Ahsanullah University of Science & Technology

Department of Computer Science & Engineering

Course No	: CSE4108
Course Title	: Artificial Intelligence Lab
Assignment No	: 4
Date of Submission	: 18.07.23
Submitted To	: Dr. S.M.A. Al-Mamun & Mr. Raihan Tanvir

Submitted By

Group	: B2
Name	: MD Fardin Jaman Aranyak
Id	: 190204093
Section	: B2

Answer:

```
import random


def initial_state():
    """Generate a random initial state."""
    state = list(range(8))
    random.shuffle(state)
    return state


def calculate_attacks(state):
    """Calculate the number of queen attacks in the given state."""
    attacks = 0
    for i in range(8):
        for j in range(i + 1, 8):
            if state[i] == state[j] or abs(state[i] - state[j]) == j - i:
                attacks += 1
    return attacks


def get_neighbors(state):
    """Generate all neighboring states by swapping two queens."""
    neighbors = []
    for i in range(8):
        for j in range(i + 1, 8):
            neighbor = state[:]
            neighbor[i], neighbor[j] = neighbor[j], neighbor[i]
            neighbors.append(neighbor)
    return neighbors


def stochastic_hill_climbing():
    """Solve the 8-Queen Problem using Stochastic Hill Climbing."""
    current_state = initial_state()
    current_attacks = calculate_attacks(current_state)

    while current_attacks > 0:
        neighbors = get_neighbors(current_state)
        best_neighbor = None
        best_attacks = current_attacks

        for neighbor in neighbors:
            neighbor_attacks = calculate_attacks(neighbor)
            if neighbor_attacks < best_attacks:
                best_neighbor = neighbor
                best_attacks = neighbor_attacks

        if best_attacks >= current_attacks:
            # Randomly select a neighbor with the same number of attacks
            same_attacks_neighbors = [neighbor for neighbor in neighbors if
calculate_attacks(neighbor) == current_attacks]
            if same_attacks_neighbors:
                best_neighbor = random.choice(same_attacks_neighbors)

        if best_neighbor is None:
            break

        current_state = best_neighbor
        current_attacks = best_attacks

    return current_state

# Example usage
solution = stochastic_hill_climbing()
print("Solution:", solution)
```