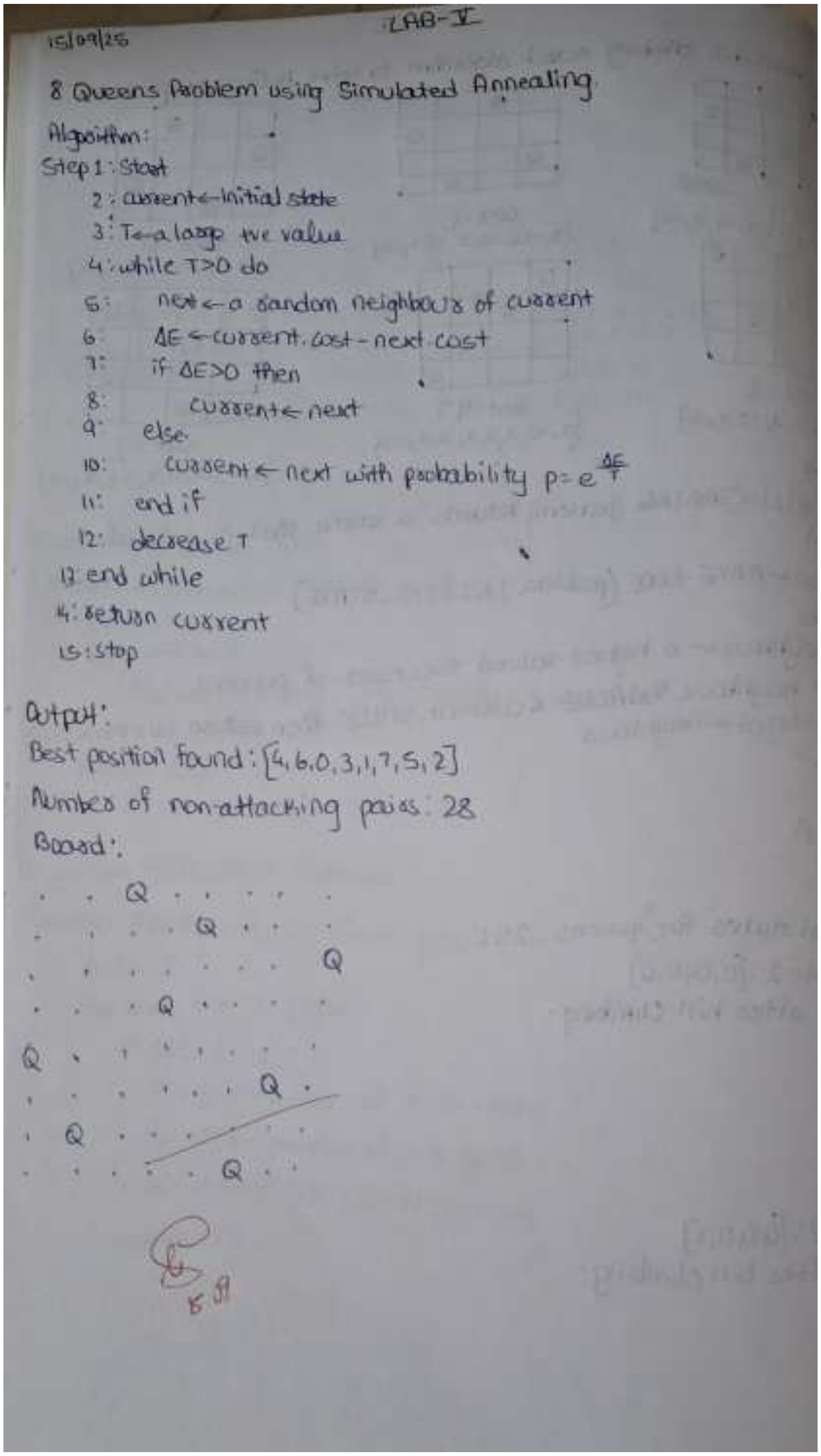
Lab 5

8 Queens Problem using Simulated Annealing

Algorithm:



