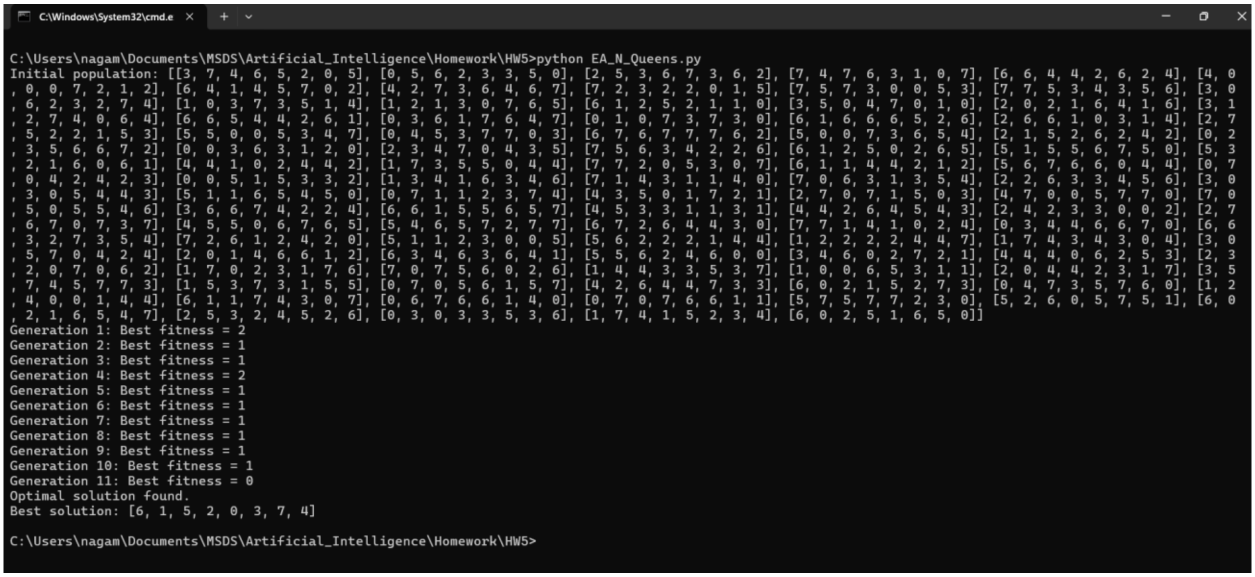
Homework 5

Problem 1: EA for N-Queens Problem - EA_N_Queens.py

This program first generates the initial population of random solutions. Each individual is a list of size N, where each entry represents a queen placed in a random row in a specific column. This process is repeated population_size times to create diverse individuals that will evolve.

a) Fitness Function: This function calculates the number of attacking queen pairs for a given individual solution. It compares every pair of queens to check if they share the same column or diagonal and increments the attack count for each such pair. The fewer the attacks, the better the fitness.
b) Fitness of the Population: This function applies the fitness function to each individual in the population. It returns a list of fitness scores, allowing the algorithm to evaluate which individuals have fewer attacking queens and are better candidates for reproduction.
c) Crossover Function: This function takes two parent solutions and produces two children by swapping segments between the parents. Two random crossover points are chosen, and the middle portion between those points is exchanged, combining traits from both parents in the child.
d) Mutation Function: This function introduces variation into a solution by randomly swapping the positions of two queens with a certain probability.
e) Selection Function: This function performs tournament selection, where small groups of individuals compete, and the one with the best fitness is chosen.
f) Evolution Loop: In each generation, new children are created by crossover and mutation. The current population is combined with the children, and the best solutions are selected for the next generation. This process continues until an optimal solution is found or the maximum number of generations is reached.

