**Experiment-7**

**Aim-** **⁠** From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.

**Theory-**

**Dijkstra's Algorithm** is used to find the shortest paths from a given starting vertex to all other vertices in a weighted graph. It uses a priority queue to explore the closest unvisited node, ensuring that the shortest known distance to each vertex is maintained. Dijkstra's algorithm is widely used in networking and real-time systems for route optimization.

**Software Used –** Visual Studio Code

**Code-**

#include <iostream>

#include <vector>

#include <queue>

#include <climits> // For INT\_MAX

using namespace std;

class Edge {

public:

int v;

int wt;

Edge(int v, int wt) {

this->v = v;

this->wt = wt;

}

};

void dijkstra(int src, vector<vector<Edge> > &graph, int V) {

// Min-heap priority queue (distance, vertex)

priority\_queue<pair<int, int>, vector<pair<int, int> >, greater<pair<int, int> > > pq;

vector<int> dist(V, INT\_MAX); // Distance from source to each vertex

dist[src] = 0;

pq.push(make\_pair(0, src)); // Push starting vertex with distance 0

while (!pq.empty()) {

int u = pq.top().second; // Get the vertex with the smallest distance

pq.pop();

// Explore neighbors of u

for (Edge e : graph[u]) {

int v = e.v;

int weight = e.wt;

// Relaxation step

if (dist[v] > dist[u] + weight) {

dist[v] = dist[u] + weight;

pq.push(make\_pair(dist[v], v));

}

}

}

// Print shortest distances from the source vertex

cout << "Vertex Distance from Source" << endl;

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

cout << i << " \t\t " << dist[i] << endl;

}

}

int main() {

int V = 6; // Number of vertices

vector<vector<Edge> > graph(V);

// Add edges to the graph (directed graph)

graph[0].push\_back(Edge(1, 2));

graph[0].push\_back(Edge(2, 4));

graph[1].push\_back(Edge(2, 1));

graph[1].push\_back(Edge(3, 7));

graph[2].push\_back(Edge(4, 3));

graph[3].push\_back(Edge(5, 1));

graph[4].push\_back(Edge(3, 2));

graph[4].push\_back(Edge(2, 5));

// Run Dijkstra's algorithm from vertex 0

dijkstra(0, graph, V);

return 0;

}

**Output-**

