CNL ASSIGNMENT 09
SHORTEST PATH ALGORITHM

ANTRIKSH SHARMA **01 / 10 / 2022**
20070122021
CS-A1

Objective: To study and implement the Shortest Path Routing Algorithm.

Theory:

In computer networks, the shortest path algorithms aim to find the optimal paths between the network nodes so that routing cost is minimized. They are direct applications of the shortest path algorithms proposed in graph theory.

Explanation

Consider that a network comprises of N vertices (nodes or network devices) that are connected by M edges (transmission lines). Each edge is associated with a weight, representing the physical distance or the transmission delay of the transmission line. The target of shortest path algorithms is to find a route between any pair of vertices along the edges, so the sum of weights of edges is minimum. If the edges are of equal weights, the shortest path algorithm aims to find a route having minimum number of hops.

SHORTEST PATH ALGORITHM (DIJKSTRA'S ALGO) IMPLEMENTATION

```
#include <iostream>
#include <limits.h>
using namespace std;
#define V 9
int minDistance(int dist[], bool sptSet[])
{
    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;
}
void printSolution(int dist[], int n)
```

```
{
    cout << "Source - A" << endl;
    cout << "Vertex Distance from Source" << endl;
    for (int i = 0; i < V; i++)
    {
        cout << i << "\t\t" << dist[i] << "ms" << endl;
    }
}
void dijkstra(int graph[V][V], int src)
{
    int dist[V];
    bool sptSet[V];
    for (int i = 0; i < V; i++)
    {
        dist[i] = INT_MAX, sptSet[i] = false;
    }
    dist[src] = 0;
    for (int count = 0; count < V - 1; count++)
    {
        int u = minDistance(dist, sptSet);
        sptSet[u] = true;
        for (int v = 0; v < V; 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];
            }
        }
    }
    printSolution(dist, V);
}
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;
}
```

```
Source - A
Vertex   Distance from Source
0                0ms
1                4ms
2                12ms
3                19ms
4                21ms
5                11ms
6                9ms
7                8ms
8                14ms
```