



Data Structures And Algorithms

(Course Code: CSC211)

Name :Syeda Aima Abbas

Roll No # FA24-BSE-010

Instructor Name : Sir Imran

Assignment # 3

Lab Part

Lab 10

```
import numpy as np
import random

def create_8_queens_board():
    queens_board = np.full((8,8), '.')
    print(queens_board)

        # Place queens (represented by 'Q') on the board
    for row in range(0,8):
        col = random.randint(0, 7)
    print(row, col)           queens_board[row,
    col] = 'Q'
        return
queens_board

def board_chromosome(board):
    queens_positions = np.argwhere(board == 'Q')
    print('Q Positions', queens_positions)

        # Returns columns      cols =
queens_positions[:, 1]
    print('chromosome', cols)
        return
cols
    def
calculate_fitness_numpy(board):
    queens_positions = np.argwhere(board == 'Q')
num_queens = len(queens_positions)

    rows = queens_positions[:, 0]
cols = queens_positions[:, 1]

    print("cols", cols)
        unique_cols, counts = np.unique(cols,
return_counts=True)      unique_only = unique_cols[counts ==
1]      col_attacks = num_queens - len(unique_only)
    print("Column attacks:", col_attacks)

        # diagnol attack
```

```

        diagonal_attacks = 0      for i
in range(num_queens):          for j in
range(i + 1, num_queens):

            if abs(rows[i] - rows[j]) == abs(cols[i] -
cols[j]):
                diagonal_attacks += 1
        print("Diagonal attacks:",
diagonal_attacks)
        total_attacks = col_attacks +
diagonal_attacks

    fitness = 8 - total_attacks
return fitness


def top_two_chromosomes(boards, chromosomes, fitness_values):
combined = list(zip(fitness_values, boards, chromosomes))
sorted_combined = sorted(combined, key=lambda x:
x[0])
top_two = sorted_combined[-
2:]      return top_two


def single_point_crossover(parent1, parent2):
crossover_point = 4      print('crossoverpoint',
crossover_point)
offspring1 =
np.concatenate((parent1[:crossover_point],
parent2[crossover_point:]))      offspring2 =
np.concatenate((parent2[:crossover_point],
parent1[crossover_point:]))


return offspring1, offspring2


def mutate(chromosome):
    index =
random.randint(0, 7)
    new_value =
random.randint(0, 7)

```

```

    print(f"Mutating index {index} →
{new_value}")
    chromosome[index] =
new_value      return chromosome


population_size = int(input("Enter number of boards (population size): "))
boards = []
chromosomes = []
fitness_values =
[]

# Generate population for i in
range(population_size):      print(f"\n-
-- Board {i+1} ---")      board =
create_8_queens_board()      chrom =
board_chromosome(board)      fit =
calculate_fitness_numpy(board)
    boards.append(board)
chromosomes.append(chrom)
fitness_values.append(fit)

# Select top 2 top_two = top_two_chromosomes(boards, chromosomes,
fitness_values) parent1 = top_two[-1][2] parent2 = top_two[-2][2]
print("\nSelected Parent 1:",
parent1) print("Selected Parent 2:",
parent2)

# Crossover offspring1, offspring2 =
single_point_crossover(parent1, parent2)
print("\nOffspring 1 before mutation:",
offspring1) print("Offspring 2 before mutation:",
offspring2)

# Mutation offspring1 =
mutate(offspring1) offspring2
= mutate(offspring2)
print("\nOffspring 1 after mutation:",
offspring1) print("Offspring 2 after mutation:",
offspring2)

```

output:

- PS C:\Users\khans\OneDrive\Desktop\ai> **python** ga.py
Enter number of boards (population size): 4

--- Board 1 ---
[['.', '.', '.', '.', '.', '.', '.', '.', '.', '.']]
[['.', '.', '.', '.', '.', '.', '.', '.', '.', '.']]
[['.', '.', '.', '.', '.', '.', '.', '.', '.', '.']]
[['.', '.', '.', '.', '.', '.', '.', '.', '.', '.']]
[['.', '.', '.', '.', '.', '.', '.', '.', '.', '.']]
[['.', '.', '.', '.', '.', '.', '.', '.', '.', '.']]
[['.', '.', '.', '.', '.', '.', '.', '.', '.', '.']]
[['.', '.', '.', '.', '.', '.', '.', '.', '.', '.']]
0 4
1 5
2 5
3 2
4 4
5 1
6 5
7 7
Q Positions [[0 4]
[1 5]
[2 5]
[3 2]
[4 4]
[5 1]
[6 5]
[7 7]]
chromosome [4 5 5 2 4 1 5 7]
cols [4 5 5 2 4 1 5 7]
Column attacks: 5
Diagonal attacks: 4

--- Board 2 ---

```
[['. . . . . . .'],
 ['. . . . . . .'],
 ['. . . . . . .'],
 ['. . . . . . .'],
 ['. . . . . . .'],
 ['. . . . . . .'],
 ['. . . . . . .'],
 ['. . . . . . .']]
```

0 1

1 6

2 1

3 6

4 4

5 2

6 1

7 2

Q Positions [[0 1]

[1 6]

[2 1]

[3 6]

[4 4]

[5 2]

[6 1]

[7 2]]

chromosome [1 6 1 6 4 2 1 2]

cols [1 6 1 6 4 2 1 2]

Column attacks: 7

Diagonal attacks: 5

--- Board 3 ---

0 1

16

26

3 4

4 3

5 3

6 4

75

Q P

[1 6]

[2 6]

[3 4]

[4 3]

[5 3]

[5 5]
[6 4]

[6 4]
[7 5]

[/ 5]

chromosome [1 6 6 4 3 3 4 5]
label [1 5 5 5 5 3 4 5]

cols [1 6 6 4 3 3 4 5]

Column attacks: 6

Diagonal attacks: 8

```
--- Board 4 ---
[[ '.', '.', '.', '.', '.', '.', '.', '.', '.', '.'],
 [ '.', '.', '.', '.', '.', '.', '.', '.', '.', '.'],
 [ '.', '.', '.', '.', '.', '.', '.', '.', '.', '.'],
 [ '.', '.', '.', '.', '.', '.', '.', '.', '.', '.'],
 [ '.', '.', '.', '.', '.', '.', '.', '.', '.', '.'],
 [ '.', '.', '.', '.', '.', '.', '.', '.', '.', '.'],
 [ '.', '.', '.', '.', '.', '.', '.', '.', '.', '.'],
 [ '.', '.', '.', '.', '.', '.', '.', '.', '.', '.]]]
0 1
1 2
2 5
3 7
4 6
5 7
6 5
7 7
Q Positions [[0 1]
 [1 2]
 [2 5]
 [3 7]
 [4 6]
 [5 7]
 [6 5]
 [7 7]]
chromosome [1 2 5 7 6 7 5 7]
cols [1 2 5 7 6 7 5 7]
Column attacks: 5
Diagonal attacks: 3

Selected Parent 1: [1 2 5 7 6 7 5 7]
Selected Parent 2: [4 5 5 2 4 1 5 7]
crossoverpoint 4

Offspring 1 before mutation: [1 2 5 7 4 1 5 7]
Offspring 2 before mutation: [4 5 5 2 6 7 5 7]
Mutating index 0 → 7
Mutating index 6 → 3

Offspring 1 after mutation: [7 2 5 7 4 1 5 7]
Offspring 2 after mutation: [4 5 5 2 6 7 3 7]
○ PS C:\Users\khans\OneDrive\Desktop\ai> █
```