At the end of the evolution process, the best individual in the population is identified based on their fitness. If an optimal solution (with zero attacking queens) is found, the algorithm outputs this solution as the final result.



```
C:\Users\nagam\Documents\MSDS\Artificial_Intelligence\Homework\HW5>python EA_N_Queens.py
Initial population: [[3, 7, 4, 6, 5, 2, 0, 5], [0, 5, 6, 2, 3, 3, 5, 0], [2, 5, 3, 6, 7, 3, 6, 2], [7, 4, 7, 6, 3, 1, 0, 7], [6, 6, 4, 4, 2, 6, 2, 4], [4, 0
, 0, 0, 7, 2, 1, 2], [6, 4, 1, 4, 5, 7, 0, 2], [4, 2, 7, 3, 6, 4, 6, 7], [7, 2, 3, 2, 2, 0, 1, 5], [7, 5, 7, 3, 0, 0, 5, 3], [7, 7, 5, 3, 4, 3, 5, 6], [3, 0
, 6, 2, 3, 2, 7, 4], [1, 0, 3, 7, 3, 5, 1, 4], [1, 2, 1, 3, 0, 7, 6, 5], [6, 1, 2, 5, 2, 1, 1, 0], [3, 5, 0, 4, 7, 0, 1, 0], [2, 0, 2, 1, 6, 4, 1, 6], [3, 1
, 2, 7, 4, 0, 6, 4], [6, 6, 5, 4, 4, 2, 6, 1], [0, 3, 6, 1, 7, 6, 4, 7], [0, 1, 0, 7, 3, 7, 3, 0], [6, 1, 6, 6, 6, 5, 2, 6], [2, 6, 6, 1, 0, 3, 1, 4], [2, 7
, 5, 2, 2, 1, 5, 3], [5, 5, 0, 0, 5, 3, 4, 7], [0, 4, 5, 3, 7, 7, 0, 3], [6, 7, 6, 7, 7, 7, 6, 2], [5, 0, 0, 7, 3, 6, 5, 4], [2, 1, 5, 2, 6, 2, 4, 2], [0, 2
, 3, 5, 6, 6, 7, 2], [0, 0, 3, 6, 3, 1, 2, 0], [2, 3, 4, 7, 0, 4, 3, 5], [7, 5, 6, 3, 4, 2, 2, 6], [6, 1, 2, 5, 0, 2, 6, 5], [5, 1, 5, 5, 6, 7, 5, 0], [5, 3
, 2, 1, 6, 0, 6, 1], [4, 4, 1, 0, 2, 4, 4, 2], [1, 7, 3, 5, 5, 0, 4, 4], [7, 7, 2, 0, 5, 3, 0, 7], [6, 1, 1, 4, 4, 2, 1, 2], [5, 6, 7, 6, 6, 0, 4, 4], [0, 7
, 0, 4, 2, 4, 2, 3], [0, 0, 5, 1, 5, 3, 3, 2], [1, 3, 4, 1, 6, 3, 4, 6], [7, 1, 4, 3, 1, 1, 4, 0], [7, 0, 6, 3, 1, 3, 5, 4], [2, 2, 6, 3, 3, 4, 5, 6], [3, 0
, 3, 0, 5, 4, 4, 3], [5, 1, 1, 6, 5, 4, 5, 0], [0, 7, 1, 1, 2, 3, 7, 4], [4, 3, 5, 0, 1, 7, 2, 1], [2, 7, 0, 7, 1, 5, 0, 3], [4, 7, 0, 0, 5, 7, 7, 0], [7, 0
, 5, 0, 5, 5, 4, 6], [3, 6, 6, 7, 4, 2, 2, 4], [6, 6, 1, 5, 5, 6, 5, 7], [4, 5, 3, 3, 1, 1, 3, 1], [4, 4, 2, 6, 4, 5, 4, 3], [2, 4, 2, 3, 3, 0, 0, 2], [2, 7
, 6, 7, 0, 7, 3, 7], [4, 5, 5, 0, 6, 7, 6, 5], [5, 4, 6, 5, 7, 2, 7, 7], [6, 7, 2, 6, 4, 4, 3, 0], [7, 7, 1, 4, 1, 0, 2, 4], [0, 3, 4, 4, 6, 6, 7, 0], [6, 6
, 3, 2, 7, 3, 5, 4], [7, 2, 6, 1, 2, 4, 2, 0], [5, 1, 1, 2, 3, 0, 0, 5], [5, 6, 2, 2, 2, 1, 4, 4], [1, 2, 2, 2, 2, 4, 4, 7], [1, 7, 4, 3, 4, 3, 0, 4], [3, 0
, 5, 7, 0, 4, 2, 4], [2, 0, 1, 4, 6, 6, 1, 2], [6, 3, 4, 6, 3, 6, 4, 1], [5, 5, 6, 2, 4, 6, 0, 0], [3, 4, 6, 0, 2, 7, 2, 1], [4, 4, 4, 0, 6, 2, 5, 3], [2, 3
, 2, 0, 7, 0, 6, 2], [1, 7, 0, 2, 3, 1, 7, 6], [7, 0, 7, 5, 6, 0, 2, 6], [1, 4, 4, 3, 3, 5, 3, 7], [1, 0, 0, 6, 5, 3, 1, 1], [2, 0, 4, 4, 2, 3, 1, 7], [3, 5
, 7, 4, 5, 7, 7, 3], [1, 5, 3, 7, 3, 1, 5, 5], [0, 7, 0, 5, 6, 1, 5, 7], [4, 2, 6, 4, 4, 7, 3, 3], [6, 0, 2, 1, 5, 2, 7, 3], [0, 4, 7, 3, 5, 7, 6, 0], [1, 2
, 4, 0, 0, 1, 4, 4], [6, 1, 1, 7, 4, 3, 0, 7], [0, 6, 7, 6, 6, 1, 4, 0], [0, 7, 0, 7, 6, 6, 1, 1], [5, 7, 5, 7, 7, 2, 3, 0], [5, 2, 6, 0, 5, 7, 5, 1], [6, 0
, 2, 1, 6, 5, 4, 7], [2, 5, 3, 2, 4, 5, 2, 6], [0, 3, 0, 3, 3, 5, 3, 6], [1, 7, 4, 1, 5, 2, 3, 4], [6, 0, 2, 5, 1, 6, 5, 0]]
Generation 1: Best fitness = 2
Generation 2: Best fitness = 1
Generation 3: Best fitness = 1
Generation 4: Best fitness = 2
Generation 5: Best fitness = 1
Generation 6: Best fitness = 1
Generation 7: Best fitness = 1
Generation 8: Best fitness = 1
Generation 9: Best fitness = 1
Generation 10: Best fitness = 1
Generation 11: Best fitness = 0
Optimal solution found.
Best solution: [6, 1, 5, 2, 0, 3, 7, 4]

C:\Users\nagam\Documents\MSDS\Artificial_Intelligence\Homework\HW5>
```

