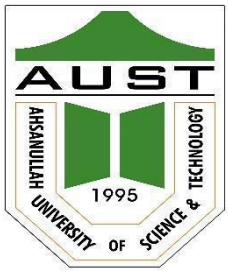

Ahsanullah University of Science & Technology

Department of Computer Science & Engineering

Course No : **CSE4108**
Course Title : **Artificial Intelligence Lab**
Assignment No : **Term Assignment 2**

Date of Performance : **11.02.2023**
Date of Submission : **12.02.2023**

Submitted To : **Mr. Faisal Muhammad Shah & Mr. Raihan Tanvir**

Submitted By-
Group : **C_2**
Name : **Ashiqul Islam**
Id : **190104140**
Section : **C**

Topic: Random Restart Hill Climbing

Source Code:

```
import random
def generate_radom_state():
    new_state = list()
    for i in range(8):
        new_state.append(random.randint(1, 8))
    return new_state

def calculate_heuristic(state):
    h_value = 0
    for i in range(1, 9):
        for j in range(i+1, 9):
            if (state[i-1] == state[j-1]): # Checking face to face
                h_value += 1
            if (abs(i - j) == abs(state[i-1] - state[j-1])): #Checking diagonally
                h_value += 1
    return h_value

def calculate_fitness(state):
    # Max value of fitness can be 28 (max number of attacking pair for 8-Queen)
    return 28 - calculate_heuristic(state)

input_state = input("Current state of board for 8-Queen : ")
state = list(map(int, input_state))
goal_fitness = int(input("Goal fitness (max 28) : "))

fitness = calculate_fitness(state)
f = open("intermediate_states.txt", "w")

if(fitness == goal_fitness):
    print("Fitness of given state is same as Goal Fitness.")
else:
    print("Initial Fitness: ", fitness)
    flag = False
    while (1):
        for i in range(8):
            state[i] = random.randint(1, 8)
            new_fitness = calculate_fitness(state)
            print("\tCurrent State: ", state, "Fitness:", new_fitness)
             f.write("Current State"+str(state)+" Fitness: "+ str(new_fitness) + "\n")

            if (new_fitness > fitness):
                fitness = new_fitness
            elif (new_fitness == fitness):
                print("Local Maxima Reached,\n\tFitness value: ", new_fitness)
                print("Restarting Randomly!\n")
                f.write("Local Maxima Reached, Restarting Randomly!\n\n")

                state = generate_radom_state()
                fitness = calculate_fitness(state)
                print("\tCurrent State: ", state, "Fitness:", fitness)
```

```
            if(fitness >= goal_fitness):
                flag = True
                print("Goal fitness found with state, ", state)
                break;
        if flag == True : break;

if (fitness == 28):
    print("\nGlobal Maxima Reached with state ", state)
```

