**Dijkstra's algorithm**

**Design and implement C/C++ Program to find shortest paths from a given vertex in a weighted connected graph to other V using Dijkstra's algorithm**

#include <stdio.h>

#include <limits.h>

#define MAX 50

int minDistance(int dist[], int sptSet[], int V)

{

int min = INT\_MAX, minIndex;

for (int v = 0; v < V; v++)

{

if (!sptSet[v] && dist[v] < min)

{

min = dist[v];

minIndex = v;

}

}

return minIndex;

}

void printSolution(int dist[], int V)

{

printf("Vertex \tDistance from Source\n");

for (int i = 0; i < V; i++)

{

printf("%d \t%d\n", i, dist[i]);

}

}

void dijkstra(int graph[MAX][MAX], int src, int V)

{

int dist[MAX];

int sptSet[MAX];

for (int i = 0; i < V; i++)

{

dist[i] = INT\_MAX;

sptSet[i] = 0;

}

dist[src] = 0;

for (int count = 0; count < V - 1; count++)

{

int u = minDistance(dist, sptSet, V);

sptSet[u] = 1;

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 V;

printf("Input the number of vertices: ");

scanf("%d", &V);

int graph[MAX][MAX];

printf("Input the adjacency matrix for the graph:\n");

for (int i = 0; i < V; i++)

{

for (int j = 0; j < V; j++)

{

scanf("%d", &graph[i][j]);

}

}

int source;

printf("Input the source vertex: ");

scanf("%d", &source);

dijkstra(graph, source, V);

return 0;

}

Output:

1. Input the number of vertices: 5

Input the adjacency matrix for the graph:

0 3 2 0 0

3 0 0 1 0

2 0 0 1 4

0 1 1 0 2

0 0 4 2 0

Input the source vertex: 0

Vertex Distance from Source

0 0

1 3

2 2

3 3

4 5

1. Input the number of vertices: 7

Input the adjacency matrix for the graph:

0 4 0 0 0 8 0

4 0 8 0 0 11 0

0 8 0 7 0 4 0

0 0 7 0 9 14 0

0 0 0 9 0 10 2

0 0 4 14 10 0 2

0 0 0 0 2 0 1

Input the source vertex: 0

Vertex Distance from Source

0 0

1 4

2 12

3 19

4 12

5 8

6 10