Problem 2: EA for TSP – EA_TSP.py & EA_TSP_for_large_data.py

a) Fitness Function: This function calculates the total distance of a given tour. It iterates through each city in the tour, calculating the Euclidean distance between consecutive cities, including the return to the starting city. The sum of these distances is returned as the tour's fitness value, where lower values represent shorter distances and better solutions.
b) Crossover Function: This function performs ordered crossover between two parent tours. Two random points are chosen to define a segment, and this segment is copied from each parent to the respective child. The remaining positions in each child are filled with cities from the other parent while maintaining the original order, ensuring no city is repeated.
c) Mutation Function: The mutation function introduces random changes in the tour with a given probability. If mutation occurs, it swaps the positions of two randomly selected cities in the tour.
d) Validate Tour Function: This function ensures that the tour generated is valid by checking if all positions in the tour contain valid city indices (i.e., no None values).
e) Selection Function: In function a small group of individuals from the population is randomly selected, and the one with the best fitness (shortest distance) is chosen to continue to the next generation.
f) Plot Tour Function: This function visualizes a given tour by plotting it on a graph, where the cities are represented as points, and the tour as a path connecting them. It saves the resulting plot as an image.
g) Evolution Process: The evolutionary loop generates new children using crossover and mutation. It combines the current population with the new children and then

