```c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

#define V 4 // Number of vertices (cities)

// Function to find the minimum of two integers
int min(int a, int b) {
    return (a < b) ? a : b;
}

// Function to find the factorial of a number
int factorial(int n) {
    if (n == 1 || n == 0)
        return 1;
    else
        return n * factorial(n - 1);
}

// Function to calculate the total distance of the current path
int calculateDistance(int path[], int graph[V][V]) {
    int totalDistance = 0;
    for (int i = 0; i < V - 1; i++) {
        totalDistance += graph[path[i]][path[i + 1]];
    }
    totalDistance += graph[path[V - 1]][path[0]]; // Return to the starting city
    return totalDistance;
}

// Function to print the path and its distance
void printPath(int path[], int graph[V][V]) {
    printf("Path: ");
    for (int i = 0; i < V; i++) {
        printf("%d ", path[i]);
    }
    printf("%d\n", path[0]); // Return to the starting city
    printf("Total Distance: %d\n", calculateDistance(path, graph));
}

// Function to solve the Traveling Salesman Problem using brute force
void tspBruteForce(int graph[V][V]) {
    int path[V];
    for (int i = 0; i < V; i++)
        path[i] = i;

    int minDistance = INT_MAX;
    int tempDistance;
    int minPath[V];
```

```
    // Calculate factorial of (V-1) because one vertex will be fixed
    int totalPaths = factorial(V - 1);

    do {
        tempDistance = calculateDistance(path, graph);
        if (tempDistance < minDistance) {
            minDistance = tempDistance;
            for (int i = 0; i < V; i++)
                minPath[i] = path[i];
        }
    } while (next_permutation(path + 1, path + V));

    printf("Shortest Path:\n");
    printPath(minPath, graph);
}

int main() {
    int graph[V][V] = {
        {0, 10, 15, 20},
        {10, 0, 35, 25},
        {15, 35, 0, 30},
        {20, 25, 30, 0}
    };

    tspBruteForce(graph);

    return 0;
}
```