#include <stdio.h>

#include <stdbool.h>

#include <limits.h>

#define V 6 // Number of vertices in the graph

// Function to find the vertex with the minimum distance value

int minDistance(int dist[], bool visited[]) {

int min = INT\_MAX, min\_index;

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

if (!visited[v] && dist[v] <= min) {

min = dist[v];

min\_index = v;

}

}

return min\_index;

}

// Function to print the path from source to destination

void printPath(int parent[], int j) {

if (parent[j] == -1) {

printf("%d ", j);

return;

}

printPath(parent, parent[j]);

printf("%d ", j);

}

// Function to print the minimum path from source to destination

void printSolution(int dist[], int parent[], int dest) {

printf("Minimum Path from 1 to %d: ", dest);

printPath(parent, dest);

printf("\nMinimum Distance: %d\n", dist[dest]);

}

// Function to find the minimum path from source to destination using Dijkstra's algorithm

void dijkstra(int graph[V][V], int src, int dest) {

int dist[V]; // Array to store the minimum distances from source to each vertex

bool visited[V]; // Array to keep track of visited vertices

int parent[V]; // Array to store the parent of each vertex in the minimum path

// Initialize dist[] and visited[]

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

dist[i] = INT\_MAX;

visited[i] = false;

}

dist[src] = 0; // Distance of source vertex from itself is always 0

parent[src] = -1; // Source vertex has no parent

// Find the minimum path for all vertices

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

int u = minDistance(dist, visited);

visited[u] = true;

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

if (!visited[v] && graph[u][v] && dist[u] != INT\_MAX && dist[u] + graph[u][v] < dist[v]) {

dist[v] = dist[u] + graph[u][v];

parent[v] = u;

}

}

}

printSolution(dist, parent, dest);

}

int main() {

// Graph represented as an adjacency matrix

int graph[V][V] = {

{0, 4, 2, 0, 0, 0},

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

{0, 0, 0, 0, 2, 0},

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

{0, 0, 0, 0, 0, 2},

{0, 0, 0, 0, 0, 0}

};

int src = 0; // Source vertex (1)

int dest = 5; // Destination vertex (6)

dijkstra(graph, src, dest);

return 0;

}