N-Queens using Simulated Annealing Algorithm:

6. N-Queens using Simulated Annealing

```
function SimulatedAnnealing():
    current ← initial state
    T ← a large positive value
    while T > 0 do
        next ← a random neighbour of current
        ΔE ← current.cost - next.cost
        if ΔE > 0 then
            current ← next
        else
            current ← next with probability p = e^(ΔE/T)
        end if
        decrease T
    end while
    return current
```

Output:

```
Enter the size of the board (N): 4
Enter the initial configuration of queens (one queen per row)
Row 1: Enter the column index for queen (0 to 3): 3
Row 2: Enter the column index for queen (0 to 3): 2
Row 3: Enter the column index for queen (0 to 3): 1
Row 4: Enter the column index for queen (0 to 3): 0

Final solution: [1, 3, 0, 2]
_ Q _ _
_ _ _ Q
Q _ _ _
_ _ Q _
Energy: 0
```

```python
import math
import random

def calculate_energy(board):
    """Calculate the number of attacking pairs of queens."""
    n = len(board)
    attacks = 0
    for i in range(n):
        for j in range(i + 1, n):
            if board[i] == board[j]:
                attacks += 1
            if abs(board[i] - board[j]) == abs(i - j):
                attacks += 1
    return attacks

def simulated_annealing_with_initial_board(initial_board,
initial_temp=1000, cooling_rate=0.95, max_iter=10000):
    """Solve the N-Queen problem using simulated annealing with an
initial board configuration."""
    n = len(initial_board)
    board = initial_board[:]
    current_energy = calculate_energy(board)
    temperature = initial_temp
    best_board = board[:]
    best_energy = current_energy

    for iteration in range(max_iter):
        if current_energy == 0:
            break

        row = random.randint(0, n-1)
        new_column = random.randint(0, n-1)

        while new_column == board[row]:
            new_column = random.randint(0, n-1)

        new_board = board[:]
        new_board[row] = new_column

        new_energy = calculate_energy(new_board)
        energy_diff = new_energy - current_energy

        if energy_diff < 0 or random.random() < math.exp(-energy_diff /
temperature):
            board = new_board
            current_energy = new_energy

        if current_energy < best_energy:
            best_board = board[:]
            best_energy = current_energy

        temperature *= cooling_rate
```

```
    return best_board, best_energy

if __name__ == "__main__":
    n = int(input("Enter the size of the board (N): "))
    print("Enter the initial configuration of queens (one queen per
row):")
    initial_board = []
    for i in range(n):
        column = int(input(f"Row {i+1}: Enter the column index for
queen (0 to {n-1}): "))
        initial_board.append(column)

    solution, energy =
simulated_annealing_with_initial_board(initial_board)

    print("\nFinal solution:", solution)
    for i in range(n):
      for j in range(n):
        if solution[i]==j:
          print("Q",end=" ")
        else:
          print("_",end=" ")
      print()
    print("Energy:", energy)
    print("Name:Vignesh Bhat \nUSN:1BM22CS327")
```

Output:

```
Enter the size of the board (N): 4
Enter the initial configuration of queens (one queen per row):
Row 1: Enter the column index for queen (0 to 3): 3
Row 2: Enter the column index for queen (0 to 3): 2
Row 3: Enter the column index for queen (0 to 3): 1
Row 4: Enter the column index for queen (0 to 3): 0

Final solution: [2, 0, 3, 1]
_ _ Q _
Q _ _ _
_ _ _ Q
_ Q _ _
Energy: 0
Name:Vignesh Bhat
USN:1BM22CS327
```