TASK:5

Implementation of **Ant Colony Optimization** to Optimize Ride-Sharing Trip Duration using Python by following constraints.

Aim: To Implement Ant Colony Optimization to Optimize Ride-Sharing Trip Duration using Python.

Algorithm:

Step 1:

[Initialization] t=0;NC=0;for each edge (I,j),initialize trail intensity.

Step 2:[starting node]

For each ant k:place ant k on a randomly chosen city and store this information in tablet.

Step 3: Build a tour for each ant.

Step 4:global update of trail.

Step 5: termination conditions,memorize the shortest tour found to this point.

Program:

```
import numpy as np
# Distance matrix
d = np.array([
   [0, 10, 12, 11, 14],
   [10, 0, 13, 15, 8],
   [12, 13, 0, 9, 14],
   [11, 15, 9, 0, 16],
   [14, 8, 14, 16, 0]
])
# Parameters
iteration = 100
n_ants = 5
n_citys = 5
e = 0.5       # evaporation rate
alpha = 1     # pheromone importance factor
beta = 2      # visibility importance factor
# Visibility matrix: 1 / distance
visibility = 1 / d.astype(float)
visibility[visibility == np.inf] = 0  # Handle
division by zero
```

```
# Initialize pheromone levels
pheromone = 0.1 * np.ones((n_citys, n_citys))
# Initialize route array (n_ants x n_citys+1) for
return to start city
route = np.ones((n_ants, n_citys + 1),
dtype=int)
for ite in range(iteration):
    route[:, 0] = 1  # Start all ants at city 1
    for i in range(n_ants):  # For each ant
        temp_visibility = np.array(visibility)  #
Copy visibility matrix
        visited = set([0])  # Start city index 0
        for j in range(n_citys - 1):
            current_city = route[i, j] - 1
            temp_visibility[:, current_city] = 0  #
Can't go back to current city
            # Pheromone and visibility factors
            p_feature = pheromone[current_city, :]
** alpha
            v_feature = temp_visibility[current_city,
:] ** beta
            combined_feature = p_feature *
v_feature
            combined_feature[list(visited)] = 0  #
Avoid visited cities
            total = np.sum(combined_feature)
            if total == 0:  # No available city
                break
            probs = combined_feature / total
            cum_probs = np.cumsum(probs)
            r = np.random.random()
            # Choose next city
            next_city = np.where(cum_probs >=
r)[0][0]
            route[i, j + 1] = next_city + 1
            visited.add(next_city)
```

```
        # Add last unvisited city if any remain
        if len(visited) < n_citys:
            remaining_cities = set(range(n_citys)) -
visited
            last_city = remaining_cities.pop()
            route[i, n_citys - 1] = last_city + 1
        route[i, n_citys] = 1  # Return to start city
    # Calculate distance for all ants
    dist_cost = np.zeros(n_ants)
    for i in range(n_ants):
        s = 0
        for j in range(n_citys):
            s += d[route[i, j] - 1, route[i, j + 1] - 1]
        dist_cost[i] = s
    dist_min_loc = np.argmin(dist_cost)
    dist_min_cost = dist_cost[dist_min_loc]
    best_route = route[dist_min_loc, :]
    # Evaporate pheromone
    pheromone = (1 - e) * pheromone
    # Update pheromone based on routes
    for i in range(n_ants):
        dt = 1 / dist_cost[i]
        for j in range(n_citys):
            from_city = route[i, j] - 1
            to_city = route[i, j + 1] - 1
            pheromone[from_city, to_city] += dt
            pheromone[to_city, from_city] += dt  #
Undirected graph
# Print results
print('Route of all the ants at the end:')
print(route)
print()
print('Best path:', best_route)
print('Cost of the best path:', int(dist_min_cost))
```

Output:

```
>>
====================== RESTART: C:/Users/mahes/VTU26520.py

Warning (from warnings module):
  File "C:/Users/mahes/VTU26520.py", line 21
    visibility = 1 / d.astype(float)
RuntimeWarning: divide by zero encountered in divide
Route of all the ants at the end:
[[1 2 5 3 4 1]
 [1 2 5 3 4 1]
 [1 2 5 3 4 1]
 [1 4 3 5 2 1]
 [1 4 3 5 2 1]]

Best path: [1 2 5 3 4 1]
Cost of the best path: 52
>>
```

Result:

Thus the Implementation of Ant Colony Optimization to Optimize Ride-Sharing Trip Duration using Python was successfully executed and output was verified.