USN : 1BM22CS253

LAB-3 : Ant Colony Op miza on for the Traveling Salesman Problem:

CODE:

```
import numpy as np import
matplotlib.pyplot as plt


# 1. Define the Problem: Create a set of ci es with their coordinates ci
es = np.array([
    [0, 0],  # City 0
    [1, 5],  # City 1
    [5, 1],  # City 2
    [6, 4],  # City 3
[7, 8],  # City 4
])

# Calculate the distance matrix between each pair of ci es
def calculate_distances(ci es):       num_ci es = len(ci es)
distances = np.zeros((num_ci   es, num_ci es))


    for i in range(num_ci es):
for j in range(num_ci es):
            distances[i][j] = np.linalg.norm(ci es[i] - ci es[j])

    return distances

distances = calculate_distances(ci es)
```

```
# 2. Ini alize Parameters
num_ants = 10 num_ci es = len(ci es) alpha = 1.0  #
Influence of pheromone beta = 5.0    # Influence of
heuris c (inverse distance) rho = 0.5    # Evapora on
rate num_itera ons = 30 ini al_pheromone = 1.0


# Pheromone matrix ini aliza on pheromone = np.ones((num_ci es,
num_ci es)) * ini al_pheromone


# 3. Heuris c informa on (Inverse of distance) def heuris
c(distances):       with np.errstate(divide='ignore'):  # Ignore
division by zero        return 1 / distances


eta = heuris c(distances)


# 4. Choose next city probabilis cally based on pheromone and heuris c info def
choose_next_city(pheromone, eta, visited):

   probs = []     for j in
range(num_ci es):

       if j not in visited:
           pheromone_ij = pheromone[visited[-1], j] ** alpha
heuris c_ij = eta[visited[-1], j] ** beta
probs.append(pheromone_ij * heuris c_ij)

       else:
           probs.append(0)     probs = np.array(probs)     return
np.random.choice(range(num_ci es), p=probs / probs.sum()) #
Construct solu on for a single ant def construct_solu on(pheromone,
eta):     tour = [np.random.randint(0, num_ci es)]     while len(tour) <
```

num_ci es: next_city = choose_next_city(pheromone, eta, tour)

tour.append(next_city)   return tour

5. Update pheromones a er all ants have constructed their tours def

update_pheromones(pheromone, all_tours, distances, best_tour):

    pheromone *= (1 - rho) # Evaporate pheromones

    # Add pheromones for each ant's tour

for tour in all_tours:

        tour_length = sum([distances[tour[i], tour[i + 1]] for i in range(-1, num_ci es - 1)])

for i in range(-1, num_ci es - 1):

            pheromone[tour[i], tour[i + 1]] += 1.0 / tour_length

    # Increase pheromones on the best tour   best_length = sum([distances[best_tour[i],

best_tour[i + 1]] for i in range(-1, num_ci es - 1)])   for i in range(-1, num_ci es - 1):

        pheromone[best_tour[i], best_tour[i + 1]] += 1.0 / best_length

6. Main ACO Loop: Iterate over mul ple itera ons to find the best solu on def

run_aco(distances, num_itera ons):

    pheromone = np.ones((num_ci es, num_ci es)) * ini al_pheromone

    best_tour = None

best_length = float('inf')

    for itera on in range(num_itera ons):

        all_tours = [construct_solu on(pheromone, eta) for _ in range(num_ants)]
        all_lengths = [sum([distances[tour[i], tour[i + 1]] for i in range(-1, num_ci es - 1)]) for tour in all_tours]

```
        current_best_length = min(all_lengths)         current_best_tour =
all_tours[all_lengths.index(current_best_length)]


        if current_best_length < best_length:
best_length = current_best_length             best_tour
= current_best_tour


        update_pheromones(pheromone, all_tours, distances, best_tour)


        print(f"Itera on {itera on + 1}, Best Length: {best_length}")


    return best_tour, best_length


# Run the ACO algorithm best_tour, best_length =
run_aco(distances, num_itera ons)


# 7. Output the Best Solu on print(f"Best
Tour: {best_tour}") print(f"Best Tour
Length: {best_length}")


# 8. Plot the Best Route def
plot_route(ci es, best_tour):

    plt.figure(figsize=(8, 6))
for i in range(len(ci es)):

        plt.sca er(ci es[i][0], ci es[i][1], color='red')
plt.text(ci es[i][0], ci es[i][1], f"City {i}", fontsize=12)
```

