**LAB – 5**

CODE : (USING DIJIKSTRAS ALGORITHM)

import numpy as np

import random

class AntColony:

def \_\_init\_\_(self, graph, num\_ants, num\_iterations, decay, alpha=1, beta=2, q=100):

self.graph = graph

self.num\_nodes = len(graph)

self.num\_ants = num\_ants

self.num\_iterations = num\_iterations

self.decay = decay

self.alpha = alpha

self.beta = beta

self.q = q

self.pheromone = np.ones((self.num\_nodes, self.num\_nodes))

def run(self, start, end):

best\_path = None

best\_distance = float('inf')

for iteration in range(self.num\_iterations):

all\_paths = []

all\_distances = []

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

path, distance = self.construct\_path(start, end)

all\_paths.append(path)

all\_distances.append(distance)

if distance < best\_distance:

best\_distance = distance

best\_path = path

self.update\_pheromones(all\_paths, all\_distances)

return best\_path, best\_distance

def construct\_path(self, start, end):

path = [start]

visited = set(path)

current = start

distance = 0

while current != end:

next\_node = self.select\_next\_node(current, visited)

if next\_node is None:

return path, float('inf')

path.append(next\_node)

visited.add(next\_node)

distance += self.graph[current][next\_node]

current = next\_node

return path, distance

def select\_next\_node(self, current, visited):

pheromone = self.pheromone[current]

distances = self.graph[current]

allowed\_nodes = [i for i in range(self.num\_nodes) if i not in visited and distances[i] != np.inf]

if not allowed\_nodes:

return None

pheromone\_values = np.array([pheromone[i] for i in allowed\_nodes])

heuristic\_values = np.array([1 / distances[i] for i in allowed\_nodes])

probabilities = (pheromone\_values \*\* self.alpha) \* (heuristic\_values \*\* self.beta)

probabilities /= probabilities.sum()

next\_node = random.choices(allowed\_nodes, weights=probabilities)[0]

return next\_node

def update\_pheromones(self, all\_paths, all\_distances):

self.pheromone \*= (1 - self.decay)

for path, dist in zip(all\_paths, all\_distances):

if dist == float('inf'):

continue

deposit = self.q / dist

for i in range(len(path) - 1):

self.pheromone[path[i]][path[i+1]] += deposit

self.pheromone[path[i+1]][path[i]] += deposit # Undirected graph

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

# Create a symmetric matrix for 10 cities with random distances between 1 and 100

num\_cities = 10

np.random.seed(42) # for reproducibility

# Random distances, 0 on diagonal, symmetric matrix

dist\_matrix = np.random.randint(1, 101, size=(num\_cities, num\_cities))

dist\_matrix = (dist\_matrix + dist\_matrix.T) / 2

np.fill\_diagonal(dist\_matrix, 0)

print("Distance Matrix (10 cities):")

print(dist\_matrix.astype(int))

ant\_colony = AntColony(dist\_matrix, num\_ants=20, num\_iterations=200, decay=0.1, alpha=1, beta=5, q=100)

start\_city = 0

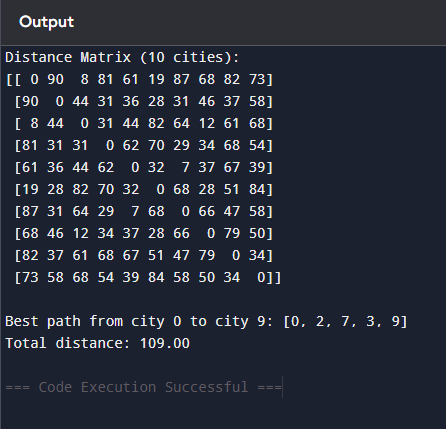
end\_city = 9

best\_path, best\_distance = ant\_colony.run(start\_city, end\_city)

print(f"\nBest path from city {start\_city} to city {end\_city}: {best\_path}")

print(f"Total distance: {best\_distance:.2f}")

**OUTPUT:**

****