

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
#-----

```