Output:

```
D:\4.1\AI\Lab\AI_Codes>python -u "d:\4.1\AI\Lab\AI_Codes\190104140_Term
Assignment2.py"
Current state of board for 8-Queen : 23456782
Goal fitness (max 28) : 22
Initial Fitness:  6
        Current State:  [6, 3, 4, 5, 6, 7, 8, 2] Fitness: 11
        Current State:  [6, 8, 4, 5, 6, 7, 8, 2] Fitness: 14
        Current State:  [6, 8, 4, 5, 6, 7, 8, 2] Fitness: 14
Local Maxima Reached,
        Fitness value:  14
Restarting Randomly!

        Current State:  [5, 4, 4, 5, 2, 3, 6, 5] Fitness: 16
        Current State:  [5, 4, 4, 6, 2, 3, 6, 5] Fitness: 18
        Current State:  [5, 4, 4, 6, 7, 3, 6, 5] Fitness: 19
        Current State:  [5, 4, 4, 6, 7, 2, 6, 5] Fitness: 20
        Current State:  [5, 4, 4, 6, 7, 2, 5, 5] Fitness: 19
        Current State:  [5, 4, 4, 6, 7, 2, 5, 6] Fitness: 19
        Current State:  [5, 4, 4, 6, 7, 2, 5, 6] Fitness: 19
        Current State:  [5, 7, 4, 6, 7, 2, 5, 6] Fitness: 22
Goal fitness found with state,  [5, 7, 4, 6, 7, 2, 5, 6]

D:\4.1\AI\Lab\AI_Codes>
```

```
D:\4.1\AI\Lab\AI_Codes>python -u "d:\4.1\AI\Lab\AI_Codes\190104140_Te
rmAssignment2.py"
Current state of board for 8-Queen : 23456782
Goal fitness (max 28) : 23
Initial Fitness:  6
        Current State:  [3, 3, 4, 5, 6, 7, 8, 2] Fitness: 12
        Current State:  [3, 2, 4, 5, 6, 7, 8, 2] Fitness: 16
        Current State:  [3, 2, 5, 5, 6, 7, 8, 2] Fitness: 18
        Current State:  [3, 2, 5, 6, 6, 7, 8, 2] Fitness: 18
Local Maxima Reached,
        Fitness value:  18
Restarting Randomly!

        Current State:  [7, 3, 1, 5, 1, 4, 2, 1] Fitness: 20
        Current State:  [7, 3, 1, 5, 3, 4, 2, 1] Fitness: 18
        Current State:  [7, 3, 1, 5, 3, 6, 2, 1] Fitness: 20
Local Maxima Reached,
        Fitness value:  20
Restarting Randomly!

        Current State:  [5, 5, 3, 6, 7, 6, 8, 6] Fitness: 20
        Current State:  [5, 5, 3, 6, 7, 6, 5, 6] Fitness: 15
        Current State:  [5, 5, 3, 6, 7, 6, 5, 8] Fitness: 16
        Current State:  [1, 5, 3, 6, 7, 6, 5, 8] Fitness: 16
        Current State:  [1, 4, 3, 6, 7, 6, 5, 8] Fitness: 14
        Current State:  [1, 4, 1, 6, 7, 6, 5, 8] Fitness: 16
        Current State:  [1, 4, 1, 8, 7, 6, 5, 8] Fitness: 15
        Current State:  [1, 4, 1, 8, 1, 6, 5, 8] Fitness: 16
        Current State:  [1, 4, 1, 8, 1, 6, 5, 8] Fitness: 16
        Current State:  [1, 4, 1, 8, 1, 6, 7, 8] Fitness: 16
        Current State:  [1, 4, 1, 8, 1, 6, 7, 4] Fitness: 16
        Current State:  [6, 4, 1, 8, 1, 6, 7, 4] Fitness: 19
        Current State:  [6, 2, 1, 8, 1, 6, 7, 4] Fitness: 18
        Current State:  [6, 2, 6, 8, 1, 6, 7, 4] Fitness: 18
        Current State:  [6, 2, 6, 5, 1, 6, 7, 4] Fitness: 19
        Current State:  [6, 2, 6, 5, 8, 6, 7, 4] Fitness: 19
        Current State:  [6, 2, 6, 5, 8, 1, 7, 4] Fitness: 23
Goal fitness found with state,  [6, 2, 6, 5, 8, 1, 7, 4]

D:\4.1\AI\Lab\AI_Codes>
```

Explanation:

generate_random_state() function generates a random state for the 8-Queen problem by creating a list of 8 random integers each between 1 and 8.

calculate_heuristic() function calculates the heuristic value by counting the number of attacking pairs for the given state. It loops through each queen and checks if any other queen is attacking it either face to face or diagonally.

calculate_fitness() function calculates the fitness of a given state. The fitness is calculated by subtracting the heuristic value from 28, which is the maximum number of attacking pairs for 8 queens.

The *state of the board* for *8-Queen* problem is taken as user input and then converted to a list at first. Then initial fitness of the state provided by the user is calculated using *calculate_fitness()*. If the fitness of the state is equal to the goal fitness, then the algorithm terminates as the goal state has been reached. Otherwise, the algorithm continues in a loop. In each iteration, the state is updated by randomly choosing a new position for a queen. If the new fitness is greater than the previous fitness then fitness is updated. If the new fitness is equal to the current fitness, the algorithm has reached a local maxima and the algorithm restarts with a new random state. The loop continues until the goal fitness is reached and then prints the goal state. The program also prints the state and its fitness for each iteration and if the global maxima is reached then it also prints the state for which the global maxima is found.