BHARATIYA VIDYA BHAVAN'S

SARDAR PATEL INSTITUTE OF TECHNOLOGY

Munshi nagar, Andheri (W) ,Mumbai - 400058

DEPARTMENT OF MASTER OF COMPUTER APPLICATION

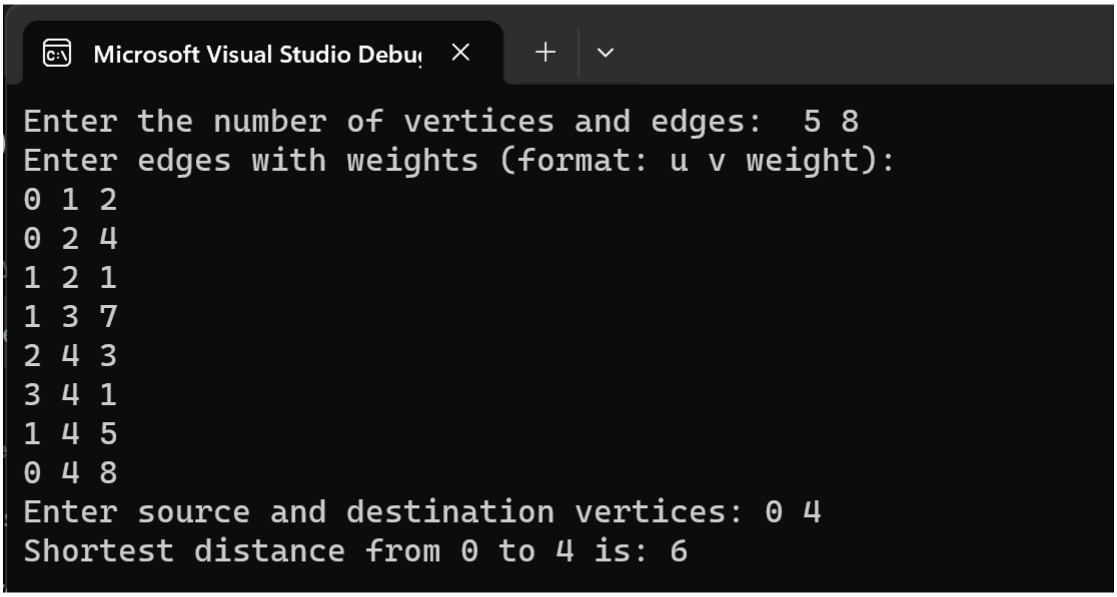
Aim:

Implement Single Source Shortest Path Algorithm, find the shortest path between the pair of vertices given by user.

Code:

```
#include <iostream>
#include <vector>
#include <queue>
#include <climits>
using namespace std;

#define INF INT_MAX
class Graph {
    int V;
    vector<vector<pair<int, int>>> adj;

public:
    Graph(int V) {
        this->V = V;
        adj.resize(V);
    }

    void addEdge(int u, int v, int weight) {
        adj[u].push_back({ v, weight });
        adj[v].push_back({ u, weight }); // For an undirected graph
    }

    vector<int> dijkstra(int src) {
        vector<int> dist(V, INF);
        dist[src] = 0;

        priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;
        pq.push({ 0, src });

        while (!pq.empty()) {
            int u = pq.top().second;
            pq.pop();

            for (auto i : adj[u]) {
                int v = i.first;
                int weight = i.second;

                if (dist[v] > dist[u] + weight) {
                    dist[v] = dist[u] + weight;
                    pq.push({ dist[v], v });
                }
            }
        }
        return dist;
    }
};

int main() {
    int V, E; // V -> Number of vertices, E -> Number of edges
    cout << "Enter the number of vertices and edges: ";
    cin >> V >> E;

    Graph g(V);

    cout << "Enter edges with weights (format: u v weight):" << endl;
    for (int i = 0; i < E; ++i) {
        int u, v, weight;
        cin >> u >> v >> weight;
        g.addEdge(u, v, weight);
    }

    int src, dest;
    cout << "Enter source and destination vertices: ";
    cin >> src >> dest;

    vector<int> shortestPath = g.dijkstra(src);

    if (shortestPath[dest] != INF) {
        cout << "Shortest distance from " << src << " to " << dest << " is: " << shortestPath[dest] << endl;
    }
    else {
        cout << "There is no path from " << src << " to " << dest << endl;
    }

    return 0;
}
```

Output:



```
Enter the number of vertices and edges:  5 8
Enter edges with weights (format: u v weight):
0 1 2
0 2 4
1 2 1
1 3 7
2 4 3
3 4 1
1 4 5
0 4 8
Enter source and destination vertices: 0 4
Shortest distance from 0 to 4 is: 6
```