

6. Implement Simulated Annealing to solve N-Queens problem.

```
import random
import math

def calculate_conflicts(board):
    conflicts = 0
    n = len(board)
    for i in range(n):
        for j in range(i + 1, n):
            # Check if two queens are attacking each other
            if board[i] == board[j] or abs(board[i] - board[j]) == j - i:
                conflicts += 1
    return conflicts

# Function to simulate the annealing process
def simulated_annealing(board, max_iterations=10000, initial_temperature=1000,
cooling_rate=0.99):
    current_board = board[:]
    current_cost = calculate_conflicts(current_board)
    temperature = initial_temperature

    for iteration in range(max_iterations):
        if current_cost == 0:
            return current_board, iteration

        new_board = current_board[:]
        queen_index = random.randint(0, len(board) - 1)
        new_board[queen_index] = random.randint(0, len(board) - 1)

        new_cost = calculate_conflicts(new_board)

        if new_cost < current_cost or random.random() < math.exp((current_cost -
new_cost) / temperature):
            current_board = new_board
            current_cost = new_cost

    print(f"Iteration {iteration}: Cost = {current_cost}, Temperature =
{temperature:.2f}")
    print_board(current_board)
    print("\n")
```

```

        temperature *= cooling_rate

    if temperature < 1e-3:
        break

    return current_board, iteration


def print_board(board):
    n = len(board)
    for i in range(n):
        row = ['Q' if board[j] == i else '.' for j in range(n)]
        print(" ".join(row))


def main():
    n = int(input("Enter the number of queens: "))

    print("Enter the initial positions of the queens as a list of row indices (0-indexed):")
    board = list(map(int, input().split()))

    if len(board) != n:
        print("Error: The number of positions provided does not match the number of queens.")
        return

    solution, iterations = simulated_annealing(board)

    print(f"\nSolution found in {iterations} iterations:")
    print_board(solution)

main()

print("Varsha P(1BMS22CS320)")

```

Output:

```

Enter the number of queens: 4
Enter the initial positions of the queens as a list of row indices (0-indexed):
3 1 2 0
Iteration 0: Cost = 4, Temperature = 1000.00
. . . Q
. . . .
. Q Q .
Q . . .

Iteration 1: Cost = 4, Temperature = 990.00
. . . .
. . . .
. Q Q Q
Q . . .

```

Last 5 Iterations:

Iteration 152: Cost = 4, Temperature = 217.04

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

Iteration 153: Cost = 6, Temperature = 214.87

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

Iteration 154: Cost = 4, Temperature = 212.73

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

Iteration 155: Cost = 3, Temperature = 210.60

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

Iteration 156: Cost = 1, Temperature = 208.49

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

Iteration 157: Cost = 0, Temperature = 206.41

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

Solution found in 158 iterations:

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

Varsha P(1BMS22CS320)