# Name: Shantanu Rohile

# Roll No: 53

*# ==============================================*

# AI Practical 01

# Problem Statement:

# Write a program to implement the Travelling Salesperson Problem (TSP) # using appropriate heuristic and search strategy

*# ==============================================*

import numpy as np #

# Function to calculate Euclidean distance # between two cities (2D coordinates)

*#*

def calculate\_distance(city1, city2):

return np.linalg.norm(np.array(city1) - np.array(city2))

*#*

# Nearest Neighbor Algorithm to generate a tour # for the Travelling Salesman Problem (TSP)

*#*

def nearest\_neighbor\_algorithm(city\_coordinates):

num\_cities = len(city\_coordinates) # Total number of cities unvisited\_cities = set(range(num\_cities)) # Cities that are yet to be visited tour = [] # Final tour (list of city indices)

# Start from a random city

current\_city = np.random.choice(list(unvisited\_cities)) unvisited\_cities.remove(current\_city) # Mark the starting city as visited tour.append(current\_city)

# Loop until all cities are visited while unvisited\_cities:

# Find the nearest unvisited city from the current city nearest\_city = min(

unvisited\_cities,

key=lambda city: calculate\_distance(city\_coordinates[current\_city], city\_coordinates[city])

*)*

unvisited\_cities.remove(nearest\_city) # Mark as visited tour.append(nearest\_city)

current\_city = nearest\_city # Move to the next city return tour

*#*

# Function to calculate the total distance

# of the complete tour including return to start #

def calculate\_total\_distance(tour, city\_coordinates): total\_distance = 0

for i in range(len(tour)):

total\_distance += calculate\_distance( city\_coordinates[tour[i]],

city\_coordinates[tour[(i + 1) % len(tour)]] # Wrap around to the first city

*)*

return total\_distance

*#*

# Example Usage #

# List of city coordinates (x, y)

city\_coordinates = [(0, 0), (1, 2), (3, 1), (5, 4), (2, 6)]

# Generate the tour using Nearest Neighbor Algorithm tour = nearest\_neighbor\_algorithm(city\_coordinates)

# Calculate the total distance of the tour

total\_distance = calculate\_total\_distance(tour, city\_coordinates)

# Display the result print("Optimal tour:", tour)

print("Total distance:", total\_distance)

*#*

# Sample Output:

# Optimal tour: [3, 2, 1, 0, 4]

# Total distance: 18.007793826264315 #