiency using data structures like priority queues, and handle edge cases such as unreachable nodes, enhancing my problem-solving and coding skills in C++.