USN : 1BM22CS235

LAB-3 : Ant Colony Optmizaton for the Traveling Salesman Problem:

CODE:

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




# 1. Define the Problem: Create a set of cites with their coordinates
cites = 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 cites
def calculate_distances(cites):
    num_cites = len(cites)
    distances = np.zeros((num_cites, num_cites))

    for i in range(num_cites):
        for j in range(num_cites):
            distances[i][j] = np.linalg.norm(cites[i] - cites[j])

    return distances

distances =

calculate_distances(cites)
# 2. Initalize

Parameters
```

```
num_ants = 10 num_cites = len(cites) alpha = 1.0 #
Influence of pheromone beta = 5.0 # Influence of
heuristc (inverse distance)

rho = 0.5   # Evaporaton rate
num_iteratons = 30
inital_pheromone = 1.0

# Pheromone matrix initalizaton
pheromone = np.ones((num_cites, num_cites)) * inital_pheromone

# 3. Heuristc informaton (Inverse of distance)
def heuristc(distances):
    with np.errstate(divide='ignore'): # Ignore division by zero
        return 1 / distances

eta = heuristc(distances)

# 4. Choose next city probabilistcally based on pheromone and heuristc info
def choose_next_city(pheromone, eta, visited):
    probs = []
    for j in range(num_cites):
        if j not in visited:
            pheromone_ij = pheromone[visited[-1], j] ** alpha
            heuristc_ij = eta[visited[-1], j] ** beta
            probs.append(pheromone_ij * heuristc_ij)
        else:
            probs.append(0)
    probs = np.array(probs)
    return np.random.choice(range(num_cites), p=probs / probs.sum())
```

```
# Construct soluton for a single ant def
construct_soluton(pheromone, eta):
    tour = [np.random.randint(0, num_cites)]
    while len(tour) < num_cites:
        next_city = choose_next_city(pheromone, eta, tour)
        tour.append(next_city)
    return tour

# 5. Update pheromones aŌer 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_cites - 1)])
        for i in range(-1, num_cites - 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_cites -
    1)])
    for i in range(-1, num_cites - 1):
        pheromone[best_tour[i], best_tour[i + 1]] += 1.0 / best_length

# 6. Main ACO Loop: Iterate over multple iteratons to find the best soluton
def run_aco(distances, num_iteratons):
    pheromone = np.ones((num_cites, num_cites)) * inital_pheromone
    best_tour = None
    best_length = float('inf')

    for iteraton in range(num_iteratons):
        all_tours = [construct_soluton(pheromone, eta) for _ in range(num_ants)]
```

```
        all_lengths = [sum([distances[tour[i], tour[i + 1]] for i in range(-1, num_cites - 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"Iteraton {iteraton + 1}, Best Length: {best_length}")

    return best_tour, best_length

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

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

# 8. Plot the Best Route
def plot_route(cites, best_tour):
    plt.figure(figsize=(8, 6))
    for i in range(len(cites)):
        plt.scaΣer(cites[i][0], cites[i][1], color='red')
        plt.text(cites[i][0], cites[i][1], f"City {i}", fontsize=12)

    # Plot the tour as lines connectng the cites
```

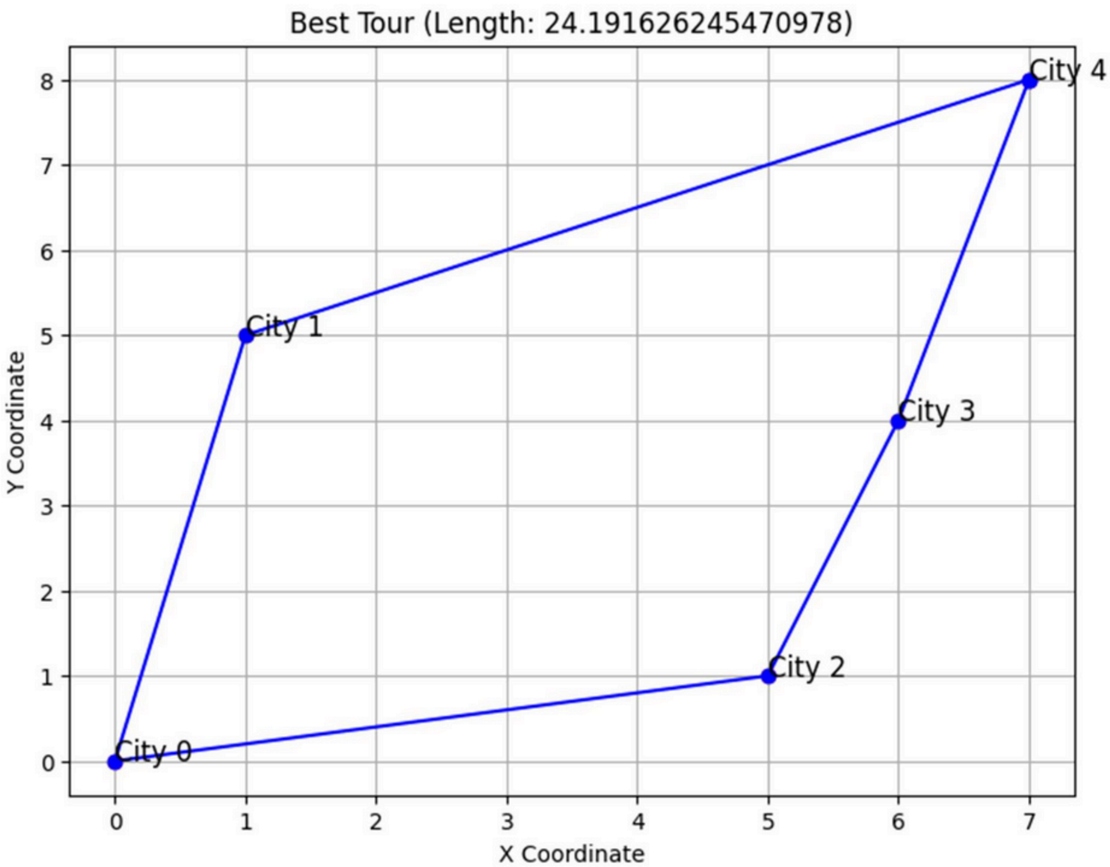
```
 tour_cites = np.array([cites[i] for i in best_tour] + [cites[best_tour[0]]]) # Complete the loop by
returning to the start
    plt.plot(tour_cites[:, 0], tour_cites[:, 1], linestyle='-', marker='o', color='blue')


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


# Call the plot functon
plot_route(cites, 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
X Coordinate
Y Coordinate
0
1
2
3
4
5
6
7
8