**Exp-2.12**

**Title:**

Exhaustive Search Solution for the Traveling Salesman Problem (TSP)

**Aim:**

To develop a brute force solution for TSP that enumerates all possible routes, calculates their distances, and finds the shortest route and distance.

**Procedure:**

1. Define a function distance(city1, city2) to calculate Euclidean distance between two cities.
2. Implement tsp(cities) to:
    - Use itertools. permutations to generate all permutations of cities except the starting city.
    - For each permutation, calculate total round-trip distance including return to start.
    - Track shortest distance and corresponding path.
3. Run test cases with example city configurations, print shortest distance and path for each.

**Algorithm:**

1. Start
2. Fix the starting city as the first city in the list.
3. Generate all permutations of the other cities.
4. For each permutation:
    - Construct full path by prepending and appending the starting city.
    - Compute total distance by summing distances between consecutive cities on path.
    - Update shortest path and distance if current route is shorter.
5. Return the shortest distance and path.
6. End.

**Input:**

[(1,2), (4,5), (7,1), (3,6)]

**Output:**

Test Case 1:

Shortest Distance: 7.0710678118654755

Shortest Path: [(1, 2), (4, 5), (7, 1), (3, 6), (1, 2)]

Test Case 2:

Shortest Distance: 14.142135623730951

Shortest Path: [(2, 4), (1, 7), (6, 3), (5, 9), (8, 1), (2, 4)]

**Program:**

```
import itertools
import math

def distance(city1, city2):
    return math.sqrt((city1[0] - city2[0]) ** 2 + (city1[1] - city2[1]) ** 2)

def tsp(cities):
    start = cities[0]
    min_dist = float('inf')
    shortest_path = []

    for perm in itertools.permutations(cities[1:]):
        path = [start] + list(perm) + [start]
        dist = 0
```

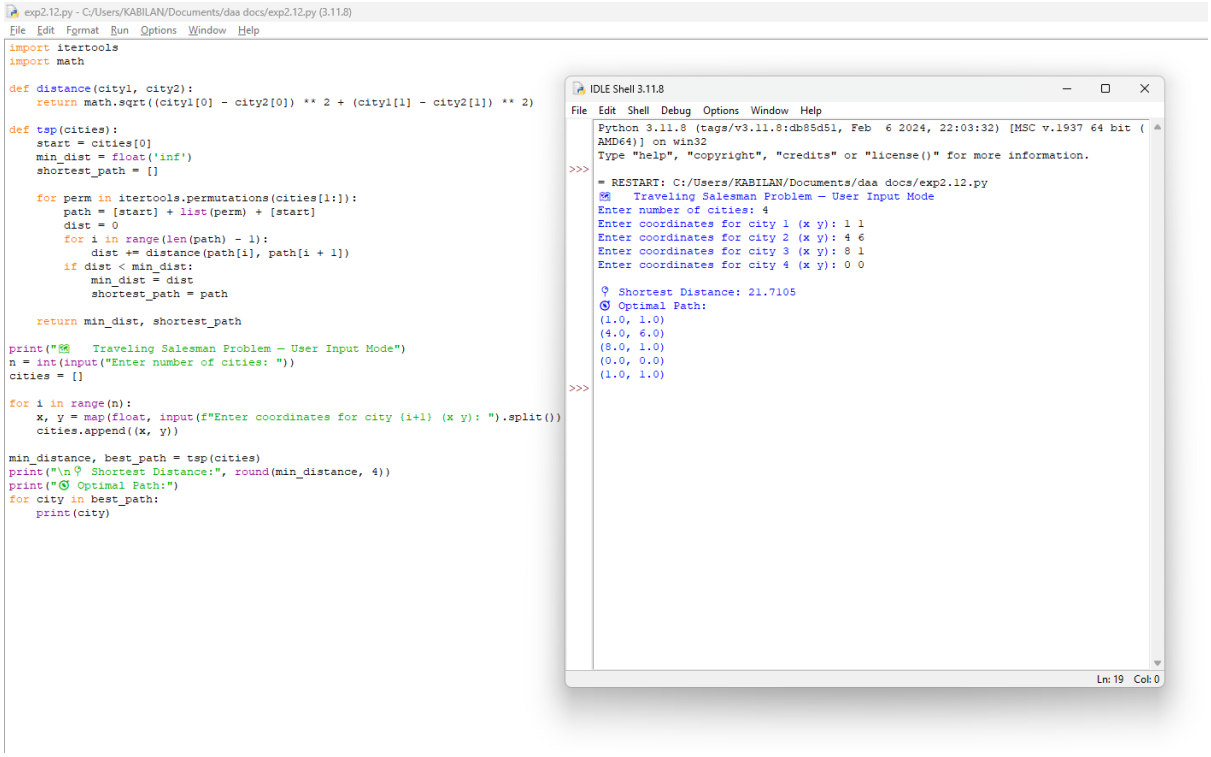
```
        for i in range(len(path) - 1):
            dist += distance(path[i], path[i + 1])
        if dist < min_dist:
            min_dist = dist
            shortest_path = path

    return min_dist, shortest_path

print("Traveling Salesman Problem — User Input Mode")
n = int(input("Enter number of cities: "))
cities = []

for i in range(n):
    x, y = map(float, input(f"Enter coordinates for city {i+1} (x y): ").split())
    cities.append((x, y))

min_distance, best_path = tsp(cities)
print("\n Shortest Distance:", round(min_distance, 4))
print("□ Optimal Path:")
for city in best_path:
    print(city)
```

**Performance Analysis:**

**Time Complexity: O((n−1)!)**

**Space Complexity: O(n)**

## Program Output:




## Result:

The brute force TSP program produces correct shortest routes and distances for the given test cases.