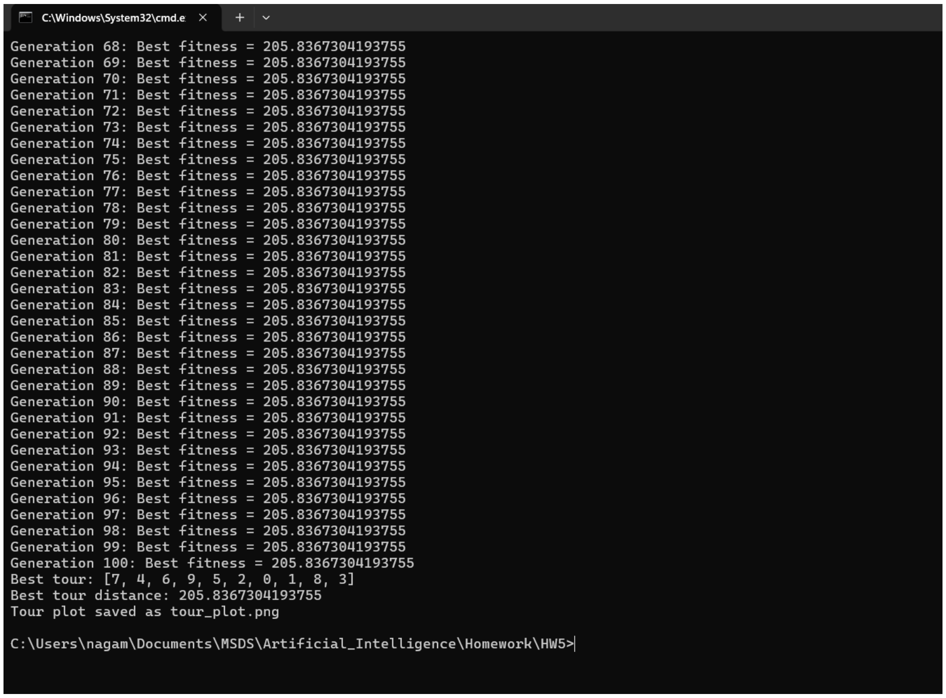
applies selection to retain the best individuals for the next generation. This process is repeated for a fixed number of generations or until an optimal solution is found.

EA_TSP.py (N=10)

In the program, cities are generated randomly using the NumPy function np.random.randint(0, 100, size=(N, 2)). This creates an array of N cities, where each city has two coordinates (x, y), representing its location on a 2D plane. The x and y values are integers between 0 and 100, simulating the random placement of cities within a 100x100 grid. These coordinates are then used to calculate distances between cities during the evolution process.

