LAB 3
ANT COLONY PROBLEM

1BM22CS263
SHREE VARNA M

CODE:

```
import numpy as np
import random

class AntColonyOptimization:
    def __init__(self, cities, num_ants, alpha, beta, rho, iterations):
        self.cities = cities
        self.num_ants = num_ants
        self.alpha = alpha          # importance of pheromone
        self.beta = beta            # importance of heuristic (inverse distance)
        self.rho = rho              # pheromone evaporation rate
        self.iterations = iterations
        self.num_cities = len(cities)

        # Initialize pheromone levels
        self.pheromone = np.ones((self.num_cities, self.num_cities))
        self.distances = self.calculate_distances()

        # Track the best route found
        self.best_route = None
        self.best_distance = float('inf')

    def calculate_distances(self):
        """ Calculate distance matrix between cities """
        distances = np.zeros((self.num_cities, self.num_cities))
        for i in range(self.num_cities):
            for j in range(self.num_cities):
                if i != j:
                    distances[i][j] = np.linalg.norm(np.array(self.cities[i]) - np.array(self.cities[j]))
        return distances

    def probability(self, i, j, visited):
        """ Calculate probability of moving from city i to city j """
        if j in visited:
            return 0
        pheromone = self.pheromone[i][j] ** self.alpha
        heuristic = (1.0 / self.distances[i][j]) ** self.beta if self.distances[i][j] != 0 else 0
        return pheromone * heuristic

    def construct_solution(self):
        """ Construct a route for each ant based on probabilities """
```

```
        all_routes = []
        for _ in range(self.num_ants):
            route = [random.randint(0, self.num_cities - 1)]
            while len(route) < self.num_cities:
                current_city = route[-1]
                probabilities = [self.probability(current_city, j, route) for j in
range(self.num_cities)]
                total_prob = sum(probabilities)
                probabilities = [p / total_prob if total_prob > 0 else 0 for p in probabilities]
                next_city = np.random.choice(range(self.num_cities), p=probabilities)
                route.append(next_city)
            route.append(route[0])  # return to starting city
            all_routes.append(route)
        return all_routes

    def route_distance(self, route):
        """ Calculate total distance of a route """
        return sum([self.distances[route[i]][route[i + 1]] for i in range(len(route) - 1)])

    def update_pheromones(self, all_routes):
        """ Update pheromone levels based on routes found """
        self.pheromone *= (1 - self.rho)  # evaporate some pheromone
        for route in all_routes:
            distance = self.route_distance(route)
            if distance < self.best_distance:
                self.best_distance = distance
                self.best_route = route
            pheromone_contribution = 1 / distance
            for i in range(len(route) - 1):
                self.pheromone[route[i]][route[i + 1]] += pheromone_contribution

    def run(self):
        """ Execute the ACO algorithm """
        for _ in range(self.iterations):
            all_routes = self.construct_solution()
            self.update_pheromones(all_routes)

        return self.best_route, self.best_distance

# Example usage
cities = [(0, 0), (1, 5), (5, 1), (6, 6), (8, 3)]  # example set of city coordinates
aco = AntColonyOptimization(cities, num_ants=10, alpha=1, beta=5, rho=0.5,
iterations=100)
best_route, best_distance = aco.run()

print("Best route:", best_route)
print("Best distance:", best_distance)
```

OUTPUT :

```
Best route: [4, 3, 1, 0, 2, 4]
Best distance: 22.50816109170633
```