PROGRAM-9

TRAVELING SALESMAN PROBLEM

AIM:-

To write and execute the python program for the Traveling salesman program.

PROCEDURE:-

- **Import itertools Module**:
  - Import the itertools module to generate all possible permutations of cities.
- **Distance Calculation Functions**:
  - Define the calculate_distance function to calculate the distance between two cities using the given distance matrix.
  - Define the total_distance function to calculate the total distance of a path by summing up the distances between consecutive cities and returning to the starting city.
- **TSP Algorithm**:
  - Define the tsp function to find the optimal solution for the TSP using brute force.
  - Initialize variables min_distance and optimal_path to store the minimum distance and the optimal path, respectively.
  - Iterate through all permutations of cities and calculate the total distance for each permutation.
  - Update min_distance and optimal_path if a shorter path is found.
- **Example Usage**:
  - Define the list of cities and the distance matrix.
  - Call the tsp function with the cities and distances to find the optimal path and minimum distance.
  - Print the optimal path and minimum distance

CODING:-

```
import itertools

def calculate_distance(city1, city2, distances):

    return distances[city1][city2]
```

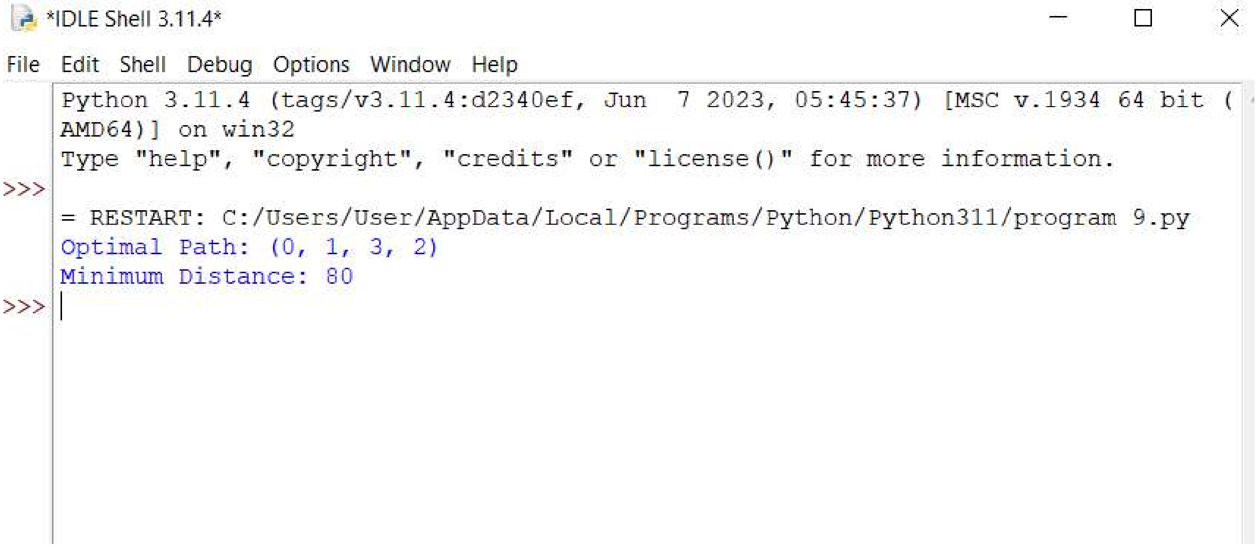
```
def total_distance(path, distances):
    total = 0
    for i in range(len(path) - 1):
        total += calculate_distance(path[i], path[i + 1], distances)
    total += calculate_distance(path[-1], path[0], distances)  # Return to starting city
    return total
def tsp(cities, distances):
    min_distance = float('inf')
    optimal_path = []
    for perm in itertools.permutations(cities):
        distance = total_distance(perm, distances)
        if distance < min_distance:
            min_distance = distance
            optimal_path = perm
    return min_distance, optimal_path
cities = [0, 1, 2, 3]
distances = {
    0: {0: 0, 1: 10, 2: 15, 3: 20},
    1: {0: 10, 1: 0, 2: 35, 3: 25},
    2: {0: 15, 1: 35, 2: 0, 3: 30},
    3: {0: 20, 1: 25, 2: 30, 3: 0}
}

min_distance, optimal_path = tsp(cities, distances)

print("Optimal Path:", optimal_path)

print("Minimum Distance:", min_distance)
```

OUTPUT:-



```
    Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (
    AMD64)] on win32
    Type "help", "copyright", "credits" or "license()" for more information.
>>>
    = RESTART: C:/Users/User/AppData/Local/Programs/Python/Python311/program 9.py
    Optimal Path: (0, 1, 3, 2)
    Minimum Distance: 80
>>> |
```

RESULT:-

Hence the program has been successfully executed and verified.