Practical-3.2

Student Name: Rajdeep Jaiswal **UID:** 20BCS2761
Branch: CSE **Section/Group: 902 WM B**
Semester: 05
Subject Name: Design & Analysis Algorithm **Subject Code:** 20CSP-312

1. Aim:

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

2. Task to be done:

To implement Dijkstra's algorithm.

3. Algorithm:

Dijkstra's Algorithm (G, w, s)
1. INITIALIZE - SINGLE - SOURCE (G, s)
2. S←Ø
3. Q←V [G]
4. while Q ≠ Ø
5. do u ← EXTRACT - MIN (Q)
6. S ← S ∪ {u}
7. for each vertex v ∈ Adj [u]
8. do RELAX (u, v, w)

Code:

```
 #include <iostream>
using namespace std;
#include <limits.h>

// Number of vertices in the graph
#define V 9

// A utility 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[])
{

   // Initialize min value
   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;
}

// 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 matrix representation
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 or shortest distance from src to i is
   // finalized

   // Initialize all distances as INFINITE and stpSet[] 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;
```

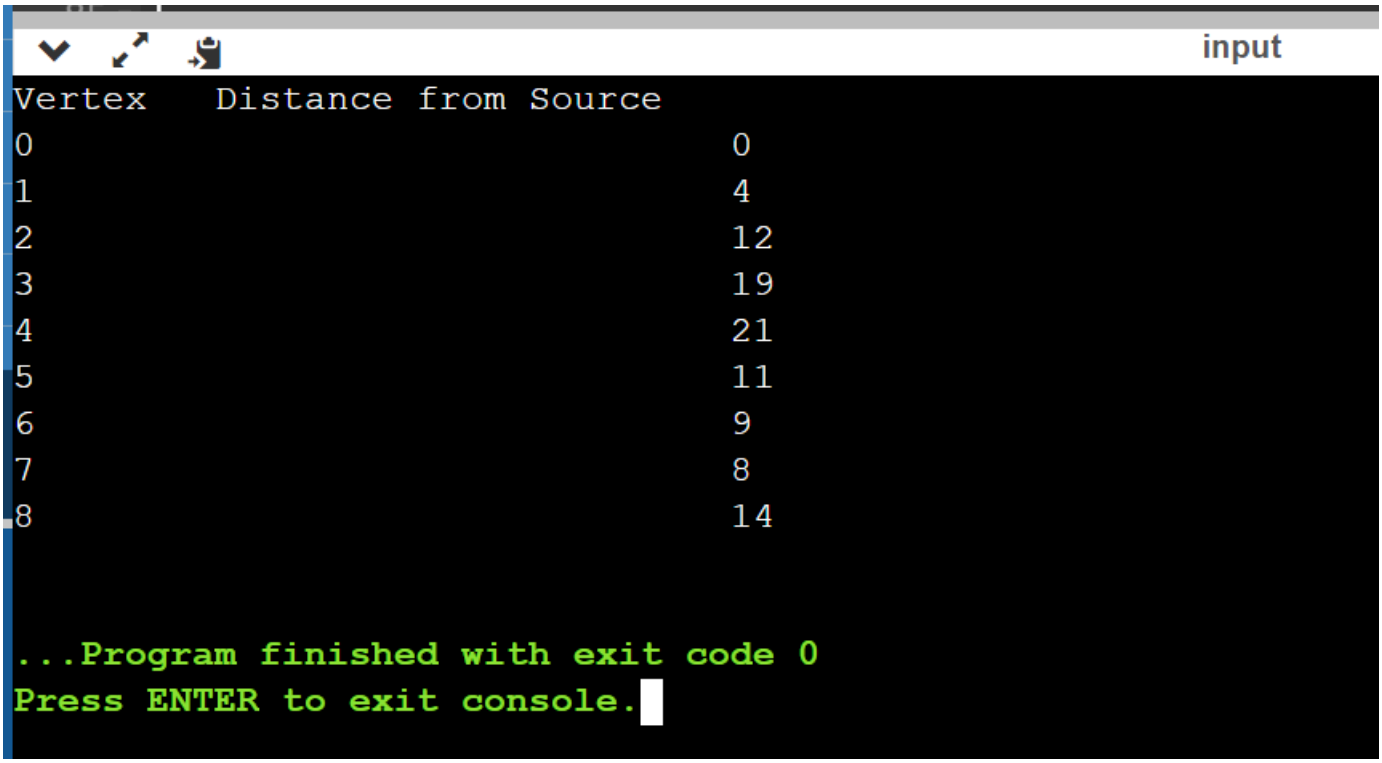
```
    // Find shortest path for all vertices
    for (int count = 0; count < V - 1; count++) {
        // Pick the minimum distance vertex from the set of
        // vertices not yet processed. u is always equal to
        // src in the first iteration.
        int u = minDistance(dist, sptSet);

        // 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 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);
}
```

```
// driver's code
int main()
{

    /* Let us create the example graph discussed above */
    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 } };

    // Function call
    dijkstra(graph, 0);

    return 0;
}
```

Complexity Analysis:

Time Complexity: O(V2)
Auxiliary Space: O(V)

5. **Result:**



Learning outcomes (What I have learnt):

1. Learn about finding shortest path in the graph.
2. Learn about time complexity of program.
3. Learnt to implement Dijsktra's algorithm using Prim's algo.