PROGRAM TITLE 09

TRAVELLING SALESMAN PROBLEM

AIM:

To Write the python to implement Travelling Salesman Problem.

PROCEDURE:

1. Define City Coordinates: Define the coordinates of each city in a list. Each city is represented by a tuple containing its (x, y) coordinates.

2. Calculate Distance: Implement a function to calculate the distance between two cities using their coordinates. In this program, Euclidean distance is used.

3. Calculate Total Distance for a Path: Define a function to calculate the total distance for a given path that visits all cities once and returns to the starting city.

4. Brute Force Approach: Implement the brute force approach to solve the TSP problem. Iterate through all permutations of city paths, calculate the total distance for each path, and keep track of the minimum distance and corresponding path.

5. Main Program: In the main section of the program, provide an example set of cities. Call the traveling_salesman_brute_force function with the list of cities to find the optimal path and minimum distance. Finally, print the optimal path and minimum distance.

CODING:

```
import itertools


def calculate_distance(city1, city2):
    return ((city1[0] - city2[0])**2 + (city1[1] - city2[1])**2) ** 0.5


def total_distance(path, cities):
    distance = 0
    for i in range(len(path) - 1):
        distance += calculate_distance(cities[path[i]], cities[path[i + 1]])
```

```
    distance += calculate_distance(cities[path[-1]], cities[path[0]])  # Return to start
    return distance

def traveling_salesman_brute_force(cities):
    num_cities = len(cities)
    min_distance = float('inf')
    min_path = []

    for path in itertools.permutations(range(num_cities)):
        distance = total_distance(path, cities)
        if distance < min_distance:
            min_distance = distance
            min_path = path

    return min_path, min_distance

# Example usage:
if __name__ == "__main__":
    cities = [(0, 0), (1, 2), (3, 1), (5, 3)]
    optimal_path, min_distance = traveling_salesman_brute_force(cities)
    print("Optimal path:", optimal_path)
    print("Minimum distance:", min_distance)
```

OUTPUT:

```
Optimal path: (2, 0, 1, 3)
Minimum distance: 12.34987838803202
```

RESULT:

Hence the program been successfully executed and verified.