Assignment 9: Genetic algorithm on a Search problem

Objective: Solving the 8 Queens Problem using genetic algorithm (GA)

Problem Statement: The objective of this assignment is to design and implement a Genetic Algorithm to solve the 8 Queens Problem, a classic constraint-based problem where 8 queens must be placed on a chessboard such that no two queens threaten each other.

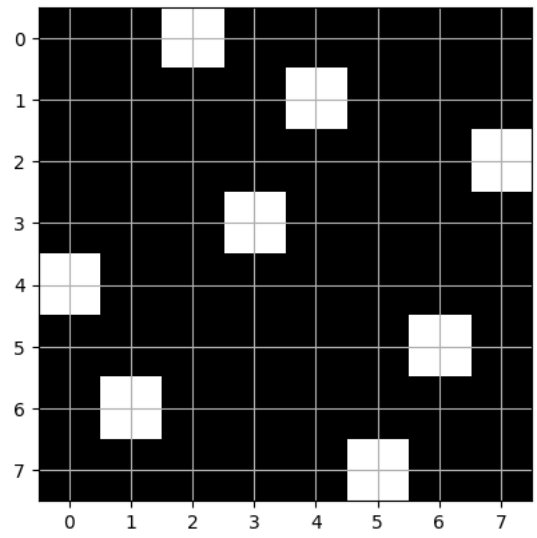
Tasks:

1. Implement GA for 8-queens problem. Visualize the chessboard and the placement of queens for the best solution.
2. Extend the algorithm to solve the N-Queens Problem for larger values of N (e. g., N = 10, 20).
3. Compare GA with A* for the 8 Queens problem in terms of number of states explored (Heuristic function can be the number of conflicting queens in the current state).

```
import random
import numpy as np
import matplotlib.pyplot as plt

# Constants
N = 8  # Change for larger N-Queens
POPULATION_SIZE = 100
MUTATION_RATE = 0.2
CROSSOVER_RATE = 0.8
MAX_GENERATIONS = 1000

def fitness(chromosome):
    """Calculate the fitness of a chromosome. Higher fitness means fewer conflicts."""
    non_attacking_pairs = (N * (N - 1)) // 2
    conflicts = 0
    for i in range(N):
        for j in range(i + 1, N):
            if abs(chromosome[i] - chromosome[j]) == abs(i - j):  # Diagonal attack
                conflicts += 1
    return non_attacking_pairs - conflicts

def initialize_population():
    """Generate initial population with random permutations of columns."""
    return [random.sample(range(1, N + 1), N) for _ in range(POPULATION_SIZE)]

def selection(population):
    """Select parents using tournament selection."""
    tournament = random.sample(population, 5)
    return max(tournament, key=fitness)

def crossover(parent1, parent2):
    """Perform Order Crossover (OX)."""
    if random.random() > CROSSOVER_RATE:
        return parent1[:], parent2[:]

    point1, point2 = sorted(random.sample(range(N), 2))
    child1, child2 = [-1] * N, [-1] * N
    child1[point1:point2] = parent1[point1:point2]
    child2[point1:point2] = parent2[point1:point2]

    def fill_child(child, parent):
        fill_pos = 0
        for gene in parent:
            if gene not in child:
                while child[fill_pos] != -1:
                    fill_pos += 1
                child[fill_pos] = gene
        return child

    return fill_child(child1, parent2), fill_child(child2, parent1)

def mutate(chromosome):
    """Swap two random positions with a mutation probability."""
    if random.random() < MUTATION_RATE:
        i, j = random.sample(range(N), 2)
        chromosome[i], chromosome[j] = chromosome[j], chromosome[i]
    return chromosome

def genetic_algorithm():
```

```
    """Main GA loop."""
    population = initialize_population()
    for generation in range(MAX_GENERATIONS):
        population = sorted(population, key=fitness, reverse=True)
        if fitness(population[0]) == (N * (N - 1)) // 2:
            print(f"Solution found in {generation} generations!")
            return population[0]

        new_population = population[:10]  # Elitism
        while len(new_population) < POPULATION_SIZE:
            parent1, parent2 = selection(population), selection(population)
            child1, child2 = crossover(parent1, parent2)
            new_population += [mutate(child1), mutate(child2)]
        population = new_population[:POPULATION_SIZE]
    return population[0]

def visualize_board(solution):
    """Display the chessboard with queens."""
    board = np.zeros((N, N))
    for row, col in enumerate(solution):
        board[row, col - 1] = 1

    plt.imshow(board, cmap='gray')
    plt.xticks(range(N))
    plt.yticks(range(N))
    plt.grid()
    plt.show()

if __name__ == "__main__":
    best_solution = genetic_algorithm()
    print("Best solution:", best_solution)
    visualize_board(best_solution)


    from datetime import datetime
# Display the current date and time in a formatted style
current_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
print("Current Date and Time:", current_time)
print("\nName: Priyanka Sanyal")
print("Section: CSE 38")
print("Semester: 6th")
print("Roll Number:22051532")
print("Date: 18-03-2025")
print("Assignment 9: Genetic algorithm on a Search problem")
```

```
Solution found in 4 generations!
Best solution: [3, 5, 8, 4, 1, 7, 2, 6]
```




```
Current Date and Time: 2025-03-23 13:31:04

Name: Priyanka Sanyal
Section: CSE 38
Semester: 6th
Roll Number:22051532
Date: 18-03-2025
Assignment 9: Genetic algorithm on a Search problem
```