**Worksheet-3.2**

**Student Name:-** Pushpraj Roy **UID:-** 20BCS9866

**Branch:-** BE- CSE **Section/Group:-** WM\_617 “A”

**Subjetct Code:-** 20CSP-312 **Semester:-** 5th

**Subject Name:-** DAA Lab

1. **Aim/Overview of the practical: -**

Code and analyze to find shortest paths in a graph with positive edge weights using Dijkstra’s algorithm.

1. **Task to be done/which logistics used:-**

* Dijkstra’s algorithm

1. **Algorithm/Flowchart :-**

Dijkstra's Algorithm (G, w, s)

1. INITIALIZE - SINGLE - SOURCE (G, s)

2. S←∅

3. Q←V [G]

4. while Q ≠ ∅

5. do u ← EXTRACT - MIN (Q)

6. S ← S ∪ {u}

7. for each vertex v ∈ Adj [u]

8. do RELAX (u, v, w)

1. **Steps for experiment/practical/Code :-**

#include <iostream>

using namespace std;

#include <limits.h>

#define V 9

int minDistance(int dist[], bool sptSet[])

{

int min = INT\_MAX, min\_index;

for (int v = 0; v < V; v++)

if (sptSet[v] == false && dist[v] <= min)

min = dist[v], min\_index = v;

return min\_index;

}

void printSolution(int dist[])

{

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

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

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

}

void dijkstra(int graph[V][V], int src)

{

int dist[V];

bool sptSet[V];

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

dist[i] = INT\_MAX, sptSet[i] = false;

dist[src] = 0;

for (int count = 0; count < V - 1; count++) {

int u = minDistance(dist, sptSet);

sptSet[u] = true;

for (int v = 0; v < V; v++)

if (!sptSet[v] && graph[u][v]

&& dist[u] != INT\_MAX

&& dist[u] + graph[u][v] < dist[v])

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

}

printSolution(dist);

}

int main()

{

int graph[V][V] = { { 0, 4, 0, 0, 0, 0, 0, 8, 0 },

{ 4, 0, 8, 0, 0, 0, 0, 11, 0 },

{ 0, 8, 0, 7, 0, 4, 0, 0, 2 },

{ 0, 0, 7, 0, 9, 14, 0, 0, 0 },

{ 0, 0, 0, 9, 0, 10, 0, 0, 0 },

{ 0, 0, 4, 14, 10, 0, 2, 0, 0 },

{ 0, 0, 0, 0, 0, 2, 0, 1, 6 },

{ 8, 11, 0, 0, 0, 0, 1, 0, 7 },

{ 0, 0, 2, 0, 0, 0, 6, 7, 0 } };

dijkstra(graph, 0);

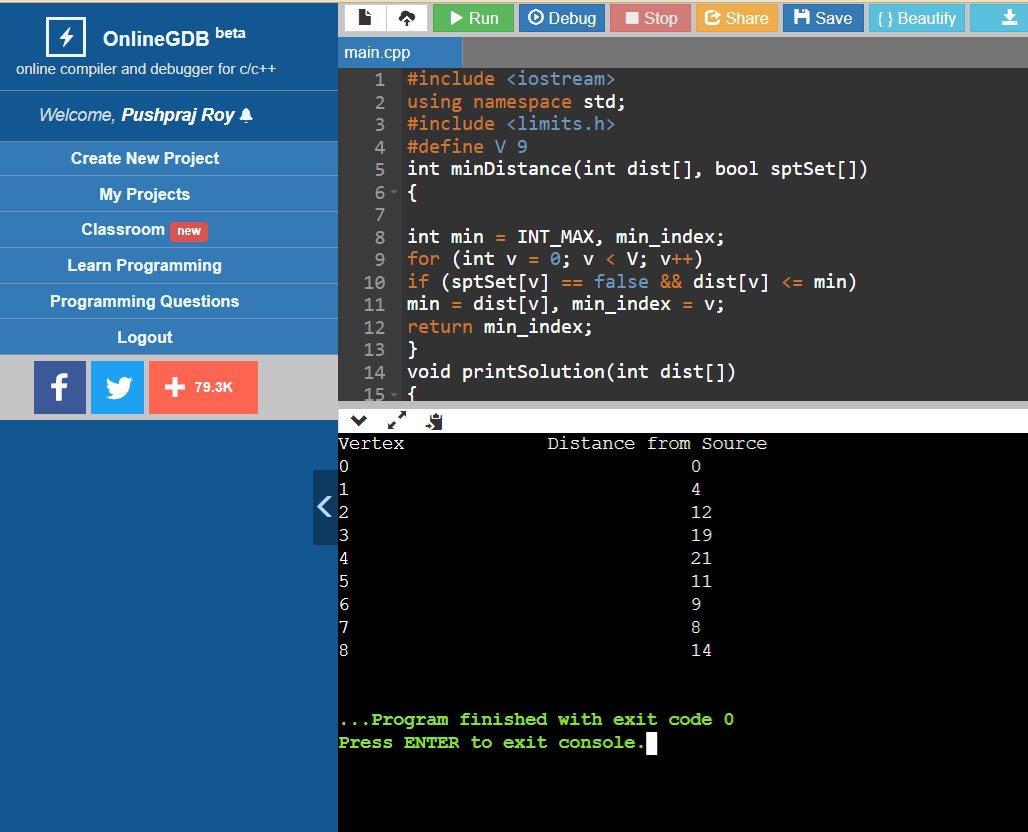
return 0;

}

1. **Observations/Discussions/ Complexity Analysis:**

* Time Complexity: O (E\*log V)
* Space complexity: O(V)

1. **Result/Output :-**

****

1. **Learning outcomes (What I have learnt) –**

* The code finds the shortest distances from the source to all vertices.
* If we are interested only in the shortest distance from the source to a single target,

break them for a loop when the picked minimum distance vertex is equal to the

target.

* To learn the importance of designing an algorithm in an effective way by considering space and time complexity.