AI
LAB 5

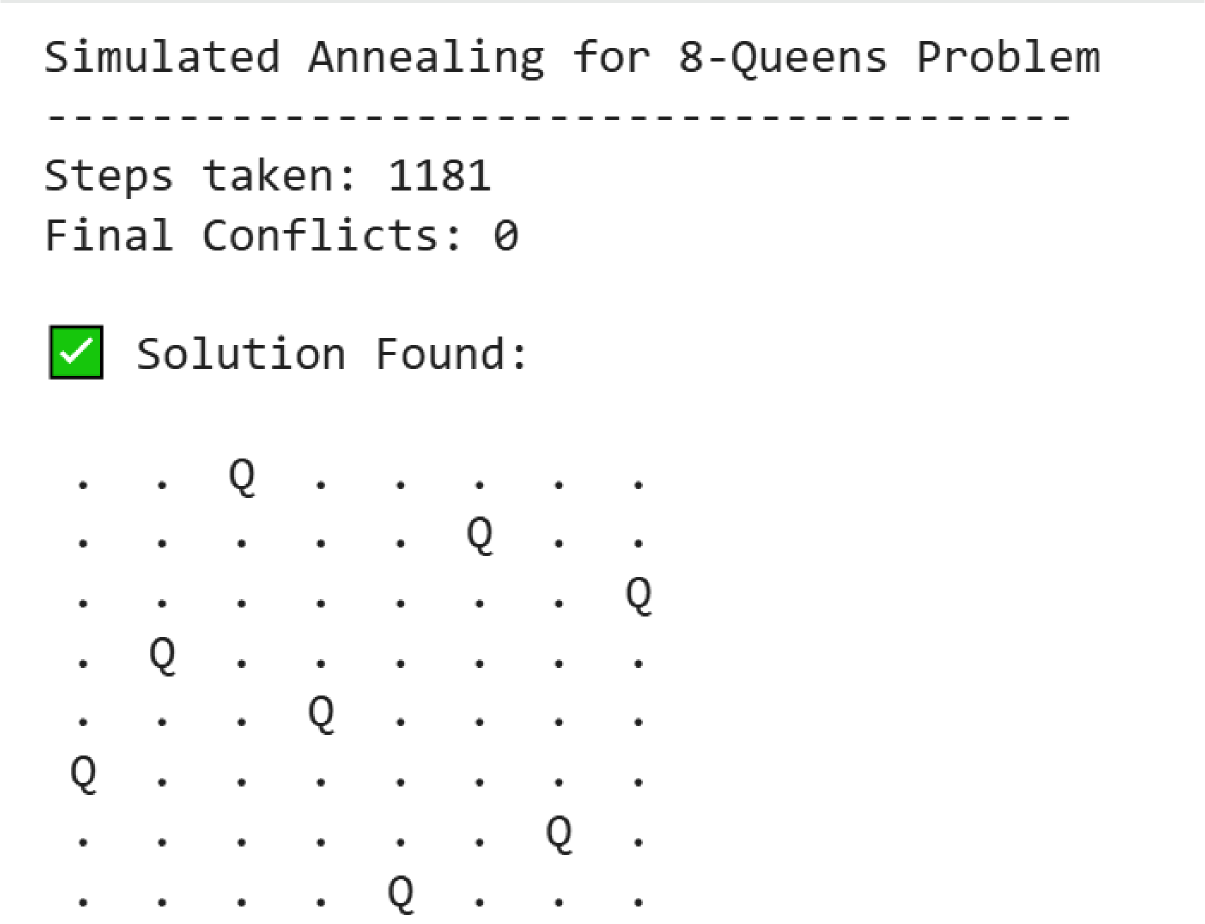
Simulated Annealing

To Solve 8-Queens problem

```
# Simulated Annealing Algorithm to solve 8-Queens Problem
# Date: 5 November 2024
# College: BMSCE

import random
import math

N = 8  # Number of queens

# ----- Generate a random state -----
def random_state():
    """Generate a random arrangement of queens (one per column)."""
    return [random.randint(0, N - 1) for _ in range(N)]

# ----- Heuristic: Number of attacking pairs -----
def compute_conflicts(state):
    """Calculate number of pairs of queens attacking each other."""
    conflicts = 0
    for i in range(N):
        for j in range(i + 1, N):
            if state[i] == state[j] or abs(state[i] - state[j]) ==
abs(i - j):
                conflicts += 1
    return conflicts

# ----- Generate a random neighbor -----
def random_neighbor(state):
    """Generate a neighboring state by moving one queen to a new
row."""
    neighbor = state[:]
    col = random.randint(0, N - 1)
    new_row = random.randint(0, N - 1)
    neighbor[col] = new_row
    return neighbor

# ----- Simulated Annealing algorithm -----
def simulated_annealing(max_iterations=100000, initial_temp=1000,
cooling_rate=0.99):
    """Solve N-Queens using Simulated Annealing."""
    current = random_state()
```

```python
    current_cost = compute_conflicts(current)
    T = initial_temp

    for step in range(max_iterations):
        if current_cost == 0:
            break

        neighbor = random_neighbor(current)
        neighbor_cost = compute_conflicts(neighbor)
        delta = neighbor_cost - current_cost

        # Acceptance condition
        if delta < 0 or random.uniform(0, 1) < math.exp(-delta / T):
            current = neighbor
            current_cost = neighbor_cost

        T *= cooling_rate  # Cool down

        if T < 1e-3:  # Stop if temperature is too low
            break

    return current, current_cost, step

# ----- Display Board -----
def print_board(state):
    """Pretty print the board."""
    for i in range(N):
        row = ""
        for j in range(N):
            row += " Q " if state[j] == i else " . "
        print(row)
    print("\n")


# ----- Run the Algorithm -----
solution, cost, steps = simulated_annealing()

print("Simulated Annealing for 8-Queens Problem")
print("----------------------------------------")
print(f"Steps taken: {steps}")
print(f"Final Conflicts: {cost}\n")

if cost == 0:
    print("✅ Solution Found:\n")
else:
    print("⚠️ Approximate Solution (local minimum):\n")

print_board(solution)
```

OUTPUT:



```
Simulated Annealing for 8-Queens Problem
---------------------------------------
Steps taken: 1181
Final Conflicts: 0

✅ Solution Found:

 .  .  Q  .  .  .  .  .
 .  .  .  .  .  Q  .  .
 .  .  .  .  .  .  .  Q
 .  Q  .  .  .  .  .  .
 .  .  .  Q  .  .  .  .
 Q  .  .  .  .  .  .  .
 .  .  .  .  .  .  Q  .
 .  .  .  .  Q  .  .  .
```