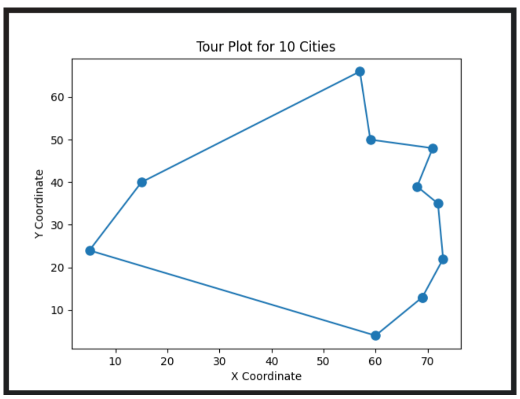
```
C:\Windows\System32\cmd.e

C:\Users\nagam\Documents\MSDS\Artificial_Intelligence\Homework\HW5>python EA_TSP.py
Generated cities: [[57 66]
 [15 40]
 [59 50]
 [60  4]
 [73 22]
 [71 48]
 [72 35]
 [69 13]
 [ 5 24]
 [68 39]]
Generation 1: Best fitness = 279.88273158538306
Generation 2: Best fitness = 247.12238908435833
Generation 3: Best fitness = 247.1223890843583
Generation 4: Best fitness = 245.39036967303986
Generation 5: Best fitness = 245.39036967303983
Generation 6: Best fitness = 245.39036967303983
Generation 7: Best fitness = 245.39036967303983
Generation 8: Best fitness = 235.8984475212012
Generation 9: Best fitness = 235.8984475212012
Generation 10: Best fitness = 235.8984475212012
Generation 11: Best fitness = 234.28036983442905
Generation 12: Best fitness = 225.58382949723523
Generation 13: Best fitness = 220.43362657672608
Generation 14: Best fitness = 215.26542632324387
Generation 15: Best fitness = 214.0699425855397
Generation 16: Best fitness = 214.06994258553974
Generation 17: Best fitness = 214.0699425855397
Generation 18: Best fitness = 214.0699425855397
Generation 19: Best fitness = 214.0699425855397
Generation 20: Best fitness = 214.0699425855397
Generation 21: Best fitness = 214.0699425855397
Generation 22: Best fitness = 214.0699425855397
Generation 23: Best fitness = 214.0699425855397
Generation 24: Best fitness = 214.0699425855397
Generation 25: Best fitness = 214.0699425855397
Generation 26: Best fitness = 214.0699425855397
Generation 27: Best fitness = 214.0699425855397
Generation 28: Best fitness = 205.8367304193755
Generation 29: Best fitness = 205.8367304193755
```

```
C:\Windows\System32\cmd.e

Generation 29: Best fitness = 205.8367304193755
Generation 30: Best fitness = 205.8367304193755
Generation 31: Best fitness = 205.8367304193755
Generation 32: Best fitness = 205.8367304193755
Generation 33: Best fitness = 205.8367304193755
Generation 34: Best fitness = 205.8367304193755
Generation 35: Best fitness = 205.8367304193755
Generation 36: Best fitness = 205.8367304193755
Generation 37: Best fitness = 205.8367304193755
Generation 38: Best fitness = 205.8367304193755
Generation 39: Best fitness = 205.8367304193755
Generation 40: Best fitness = 205.8367304193755
Generation 41: Best fitness = 205.8367304193755
Generation 42: Best fitness = 205.8367304193755
Generation 43: Best fitness = 205.8367304193755
Generation 44: Best fitness = 205.8367304193755
Generation 45: Best fitness = 205.8367304193755
Generation 46: Best fitness = 205.8367304193755
Generation 47: Best fitness = 205.8367304193755
Generation 48: Best fitness = 205.8367304193755
Generation 49: Best fitness = 205.8367304193755
Generation 50: Best fitness = 205.8367304193755
Generation 51: Best fitness = 205.8367304193755
Generation 52: Best fitness = 205.8367304193755
Generation 53: Best fitness = 205.8367304193755
Generation 54: Best fitness = 205.8367304193755
Generation 55: Best fitness = 205.8367304193755
Generation 56: Best fitness = 205.8367304193755
Generation 57: Best fitness = 205.8367304193755
Generation 58: Best fitness = 205.8367304193755
Generation 59: Best fitness = 205.8367304193755
Generation 60: Best fitness = 205.8367304193755
Generation 61: Best fitness = 205.8367304193755
Generation 62: Best fitness = 205.8367304193755
Generation 63: Best fitness = 205.8367304193755
Generation 64: Best fitness = 205.8367304193755
Generation 65: Best fitness = 205.8367304193755
Generation 66: Best fitness = 205.8367304193755
Generation 67: Best fitness = 205.8367304193755
Generation 68: Best fitness = 205.8367304193755
Generation 69: Best fitness = 205.8367304193755
```



