

PROGRAM TITLE 09

TRAVELLING SALESMAN PROBLEM

AIM:

To Write the python to implement Travelling Salesman Problem.

PROCEDURE:

1. **Subset Enumeration:** Enumerate all subsets of cities, excluding the starting city.
2. **Compute Optimal Subtour:** For each subset, compute the shortest possible tour that starts at the starting city, includes all cities in the subset, and ends at any city in the subset.
3. **Optimal Tour:** The optimal tour is the one with the minimum total distance among all computed subtours.
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

```

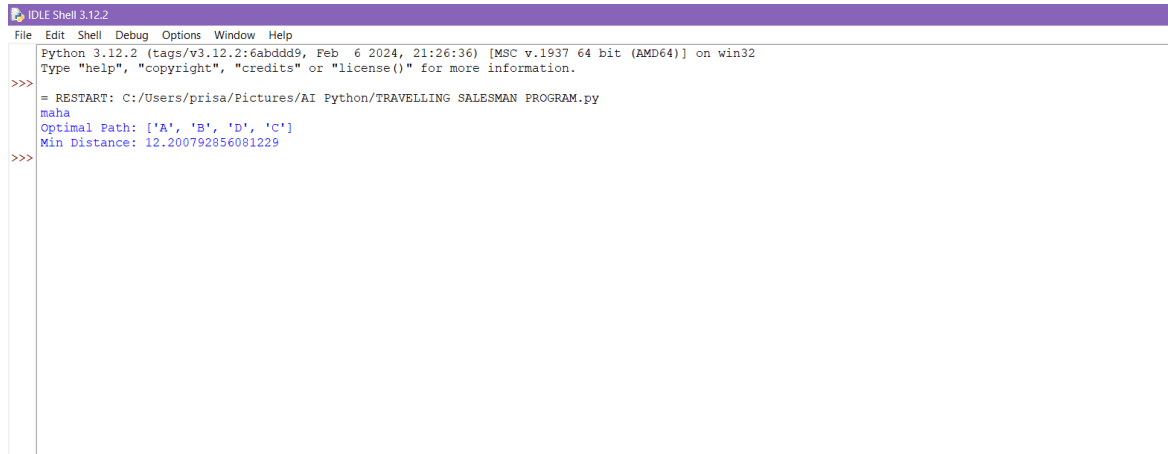
Example usage:

```

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:

A screenshot of the IDLE Shell 3.12.2 interface. The window has a purple title bar and a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following output: Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32, Type "help", "copyright", "credits" or "license()" for more information. The prompt is >>>. The first line of output is = RESTART: C:/Users/prisa/Pictures/AI Python/TRAVELLING SALESMAN PROGRAM.py. The second line is maha. The third line is Optimal Path: ['A', 'B', 'D', 'C']. The fourth line is Min Distance: 12.200792856081229. The prompt is >>>.

```
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/prisa/Pictures/AI Python/TRAVELLING SALESMAN PROGRAM.py
maha
Optimal Path: ['A', 'B', 'D', 'C']
Min Distance: 12.200792856081229
>>>
```

RESULT:

Hence the program been successfully executed and verified.