# 2.12 EXHAUSTIVE SEARCH TO SOLVE THE TSP

**Question:**

You are given a list of cities represented by their coordinates. Develop a program that utilizes exhaustive search to solve the TSP. The program should:

1. Define a function distance(city1, city2) to calculate the distance between two cities (e.g., Euclidean distance).
2. Implement a function tsp(cities) that takes a list of cities as input and performs the following:
    - Generate all possible permutations of the cities (excluding the starting city) using itertools.permutations.
    - For each permutation (representing a potential route):
        - Calculate the total distance traveled by iterating through the path and summing the distances between consecutive cities.
        - Keep track of the shortest distance encountered and the corresponding path.
    - Return the minimum distance and the shortest path (including the starting city at the beginning and end).
3. Include test cases with different city configurations to demonstrate the program's functionality. Print the shortest distance and the corresponding path for each test case.

## AIM

To find the shortest route that visits every city exactly once and returns to the starting city using the brute force (exhaustive search) approach.

## ALGORITHM

1. Start
2. Define distance(city1, city2) to calculate Euclidean distance.
3. Define tsp(cities) function and Fix the starting city.
4. Generate all permutations of the remaining cities.
5. For each permutation: Form a complete path including the start city at the beginning and end.
6. Calculate the total travel distance.
7. Update minimum distance and shortest path if the current distance is smaller.
8. Return the shortest path and its distance.
9. End

## PROGRAM

```
import itertools
import math

def distance(p1, p2):
    return math.hypot(p1[0] - p2[0], p1[1] - p2[1])

def tsp(cities):
    start = cities[0]
    min_dist = float('inf')
    best_path = []
    for perm in itertools.permutations(cities[1:]):
        path = [start] + list(perm) + [start]
        dist = sum(distance(path[i], path[i+1]) for i in range(len(path)-1))
        if dist < min_dist:
            min_dist = dist
            best_path = path
    return min_dist, best_path

def run_tsp():
    raw = input("Enter city coordinates as x,y separated by space: ").split()
    cities = [tuple(map(int, p.split(','))) for p in raw]
    dist, path = tsp(cities)
    print("Shortest Distance:", dist)
    print("Shortest Path:", path)

run_tsp()
```

Input:

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

Output:

```
Enter city coordinates as x,y separated by space: 1,2 4,5 7,1 3,6
Shortest Distance: 16.969112047670894
Shortest Path: [(1, 2), (7, 1), (4, 5), (3, 6), (1, 2)]
>>>
```

**RESULT:**

Thus travelling salesman problem using exhaustive search program is successfully executed and the output is verified.

**PERFORMANCE ANALYSIS:**

- **Time Complexity:** O(n!)(permutations of cities)
- **Space Complexity:** O(n)(storing the path)