**Assignment 4:**

**GitHub Link:**

<https://github.com/mdshakebj/Advance_Algorithm_Assignment-4>

**C++ program for Dijkstra's single source shortest path**

#include <iostream>

#include <limits.h>

#include <queue>

#include <vector>

#include <utility>

using namespace std;

#define V 9

// Function that implements Dijkstra's single source shortest path algorithm

// for a graph represented using adjacency matrix representation

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

{

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

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

pq.push(make\_pair(0, src));

dist[src] = 0;

while (!pq.empty()) {

int u = pq.top().second;

pq.pop();

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

if (graph[u][v] && dist[u] + graph[u][v] < dist[v]) {

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

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

}

}

}

printf("Vertex Distance from Source\n");

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

printf("%d tt %d\n", i, dist[i]);

}

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;

}