C:\Windows\System32\cmd.e

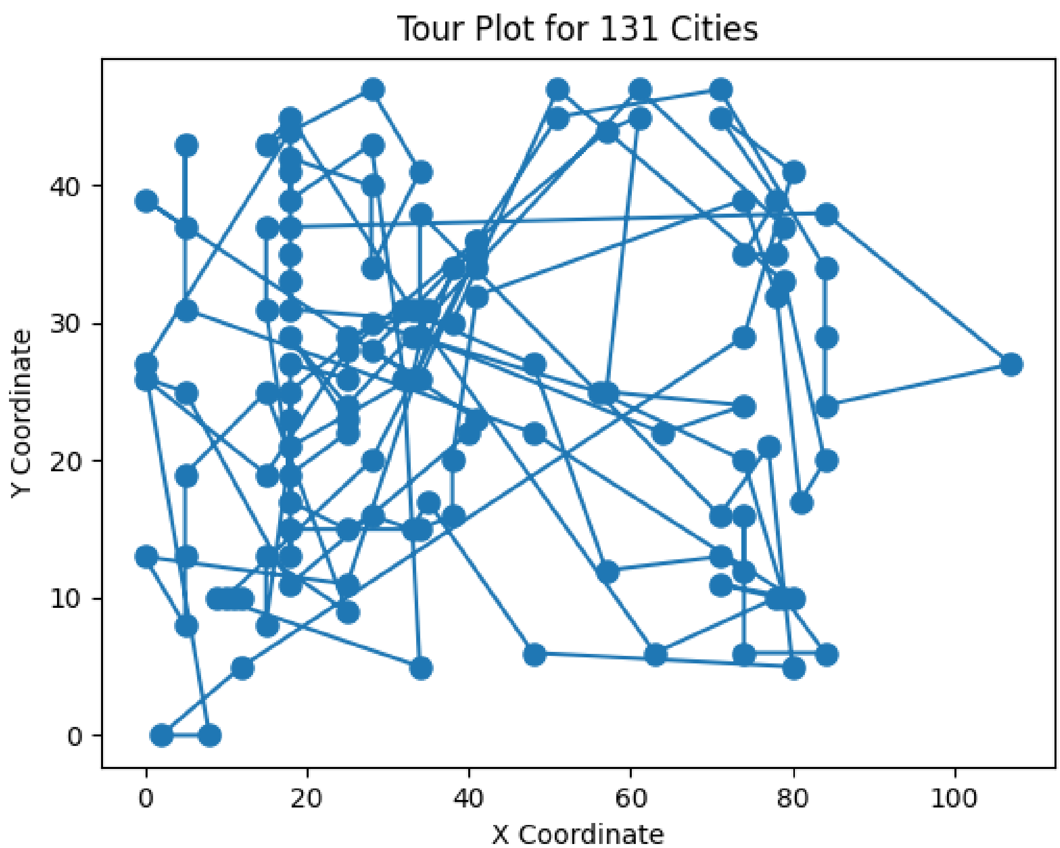
```
Generation 68: Best fitness = 205.8367304193755
Generation 69: Best fitness = 205.8367304193755
Generation 70: Best fitness = 205.8367304193755
Generation 71: Best fitness = 205.8367304193755
Generation 72: Best fitness = 205.8367304193755
Generation 73: Best fitness = 205.8367304193755
Generation 74: Best fitness = 205.8367304193755
Generation 75: Best fitness = 205.8367304193755
Generation 76: Best fitness = 205.8367304193755
Generation 77: Best fitness = 205.8367304193755
Generation 78: Best fitness = 205.8367304193755
Generation 79: Best fitness = 205.8367304193755
Generation 80: Best fitness = 205.8367304193755
Generation 81: Best fitness = 205.8367304193755
Generation 82: Best fitness = 205.8367304193755
Generation 83: Best fitness = 205.8367304193755
Generation 84: Best fitness = 205.8367304193755
Generation 85: Best fitness = 205.8367304193755
Generation 86: Best fitness = 205.8367304193755
Generation 87: Best fitness = 205.8367304193755
Generation 88: Best fitness = 205.8367304193755
Generation 89: Best fitness = 205.8367304193755
Generation 90: Best fitness = 205.8367304193755
Generation 91: Best fitness = 205.8367304193755
Generation 92: Best fitness = 205.8367304193755
Generation 93: Best fitness = 205.8367304193755
Generation 94: Best fitness = 205.8367304193755
Generation 95: Best fitness = 205.8367304193755
Generation 96: Best fitness = 205.8367304193755
Generation 97: Best fitness = 205.8367304193755
Generation 98: Best fitness = 205.8367304193755
Generation 99: Best fitness = 205.8367304193755
Generation 100: Best fitness = 205.8367304193755
Best tour: [7, 4, 6, 9, 5, 2, 0, 1, 8, 3]
Best tour distance: 205.8367304193755
Tour plot saved as tour_plot.png

C:\Users\nagam\Documents\MSDS\Artificial_Intelligence\Homework\HW5>
```



