Assignment On

"Modern Operating System and Computer Networks"

(Assignment-1.1)

Submitted by: Madhu boini

2503B09904(MCA)

Submitted to: Dr. Mathivanan D

Question:

Q. Write a C++ program to implement Dijkstra's Single Source Shortest Path Algorithm for a graph represented using an adjacency matrix.

Number of vertices: 5

Edges:

0 1 4

0 2 8

1 4 6

2 3 2

3 4 10

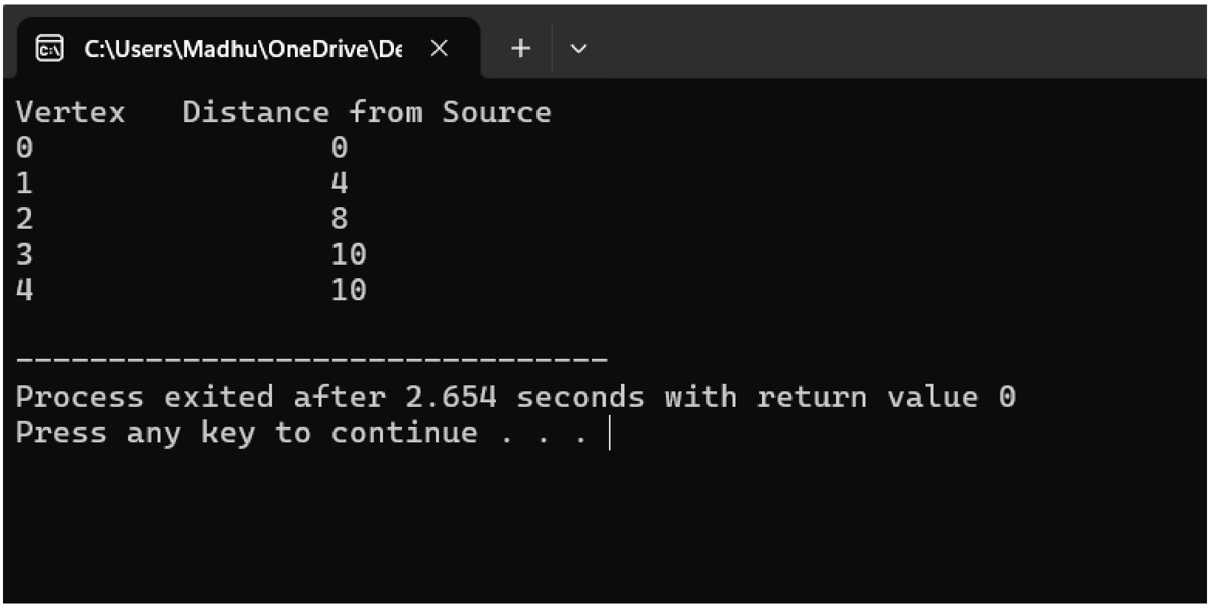
Source vertex: 0

Code :

```
#include <iostream>
#include <climits> // For INT_MAX
using namespace std;

#define V 5  // Number of vertices

// Function to find the vertex with minimum distance value, from
// the set of vertices not yet included in shortest path tree
int minDistance(int dist[], bool sptSet[]) {
   int min = INT_MAX, min_index;
```

```
    for (int v = 0; v < V; v++) {
        if (!sptSet[v] && dist[v] <= min) {
            min = dist[v];
            min_index = v;
        }
    }
    return min_index;
}

// Function to print the distance array
void printSolution(int dist[]) {
    cout << "Vertex \t Distance from Source\n";
    for (int i = 0; i < V; i++)
        cout << i << " \t\t " << dist[i] << endl;
}

// Function that implements Dijkstra's single source shortest path algorithm
// for a graph represented using adjacency matrix
void dijkstra(int graph[V][V], int src) {
    int dist[V]; // The output array. dist[i] will hold the shortest distance from src to i
    bool sptSet[V]; // sptSet[i] will be true if vertex i is included in shortest path tree

    // Initialize all distances as INFINITE and sptSet[] as false
    for (int i = 0; i < V; i++) {
        dist[i] = INT_MAX;
        sptSet[i] = false;
    }

    // Distance of source vertex from itself is always 0
    dist[src] = 0;
    // Find shortest path for all vertices
    for (int count = 0; count < V - 1; count++) {
        // Pick the minimum distance vertex from the set of vertices not yet processed
        int u = minDistance(dist, sptSet);

        // Mark the picked vertex as processed
        sptSet[u] = true;

        // Update dist value of the adjacent vertices of the picked vertex
        for (int v = 0; v < V; v++) {
            // Update dist[v] only if it is not in sptSet, there is an edge from u to v,
            // and total weight of path from src to v through u is smaller than current value of dist[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];
            }
        }
    }

    // Print the constructed distance array
    printSolution(dist);
}

int main() {
    // Adjacency matrix representation of the graph
    int graph[V][V] = {
        {0, 4, 8, 0, 0},
        {0, 0, 0, 0, 6},
        {0, 0, 0, 2, 0},
        {0, 0, 0, 0, 10},
        {0, 0, 0, 0, 0}
    };

    int source = 0;
    dijkstra(graph, source);

    return 0;
}
```

Output :



```
Vertex   Distance from Source
0                0
1                4
2                8
3                10
4                10

--------------------------------
Process exited after 2.654 seconds with return value 0
Press any key to continue . . .
```