SIMULATED ANNEALING FOR N-QUEENS

```
import random
import math

def print_board(board, n):
    """Prints the current state of the board."""
    for row in range(n):
        line = ""
        for col in range(n):
            if board[col] == row:
                line += " Q "  # Queen is represented by "Q"
            else:
                line += " . "  # Empty space represented by "."
        print(line)
    print()

def calculate_conflicts(board, n):
    """Calculates the number of conflicts (attacks) between queens."""
    conflicts = 0
    for i in range(n):
        for j in range(i + 1, n):
            # Check if queens are in the same row or diagonal
            if board[i] == board[j] or abs(board[i] - board[j]) == abs(i - j):
                conflicts += 1
    return conflicts

def simulated_annealing(n, initial_temp=1000, cooling_rate=0.995,
max_iterations=10000):
    """Simulated Annealing algorithm to solve N-Queens with detailed
steps."""
    # Initial random board configuration (one queen in each column)
    board = [random.randint(0, n - 1) for _ in range(n)]
    current_conflicts = calculate_conflicts(board, n)
    temperature = initial_temp
```

```
    iteration = 0

    print("Initial board:")
    print_board(board, n)
    print(f"Initial conflicts: {current_conflicts}\n")

    while current_conflicts > 0 and iteration < max_iterations:
        # Generate a neighboring state by moving a queen to another row in
its column
        col = random.randint(0, n - 1)
        original_row = board[col]
        new_row = random.randint(0, n - 1)
        while new_row == original_row:
            new_row = random.randint(0, n - 1)  # Ensure we are moving the
queen to a new row
        board[col] = new_row

        # Calculate the number of conflicts in the new configuration
        new_conflicts = calculate_conflicts(board, n)

        # Display the current step, board, and conflicts
        print(f"Iteration {iteration + 1}:")
        print(f"Temperature: {temperature:.2f}")
        print(f"Trying to move queen in column {col} from row {original_row}
to row {new_row}")
        print_board(board, n)
        print(f"New conflicts: {new_conflicts}, Current conflicts:
{current_conflicts}")

        # If the new state has fewer conflicts, accept it.
        # If the new state has more conflicts, accept it with a certain
probability.
        if new_conflicts < current_conflicts or random.random() <
math.exp((current_conflicts - new_conflicts) / temperature):
            current_conflicts = new_conflicts
            print("Move accepted.\n")
        else:
```

```
            # If no improvement, revert the move
            board[col] = original_row
            print("Move rejected. Reverting to previous state.\n")

        # Reduce the temperature according to the cooling schedule
        temperature *= cooling_rate

        iteration += 1

    return board, current_conflicts

def main():
    # Input dynamic parameters
    print("Welcome to the N-Queens Problem Solver using Simulated
Annealing!")
    n = int(input("Enter the size of the board (N): "))
    initial_temp = float(input("Enter the initial temperature (e.g., 1000): "))
    cooling_rate = float(input("Enter the cooling rate (e.g., 0.995): "))
    max_iterations = int(input("Enter the maximum number of iterations (e.g.,
10000): "))
    print("Tanush Prajwal S")
    print("1BM22CS304")

    solution, conflicts = simulated_annealing(n, initial_temp, cooling_rate,
max_iterations)

    print("Final solution:")
    print_board(solution, n)
    if conflicts == 0:
        print("A solution was found with no conflicts!")
    else:
        print(f"No solution was found after {max_iterations} iterations. Final
number of conflicts: {conflicts}")

if __name__ == "__main__":
    main()
```

```
Welcome to the N-Queens Problem Solver using Simulated Annealing!
Enter the size of the board (N): 4
Enter the initial temperature (e.g., 1000): 100
Enter the cooling rate (e.g., 0.995): 0.95
Enter the maximum number of iterations (e.g., 10000): 1000
Tanush Prajwal S
1BM22CS304
Initial board:
 .  .  .  .
 Q  Q  .  Q
 .  .  Q  .
 .  .  .  .

Initial conflicts: 5
```

```
New conflicts: 1, Current conflicts: 1
Move accepted.

Iteration 7:
Temperature: 73.51
Trying to move queen in column 1 from row 3 to row 2
 .  .  Q  .
 Q  .  .  .
 .  Q  .  .
 .  .  .  Q

New conflicts: 1, Current conflicts: 1
Move accepted.

Iteration 8:
Temperature: 69.83
Trying to move queen in column 1 from row 2 to row 1
 .  .  Q  .
 Q  Q  .  .
 .  .  .  .
 .  .  .  Q

New conflicts: 3, Current conflicts: 1
Move rejected. Reverting to previous state.

Iteration 9:
Temperature: 66.34
Trying to move queen in column 0 from row 1 to row 0
 Q  .  Q  .
 .  .  .  .
 .  Q  .  .
 .  .  .  Q

New conflicts: 2, Current conflicts: 1
Move accepted.

Iteration 10:
Temperature: 63.02
Trying to move queen in column 1 from row 2 to row 0
 Q  Q  Q  .
 .  .  .  .
 .  .  .  .
 .  .  .  Q

New conflicts: 4, Current conflicts: 2
```

```
Iteration 17:
Temperature: 44.01
Trying to move queen in column 1 from row 1 to row 0
 .  Q  Q  .
 .  .  .  Q
 Q  .  .  .
 .  .  .  .

New conflicts: 3, Current conflicts: 5
Move accepted.

Iteration 18:
Temperature: 41.81
Trying to move queen in column 2 from row 0 to row 3
 .  Q  .  .
 .  .  .  Q
 Q  .  .  .
 .  .  Q  .

New conflicts: 0, Current conflicts: 3
Move accepted.

Final solution:
 .  Q  .  .
 .  .  .  Q
 Q  .  .  .
 .  .  Q  .

A solution was found with no conflicts!
PS C:\Users\Admin>
```