**Find Minimum Cost Spanning Tree of a given undirected graph using Prim’s algorithm.**

//C program to implement prim's algorithm

#include <stdio.h>

int cost[10][10], n, t[10][2], sum;

void prims(int cost[10][10], int n);

int main() {

    int i, j;

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

    scanf("%d", &n);

    printf("Enter the cost adjacency matrix:\n");

    for (i = 0; i < n; i++) {

        for (j = 0; j < n; j++) {

            scanf("%d", &cost[i][j]);

        }

    }

    prims(cost, n);

    printf("Edges of the minimal spanning tree:\n");

    for (i = 0; i < n - 1; i++) {

        printf("(%d, %d) ", t[i][0], t[i][1]);

    }

    printf("\nSum of minimal spanning tree: %d\n", sum);

    return 0;

}

void prims(int cost[10][10], int n) {

    int i, j, u, v;

    int min, source;

    int p[10], d[10], s[10];

    min = 999;

    source = 0;

    // Initialize arrays

    for (i = 0; i < n; i++) {

        d[i] = cost[source][i];

        s[i] = 0;

        p[i] = source;

    }

    s[source] = 1;

    sum = 0;

    int k = 0;

    // Find MST

    for (i = 0; i < n - 1; i++) {

        min = 999;

        u = -1;

        // Find the vertex with minimum distance to the MST

        for (j = 0; j < n; j++) {

            if (s[j] == 0 && d[j] < min) {

                min = d[j];

                u = j;

            }

        }

        if (u != -1) {

            // Add edge to MST

            t[k][0] = u;

            t[k][1] = p[u];

            k++;

            sum += cost[u][p[u]];

            s[u] = 1;

            // Update distances

            for (v = 0; v < n; v++) {

                if (s[v] == 0 && cost[u][v] < d[v]) {

                    d[v] = cost[u][v];

                    p[v] = u;

                }

            }

        }

    }

}

Output:

