**ANT COLONY OPTIMIZATION FOR TRAVELLING SALES PERSON**

import random

import math

class ACO\_TSP:

def \_\_init\_\_(self, dist\_matrix, n\_ants=5, n\_iter=50, alpha=1, beta=2, rho=0.5, Q=100):

self.dist = dist\_matrix

self.n = len(dist\_matrix)

self.n\_ants = n\_ants

self.n\_iter = n\_iter

self.alpha = alpha

self.beta = beta

self.rho = rho

self.Q = Q

# Initialize pheromone trails

self.pheromone = [[1 for \_ in range(self.n)] for \_ in range(self.n)]

self.best\_length = math.inf

self.best\_tour = []

def construct\_solution(self):

all\_tours = []

all\_lengths = []

for \_ in range(self.n\_ants):

start = random.randint(0, self.n - 1)

tour = [start]

while len(tour) < self.n:

i = tour[-1]

probs = []

total = 0

for j in range(self.n):

if j not in tour and self.dist[i][j] > 0: # ✅ skip self-loops

tau = self.pheromone[i][j] \*\* self.alpha

eta = (1 / self.dist[i][j]) \*\* self.beta

total += tau \* eta

probs.append((j, tau \* eta))

# Roulette wheel selection

r = random.random() \* total

cum\_sum = 0

for j, p in probs:

cum\_sum += p

if cum\_sum >= r:

tour.append(j)

break

all\_tours.append(tour)

all\_lengths.append(self.tour\_length(tour))

return all\_tours, all\_lengths

def tour\_length(self, tour):

length = 0

for i in range(len(tour)):

j = (i + 1) % self.n # wrap around to start

length += self.dist[tour[i]][tour[j]]

return length

def update\_pheromones(self, all\_tours, all\_lengths):

for i in range(self.n):

for j in range(self.n):

self.pheromone[i][j] \*= (1 - self.rho)

for k, tour in enumerate(all\_tours):

Lk = all\_lengths[k]

for i in range(len(tour)):

j = (i + 1) % self.n

a, b = tour[i], tour[j]

self.pheromone[a][b] += self.Q / Lk

self.pheromone[b][a] += self.Q / Lk

def run(self):

for \_ in range(self.n\_iter):

tours, lengths = self.construct\_solution()

for k in range(len(tours)):

if lengths[k] < self.best\_length:

self.best\_length = lengths[k]

self.best\_tour = tours[k]

self.update\_pheromones(tours, lengths)

return self.best\_tour, self.best\_length

if \_\_name\_\_ == "\_\_main\_\_":

n = int(input("Enter number of cities: "))

print("Enter distance matrix (row by row, space-separated):")

dist\_matrix = []

for \_ in range(n):

row = list(map(int, input().split()))

dist\_matrix.append(row)

ants = int(input("Enter number of ants: "))

iterations = int(input("Enter number of iterations: "))

aco = ACO\_TSP(dist\_matrix, n\_ants=ants, n\_iter=iterations)

best\_tour, best\_length = aco.run()

print("\nBest Tour (order of cities):", best\_tour)

print("Best Length:", best\_length)

