

NAME:	Sanskar Kamble
UID:	2021300054
SUBJECT	Design and Analysis of Algorithm
EXPERIMENT NO :	06
DATE OF PERFORMANCE	27/03/2023
DATE OF SUBMISSION	03/04/2023
AIM:	To find the shortest possible path using Dijkstra algorithm.
ALGORITHM and THEORY:	<pre> function dijkstra(G, S) for each vertex V in G distance[V] <- infinite previous[V] <- NULL If V != S, add V to Priority Queue Q distance[S] <- 0 while Q IS NOT EMPTY U <- Extract MIN from Q for each unvisited neighbour V of U tempDistance <- distance[U] + edge_weight(U, V) if tempDistance < distance[V] distance[V] <- tempDistance previous[V] <- U return distance[], previous[] </pre>

PROGRAM:

```
#include <limits.h>
#include <stdbool.h>
#include <stdio.h>
int V;
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
void printSolution(int dist[])
{
    printf("Vertex \t\t Distance from Source\n");
    for (int i = 0; i < V; i++)
        printf("%d \t\t\t %d\n", i, dist[i]);
}

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];
    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;

        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);
    }

    int main()
    {

        int s;
        printf("Enter the no of elements:");
        scanf("%d",&V);
        int graph[V][V];
        printf("Enter the elements in the graph:\n");
        for(int i=0;i<V;i++)
        {
            printf("Enter the elements in row %d:",(i+1));
            for(int j=0;j<V;j++)
            {
                scanf("%d",&graph[i][j]);
            }
        }
        printf("Enter the source:\n");
        scanf("%d",&s);
        dijkstra(graph,s);
    }

```

	<pre>return 0; }</pre>
OUTPUT:	<pre>students@students-HP-280-G3-MT:~\$ cd Desktop students@students-HP-280-G3-MT:~/Desktop\$ gcc daa_Ex6.c students@students-HP-280-G3-MT:~/Desktop\$./a.out Enter the no of elements:4 Enter the elements in the graph: Enter the elements in row 1:2 3 4 5 Enter the elements in row 2:1 6 8 10 Enter the elements in row 3:7 9 11 12 Enter the elements in row 4:13 15 17 18 Enter the source: 2 Vertex Distance from Source 0 7 1 9 2 0 3 12 students@students-HP-280-G3-MT:~/Desktop\$</pre>

	<pre>students@students-HP-280-G3-MT:~/Desktop\$ gcc daa_Ex6.c students@students-HP-280-G3-MT:~/Desktop\$./a.out Enter the no of elements:5 Enter the elements in the graph: Enter the elements in row 1:1 3 5 7 9 Enter the elements in row 2:2 4 6 8 10 Enter the elements in row 3:11 13 15 17 19 Enter the elements in row 4:12 14 16 18 20 Enter the elements in row 5:21 23 25 27 29 Enter the source: 1 Vertex Distance from Source 0 2 1 0 2 6 3 8 4 10 students@students-HP-280-G3-MT:~/Desktop\$</pre>
CONCLUSION:	I have successfully understood Dijkstra's Algorithm and also found the shortest possible path using it.