**Experiment 7 (A.I)**

**Code:**

import random

import numpy as np

import networkx as nx

coordinate = np.array([

[1, 2], [30, 21], [56, 23], [8, 18], [20, 50],

[3, 4], [11, 6], [6, 7], [15, 20], [10, 9], [12, 12]

])

def generate\_matrix(coordinate):

matrix = []

for i in range(len(coordinate)):

for j in range(len(coordinate)):

p = np.linalg.norm(coordinate[i] - coordinate[j])

matrix.append(p)

matrix = np.reshape(matrix, (len(coordinate), len(coordinate)))

return matrix

# Finds a random solution

def solution(matrix):

points = list(range(0, len(matrix)))

solution = []

for i in range(0, len(matrix)):

random\_point = points[random.randint(0, len(points) - 1)]

solution.append(random\_point)

points.remove(random\_point)

return solution

# Calculate the path length based on the given solution

def path\_length(matrix, solution):

cycle\_length = 0

for i in range(0, len(solution)):

cycle\_length += matrix[solution[i]][solution[i - 1]]

return cycle\_length

def neighbors(matrix, solution):

neighbors = []

for i in range(len(solution)):

for j in range(i + 1, len(solution)):

neighbor = solution.copy()

neighbor[i], neighbor[j] = neighbor[j], neighbor[i]

neighbors.append(neighbor)

# Assume that the first neighbor is the best

best\_neighbor = neighbors[0]

best\_path = path\_length(matrix, best\_neighbor)

for neighbor in neighbors:

current\_path = path\_length(matrix, neighbor)

if current\_path < best\_path:

best\_path = current\_path

best\_neighbor = neighbor

return best\_neighbor, best\_path

# Hill Climbing algorithm

def hill\_climbing(coordinate):

matrix = generate\_matrix(coordinate)

current\_solution = solution(matrix)

current\_path = path\_length(matrix, current\_solution)

neighbor = neighbors(matrix, current\_solution)[0]

best\_neighbor, best\_neighbor\_path = neighbors(matrix, neighbor)

while best\_neighbor\_path < current\_path:

current\_solution = best\_neighbor

current\_path = best\_neighbor\_path

neighbor = neighbors(matrix, current\_solution)[0]

best\_neighbor, best\_neighbor\_path = neighbors(matrix, neighbor)

return current\_path, current\_solution

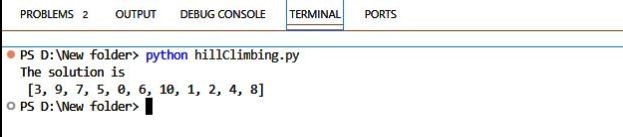
# Run the algorithm

final\_path\_length, final\_solution = hill\_climbing(coordinate)

print("The best path length found is:", final\_path\_length)

print("The solution path is:", final\_solution)

**Output:**

****