EA_TSP_for_large_data.py

The input to this program comes from a .tsp file, which contains a list of city coordinates for solving the Traveling Salesman Problem (TSP). The read_tsp_file function parses the file by looking for the section marked NODE_COORD_SECTION, which signifies the start of the city data. Each line after this section contains the index of the city followed by its x and y coordinates. The function reads these coordinates until it reaches the EOF marker, and stores them as a list of city coordinates. These coordinates are then converted into a NumPy array, where each city has a pair of floating-point numbers representing its x and y positions on a 2D plane. The total number of cities is determined by the length of this array, which in the case of the provided example (xqf131.tsp), would typically contain 131 cities. This data serves as the input for the genetic algorithm to calculate the optimal tour.

OUTPUT: Output of the ***EA_TSP_for_large_data.py*** program is copied in EA_TSP_for_large_data_OUTPUT.txt. I have ran the program for 500 iterations as its taking time for finding the solution.



Problem 3: Ant Colony Optimization for TSP – Ant_Colony_Optimization_TSP.py

a) calculate_cost(path, distance_matrix): This function calculates the total distance (cost) of a given path based on the provided distance matrix. It iterates over each city in the path, summing up the distances between consecutive cities and the final city back to the first one, ensuring the tour is complete. The result is the total cost of traveling through the cities in the specified order.
b) select_next_city(current_city, visited): This function selects the next city to visit using a probabilistic approach based on pheromone levels and visibility (inverse of distance). For each unvisited city, the probability is computed as a function of pheromone strength and visibility raised to their respective powers. After normalizing the probabilities, the next city is chosen using stochastic sampling. If no valid probabilities are left, a random unvisited city is selected.

c) two_opt_mutation(path, distance_matrix): This function applies the 2-opt mutation, a local search algorithm, to improve the current path. It generates new paths by reversing sections of the current path and compares their costs. If a new path has a lower cost, it replaces the current path. The function returns the best path found after evaluating all possible 2-opt swaps.

d) find_path(start): Starting from a given city, this function builds a complete path for an ant. It iteratively selects the next city to visit using the select_next_city function until all cities are visited. The cities are stored in the path list, and a set visited keeps track of cities already included in the path.

The main loop runs the Ant Colony Optimization (ACO) for a specified number of iterations. For each ant, it finds a path, applies 2-opt mutation, and calculates the cost. If the ant's path is the best found so far, it's saved as the global best. Each ant contributes to updating the pheromone matrix based on the quality (cost) of its path. After every iteration, pheromone evaporation occurs, and the pheromone matrix is updated. The process continues until the algorithm converges or reaches the maximum number of iterations.

```
C:\Windows\System32\cmd.e

C:\Users\nagam\Documents\MSDS\Artificial_Intelligence\Homework\HW5\Submission>python Ant_Colony_Optimization_TSP.py
Generation 1: Best fitness = 369.5
Generation 2: Best fitness = 369.5
Generation 3: Best fitness = 369.5
Generation 4: Best fitness = 347.0
Generation 5: Best fitness = 347.0
Generation 6: Best fitness = 347.0
Generation 7: Best fitness = 347.0
Generation 8: Best fitness = 347.0
Generation 9: Best fitness = 347.0
Generation 10: Best fitness = 347.0
Generation 11: Best fitness = 347.0
Generation 12: Best fitness = 347.0
Generation 13: Best fitness = 347.0
Generation 14: Best fitness = 347.0
Generation 15: Best fitness = 347.0
Generation 16: Best fitness = 347.0
Generation 17: Best fitness = 347.0
Generation 18: Best fitness = 347.0
Generation 19: Best fitness = 347.0
Generation 20: Best fitness = 347.0
Generation 21: Best fitness = 347.0
Generation 22: Best fitness = 347.0
Generation 23: Best fitness = 347.0
Generation 24: Best fitness = 347.0
Generation 25: Best fitness = 347.0
Generation 26: Best fitness = 347.0
Generation 27: Best fitness = 347.0
Generation 28: Best fitness = 347.0
Generation 29: Best fitness = 347.0
Generation 30: Best fitness = 347.0
Generation 31: Best fitness = 347.0
Generation 32: Best fitness = 347.0
Generation 33: Best fitness = 347.0
Generation 34: Best fitness = 347.0
Generation 35: Best fitness = 347.0
Generation 36: Best fitness = 347.0
Generation 37: Best fitness = 347.0
Generation 38: Best fitness = 347.0
Generation 39: Best fitness = 347.0
```

