1. irst of all, we will mark all vertex as unvisited vertex
2. Then, we will mark the source vertex as 0 and all other vertices as infinity
3. Consider source vertex as current vertex
4. Calculate the path length of all the neighboring vertex from the current vertex by adding the weight of the edge in the current vertex
5. Now, if the new path length is smaller than the previous path length then replace it otherwise ignore it
6. Mark the current vertex as visited after visiting the neighbor vertex of the current vertex
7. Select the vertex with the smallest path length as the new current vertex and go back to step 4.
8. Repeat this process until all the vertex are marked as visited.

Pseudocode of Dijkstra’s Algorithm in C++

function dijkstra(G, S)

**for** each vertex V **in** G

dist[V] <- infinite

prev[V] <- NULL

If V != S, add V to Priority Queue Q

dist[S] <- **0**

**while** Q IS NOT EMPTY

U <- Extract MIN **from** **Q**

**for** each unvisited neighbour V of U

temperoryDist <- dist[U] + edgeWeight(U, V)

**if** temperoryDist < dist[V]

dist[V] <- temperoryDist

prev[V] <- U

**return** dist[], prev[]

#include

#include

using namespace std;

int miniDist(int distance[], bool Tset[]) // finding minimum distance

{

int minimum=INT\_MAX,ind;

**for**(int k=**0**;k<**6**;k++)

{

**if**(Tset[k]==false && distance[k]<=minimum)

{

minimum=distance[k];

ind=k;

}

}

**return** ind;

}

void DijkstraAlgo(int graph[**6**][**6**],int src) // adjacency matrix

{

int distance[**6**]; // // array to calculate the minimum distance **for** each node

bool Tset[**6**];// boolean array to mark visited **and** unvisited **for** each node

**for**(int k = **0**; k<**6**; k++)

{

distance[k] = INT\_MAX;

Tset[k] = false;

}

distance[src] = **0**; // Source vertex distance **is** set **0**

**for**(int k = **0**; k<**6**; k++)

{

int m=miniDist(distance,Tset);

Tset[m]=true;

**for**(int k = **0**; k<**6**; k++)

{

// updating the distance of neighbouring vertex

**if**(!Tset[k] && graph[m][k] && distance[m]!=INT\_MAX && distance[m]+graph[m][k]<distance[k])

distance[k]=distance[m]+graph[m][k];

}

}

cout<<"Vertex**\t\t**Distance from source vertex"<<endl;

**for**(int k = **0**; k<**6**; k++)

{

char str=**65**+k;

cout<<str<<"**\t\t\t**"<<distance[k]<<endl;

}

}

int main()

{

int graph[**6**][**6**]={

{**0**, **1**, **2**, **0**, **0**, **0**},

{**1**, **0**, **0**, **5**, **1**, **0**},

{**2**, **0**, **0**, **2**, **3**, **0**},

{**0**, **5**, **2**, **0**, **2**, **2**},

{**0**, **1**, **3**, **2**, **0**, **1**},

{**0**, **0**, **0**, **2**, **1**, **0**}};

DijkstraAlgo(graph,**0**);

**return** **0**;

}