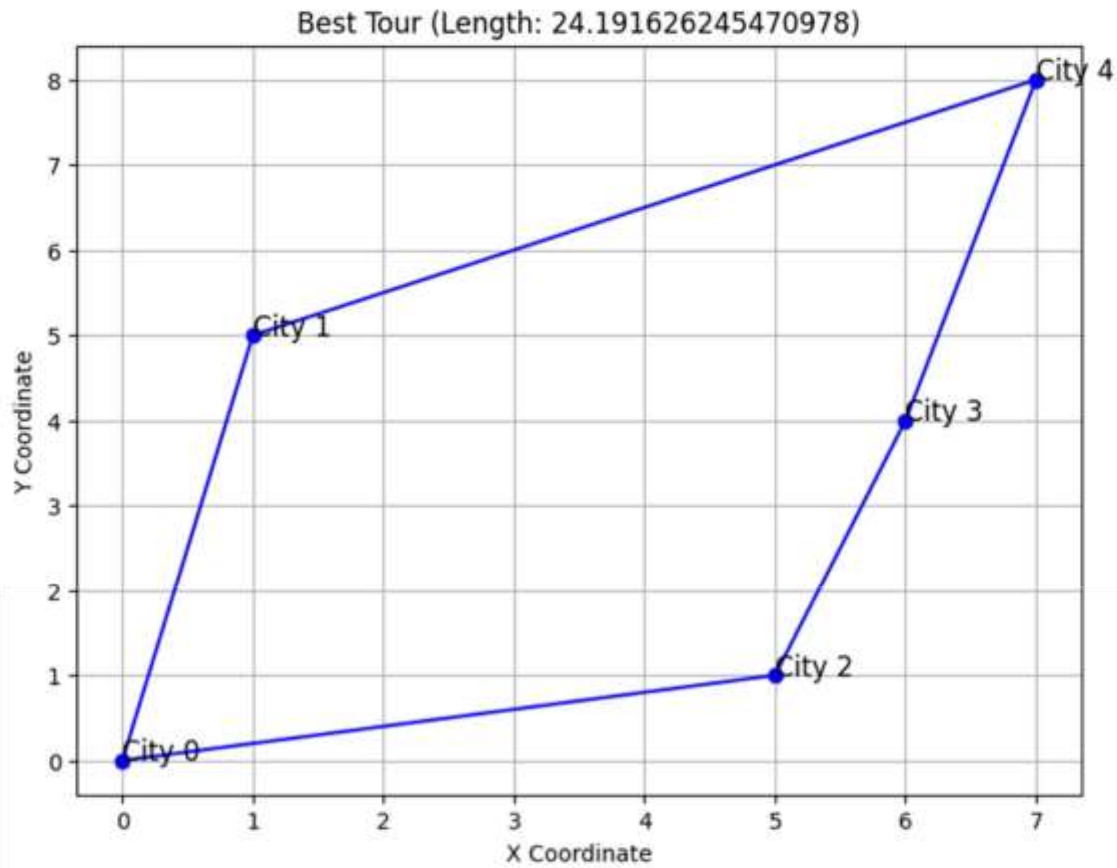
```
    # Plot the tour as lines connec ng the ci es
    tour_ci es = np.array([ci es[i] for i in best_tour] + [ci es[best_tour[0]]])  # Complete the loop by
returning to the start     plt.plot(tour_ci es[:, 0], tour_ci es[:, 1], linestyle='-', marker='o',
color='blue')


    plt. tle(f"Best Tour (Length:
{best_length})")     plt.xlabel("X Coordinate")
plt.ylabel("Y Coordinate")     plt.grid(True)
plt.show()


# Call the plot func on plot_route(ci
es, best_tour)
```

OUTPUT:

```
Iteration 1, Best Length: 24.191626245470978
Iteration 2, Best Length: 24.191626245470978
Iteration 3, Best Length: 24.191626245470978
Iteration 4, Best Length: 24.191626245470978
Iteration 5, Best Length: 24.191626245470978
Iteration 6, Best Length: 24.191626245470978
Iteration 7, Best Length: 24.191626245470978
Iteration 8, Best Length: 24.191626245470978
Iteration 9, Best Length: 24.191626245470978
Iteration 10, Best Length: 24.191626245470978
Iteration 11, Best Length: 24.191626245470978
Iteration 12, Best Length: 24.191626245470978
Iteration 13, Best Length: 24.191626245470978
Iteration 14, Best Length: 24.191626245470978
Iteration 15, Best Length: 24.191626245470978
Iteration 16, Best Length: 24.191626245470978
Iteration 17, Best Length: 24.191626245470978
Iteration 18, Best Length: 24.191626245470978
Iteration 19, Best Length: 24.191626245470978
Iteration 20, Best Length: 24.191626245470978
Iteration 21, Best Length: 24.191626245470978
Iteration 22, Best Length: 24.191626245470978
Iteration 23, Best Length: 24.191626245470978
Iteration 24, Best Length: 24.191626245470978
Iteration 25, Best Length: 24.191626245470978
Iteration 26, Best Length: 24.191626245470978
Iteration 27, Best Length: 24.191626245470978
Iteration 28, Best Length: 24.191626245470978
Iteration 29, Best Length: 24.191626245470978
Iteration 30, Best Length: 24.191626245470978
Best Tour: [4, 3, 2, 0, 1]
Best Tour Length: 24.191626245470978
```



Best Tour (Length: 24.191626245470978)
City 0
City 1
City 2
City 3
City 4
Y Coordinate
X Coordinate