C:\Windows\System32\cmd.e

```
Generation 39: Best fitness = 347.0
Generation 40: Best fitness = 347.0
Generation 41: Best fitness = 347.0
Generation 42: Best fitness = 347.0
Generation 43: Best fitness = 347.0
Generation 44: Best fitness = 347.0
Generation 45: Best fitness = 347.0
Generation 46: Best fitness = 347.0
Generation 47: Best fitness = 347.0
Generation 48: Best fitness = 347.0
Generation 49: Best fitness = 347.0
Generation 50: Best fitness = 347.0
Generation 51: Best fitness = 347.0
Generation 52: Best fitness = 347.0
Generation 53: Best fitness = 347.0
Generation 54: Best fitness = 347.0
Generation 55: Best fitness = 347.0
Generation 56: Best fitness = 347.0
Generation 57: Best fitness = 347.0
Generation 58: Best fitness = 347.0
Generation 59: Best fitness = 347.0
Generation 60: Best fitness = 347.0
Generation 61: Best fitness = 347.0
Generation 62: Best fitness = 347.0
Generation 63: Best fitness = 347.0
Generation 64: Best fitness = 347.0
Generation 65: Best fitness = 347.0
Generation 66: Best fitness = 347.0
Generation 67: Best fitness = 347.0
Generation 68: Best fitness = 347.0
Generation 69: Best fitness = 347.0
Generation 70: Best fitness = 347.0
Generation 71: Best fitness = 347.0
Generation 72: Best fitness = 347.0
Generation 73: Best fitness = 347.0
Generation 74: Best fitness = 347.0
Generation 75: Best fitness = 347.0
Generation 76: Best fitness = 347.0
Generation 77: Best fitness = 347.0
Generation 78: Best fitness = 347.0
Generation 79: Best fitness = 347.0
```

C:\Windows\System32\cmd.e

```
Generation 64: Best fitness = 347.0
Generation 65: Best fitness = 347.0
Generation 66: Best fitness = 347.0
Generation 67: Best fitness = 347.0
Generation 68: Best fitness = 347.0
Generation 69: Best fitness = 347.0
Generation 70: Best fitness = 347.0
Generation 71: Best fitness = 347.0
Generation 72: Best fitness = 347.0
Generation 73: Best fitness = 347.0
Generation 74: Best fitness = 347.0
Generation 75: Best fitness = 347.0
Generation 76: Best fitness = 347.0
Generation 77: Best fitness = 347.0
Generation 78: Best fitness = 347.0
Generation 79: Best fitness = 347.0
Generation 80: Best fitness = 347.0
Generation 81: Best fitness = 347.0
Generation 82: Best fitness = 347.0
Generation 83: Best fitness = 347.0
Generation 84: Best fitness = 347.0
Generation 85: Best fitness = 347.0
Generation 86: Best fitness = 347.0
Generation 87: Best fitness = 347.0
Generation 88: Best fitness = 347.0
Generation 89: Best fitness = 347.0
Generation 90: Best fitness = 347.0
Generation 91: Best fitness = 347.0
Generation 92: Best fitness = 347.0
Generation 93: Best fitness = 347.0
Generation 94: Best fitness = 347.0
Generation 95: Best fitness = 347.0
Generation 96: Best fitness = 347.0
Generation 97: Best fitness = 347.0
Generation 98: Best fitness = 347.0
Generation 99: Best fitness = 347.0
Generation 100: Best fitness = 347.0
Best path found: [3, 7, 0, 4, 5, 2, 6, 1, 8, 9]
Best path cost: 347.0

C:\Users\nagam\Documents\MSDS\Artificial_Intelligence\Homework\HW5\Submission>
```