**Assignment 4**

**GitHub Link:**

[https://github.com/sufiyanjunaidi13/Advance\_Algorithm\_Assignment-4 (github.com)](https://github.com/sufiyanjunaidi13/Advance_Algorithm_Assignment-4)

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

#include <iostream>

#include <vector>

#include <queue>

#include <climits>

using namespace std;

// Number of vertices in the graph

#define V 9

// A utility function to print the constructed distance

// array

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;

}

// Function that implements Dijkstra's single source

// shortest path algorithm for a graph represented using

// adjacency list representation

void dijkstra(vector<vector<pair<int, int>>> graph, int src)

{

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

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

pq.push({0, src});

dist[src] = 0;

while (!pq.empty()) {

int u = pq.top().second;

pq.pop();

for (auto v : graph[u]) {

int weight = v.second;

int vertex = v.first;

if (dist[u] != INT\_MAX && dist[u] + weight < dist[vertex]) {

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

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

}

}

}

// print the constructed distance array

printSolution(dist.data());

}

// driver's code

int main()

{

/\* Let us create the example graph discussed above \*/

vector<vector<pair<int, int>>> graph = {{{1, 4}, {7, 8}},

{{0, 4}, {2, 8}, {7, 11}},

{{1, 8}, {3, 7}, {5, 4}, {8, 2}},

{{2, 7}, {4, 9}, {5, 14}},

{{3, 9}, {5, 10}},

{{2, 4}, {3, 14}, {4, 10}, {6, 2}},

{{5, 2}, {7, 1}, {8, 6}},

{{0, 8}, {1, 11}, {6, 1}, {8, 7}},

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

// Function call

dijkstra(graph, 0);

return 0;

}