15/09/25

LAB-V

8 Queens Problem using Simulated Annealing

Algorithm:

```
Step 1: Start
    2: current←initial state
    3: T←a large +ve value
    4: while T>0 do
    5:     next←a random neighbour of current
    6:     ΔE←current.cost - next cost
    7:     if ΔE>0 then
    8:         current←next
    9:     else
    10:        current←next with probability $p=e^{\frac{\Delta E}{T}}$
    11:    end if
    12:    decrease T
   13: end while
   14: return current
   15: stop
```

Output:

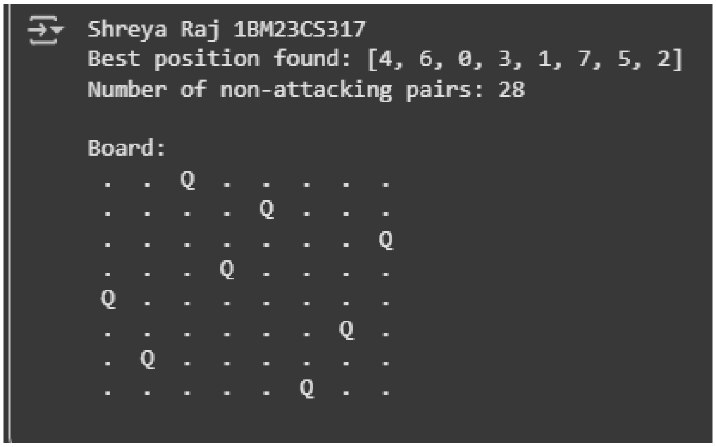
```
Best position found: [4,6,0,3,1,7,5,2]
Number of non-attacking pairs: 28
Board:
. . Q . . . . .
. . . . Q . . .
. . . . . . . Q
. . . Q . . . .
Q . . . . . . .
. . . . . . Q .
. Q . . . . . .
. . . . . Q . .
```

15/9

Code:

```
import random
import math
print("Shreya Raj 1BM23CS317")
def cost(state):
    attacks = 0
    n = len(state)
    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):
                attacks += 1
    return attacks

def get_neighbor(state):
    neighbor = state[:]
    i, j = random.sample(range(len(state)), 2)
    neighbor[i], neighbor[j] = neighbor[j], neighbor[i]
    return neighbor

def simulated_annealing(n=8, max_iter=10000):
    current = list(range(n))
    random.shuffle(current)
    current_cost = cost(current)

    temperature = 100.0
    cooling_rate = 0.95
    best = current[:]
    best_cost = current_cost

    for _ in range(max_iter):
        if temperature <= 0 or best_cost == 0:
            break

        neighbor = get_neighbor(current)
        neighbor_cost = cost(neighbor)
        delta = current_cost - neighbor_cost

        if delta > 0:
            current, current_cost = neighbor, neighbor_cost
            if neighbor_cost < best_cost:
                best, best_cost = neighbor, neighbor_cost
        else:
            probability = math.exp(delta / temperature)
            if random.random() < probability:
                current, current_cost = neighbor, neighbor_cost

        temperature *= cooling_rate

    return best, best_cost

def print_board(state):
    n = len(state)
    for row in range(n):
        line = ""
        for col in range(n):
            if state[col] == row:
                line += " Q "
            else:
                line += " . "
        print(line)
    print()

if __name__ == "__main__":
    n = 8
    solution, cost_val = simulated_annealing(n)

    print("Best position found:", solution)
    print(f"Number of non-attacking pairs: {n*(n-1)//2 - cost_val}")
    print("\nBoard:")
    print_board(solution)
```

Output:



```
Shreya Raj 1BM23CS317
Best position found: [4, 6, 0, 3, 1, 7, 5, 2]
Number of non-attacking pairs: 28

Board:
 .  .  Q  .  .  .  .  . 
 .  .  .  .  Q  .  .  . 
 .  .  .  .  .  .  .  Q 
 .  .  .  Q  .  .  .  . 
 Q  .  .  .  .  .  .  . 
 .  .  .  .  .  .  Q  . 
 .  Q  .  .  .  .  .  . 
 .  .  .  .  .  Q  .  . 
```