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

//C program to implement Kruskal’s algorithm

#include <stdio.h>

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

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

int find(int parent[10], int i);

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]);

        }

    }

    kruskal(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 kruskal(int cost[10][10], int n) {

    int min, u, v, count, k;

    int parent[10];

    k = 0;

    sum = 0;

    // Initialize parent array for Union-Find

    for (int i = 0; i < n; i++) {

        parent[i] = i;

    }

    count = 0;

    while (count < n - 1) {

        min = 999;

        u = -1;

        v = -1;

        // Find the minimum edge

        for (int i = 0; i < n; i++) {

            for (int j = 0; j < n; j++) {

                if (find(parent, i) != find(parent, j) && cost[i][j] < min) {

                    min = cost[i][j];

                    u = i;

                    v = j;

                }

            }

        }

        // Perform Union operation

        int root\_u = find(parent, u);

        int root\_v = find(parent, v);

        if (root\_u != root\_v) {

            parent[root\_u] = root\_v;

            t[k][0] = u;

            t[k][1] = v;

            sum += min;

            k++;

            count++;

        }

    }

}

int find(int parent[10], int i) {

    while (parent[i] != i) {

        i = parent[i];

    }

    return i;

}

Output:

