**LAB – 4**

**ANT COLONY OPTIMISATION**

**(APPLICATION USED TRAVELLING SALESMAN)**

**CODE :**

import random

def initialize\_pheromone\_matrix(num\_cities, initial\_pheromone):

    return [[initial\_pheromone for \_ in range(num\_cities)] for \_ in range(num\_cities)]

def calculate\_probability(current\_city, unvisited\_cities, pheromone, distance):

    pheromone\_list = []

    heuristic\_list = []

    for city in unvisited\_cities:

        pheromone\_list.append(pheromone[current\_city][city])

        heuristic\_list.append(1.0 / distance[current\_city][city])

    total = sum([p \* h for p, h in zip(pheromone\_list, heuristic\_list)])

    probabilities = [(pheromone[current\_city][city] \* (1.0 / distance[current\_city][city])) / total for city in unvisited\_cities]

    return probabilities

def select\_next\_city(unvisited\_cities, probabilities):

    return random.choices(unvisited\_cities, weights=probabilities)[0]

def update\_pheromone(pheromone, ants\_tours, ants\_lengths, evaporation\_rate, Q):

    num\_cities = len(pheromone)

    for i in range(num\_cities):

        for j in range(num\_cities):

            pheromone[i][j] \*= (1 - evaporation\_rate)

            if pheromone[i][j] < 1e-10:

                pheromone[i][j] = 1e-10

    for k in range(len(ants\_tours)):

        tour = ants\_tours[k]

        length = ants\_lengths[k]

        deposit = Q / length

        for i in range(len(tour)-1):

            a = tour[i]

            b = tour[i+1]

            pheromone[a][b] += deposit

            pheromone[b][a] += deposit

def calculate\_tour\_length(tour, distance):

    length = 0.0

    for i in range(len(tour)-1):

        length += distance[tour[i]][tour[i+1]]

    return length

def ant\_colony\_optimization(distance, num\_ants=10, num\_iterations=10, evaporation\_rate=0.5, Q=100, initial\_pheromone=1.0):

    num\_cities = len(distance)

    pheromone = initialize\_pheromone\_matrix(num\_cities, initial\_pheromone)

    best\_tour = None

    best\_length = float('inf')

    for iteration in range(num\_iterations):

        ants\_tours = []

        ants\_lengths = []

        for \_ in range(num\_ants):

            start\_city = random.randint(0, num\_cities - 1)

            tour = [start\_city]

            unvisited\_cities = set(range(num\_cities))

            unvisited\_cities.remove(start\_city)

            current\_city = start\_city

            while unvisited\_cities:

                probabilities = calculate\_probability(current\_city, list(unvisited\_cities), pheromone, distance)

                next\_city = select\_next\_city(list(unvisited\_cities), probabilities)

                tour.append(next\_city)

                unvisited\_cities.remove(next\_city)

                current\_city = next\_city

            tour.append(start\_city)

            tour\_length = calculate\_tour\_length(tour, distance)

            ants\_tours.append(tour)

            ants\_lengths.append(tour\_length)

            if tour\_length < best\_length:

                best\_length = tour\_length

                best\_tour = tour

        update\_pheromone(pheromone, ants\_tours, ants\_lengths, evaporation\_rate, Q)

        print(f"Iteration {iteration+1}/{num\_iterations}, Best tour length: {best\_length}")

    return best\_tour, best\_length

distance\_matrix = [

    [0, 10, 15, 20],

    [10, 0, 35, 25],

    [15, 35, 0, 30],

    [20, 25, 30, 0]

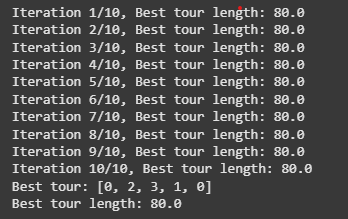
]

best\_tour, best\_length = ant\_colony\_optimization(distance\_matrix)

print("Best tour:", best\_tour)

print("Best tour length:", best\_length)

**OUTPUT :**

****