LAB - 4 22/9/25

# N-Queens Problem using Hill Climbing Approach

Algorithm :

Step 1 : Initialize

- Generate a random initial state (current_state) where each queen is in a random column of its row
- Calculate the number of attacking pairs current_cost for current_state.

Step 2 : Repeat until solution found.

- Generate all neighbors of current_state:
  * For each row, move the queen to every other column in that row
  * Each move creates a neighbour state
- Calculate the cost (attacking pairs) for each neighbour
- Select the neighbour with the lowest cost (best-neighb
-our)

Step 3 : Move or stop

- If best_neighbor cost < current_cost:
  * Set current_state = best_neighbor
  * Set current_cost = best_neighbor cost.
- Else (no improvement):
  * Stop (local maximum reached or solution found)

Step 4 : Check solution

- If current_cost == 0 → solution found
- Else → may start with a new random state

Step 5 : Return

current_state as the solution

Output :

Initial state : [3, 1, 2, 0]

| | | | |
|---|---|---|---|
| | | | Q |
| | Q | | |
| | | Q | |
| Q | | | |

Attacking pairs 2

Neighbors and their conflicts

(1, 3, 2, 0) → Attacking pair : 1

| | | | |
|---|---|---|---|
| | Q | | |
| | | | Q |
| | | Q | |
| Q | | | |

→ (2, 1, 3, 0) → Attacking pairs : 1

| | | | |
|---|---|---|---|
| | | Q | |
| | Q | | |
| | | | Q |
| Q | | | |

→ (0, 1, 2, 3) → Attacking pairs 6

| | | | |
|---|---|---|---|
| Q | | | |
| | Q | | |
| | | Q | |
| | | | Q |

→ (3, 2, 1, 0) → Attacking pairs 6

| | | | |
|---|---|---|---|
| | | | Q |
| | | Q | |
| | Q | | |
| Q | | | |

→ (3, 0, 2, 1) → Attacking pairs : 1

| | | | |
|---|---|---|---|
| | | | Q |
| Q | | | |
| | | Q | |
| | Q | | |

→ ( 3, 1, 0, 2) → Attacking pairs : 1

| | | | |
|---|---|---|---|
| | | | Q |
| | Q | | |
| Q | | | |
| | | Q | |

Chosen state [1, 3, 2, 0] attacking pairs: 1

Step 2: Current state: [1, 3, 2, 0] attacking pairs

Neighbours and their cats :

→ (3,1,2,0) Attacking pairs : 2

| | | | |
|---|---|---|---|
| | | | Q |
| | Q | | |
| | | Q | |
| Q | | | |

→ (2,3,1,0) → Attacking pairs : 2

| | | | |
|---|---|---|---|
| | | Q | |
| | | | Q |
| | Q | | |
| Q | | | |

→ (0,3,2,1) → Attacking pairs : 4

| | | | |
|---|---|---|---|
| Q | | | |
| | | | Q |
| | | Q | |
| | Q | | |

→ (1,2,3,0) → Attacking pairs : 4

| | | | |
|---|---|---|---|
| | Q | | |
| | | Q | |
| | | | Q |
| Q | | | |

→ (1,0,2,3) → Attacking pairs : 2

| | | | |
|---|---|---|---|
| | Q | | |
| Q | | | |
| | | Q | |
| | | | Q |

→ ( 1, 3, 0, 2 ) → Attacking pairs : 0

| | | | |
|---|---|---|---|
| | Q | | |
| | | | Q |
| Q | | | |
| | | Q | |

Chosen state : [ 1, 3, 0, 2 ]
Final solution : [ 1, 3, 0, 2 ]

22/9/25

# N-Queens Problem Using Simulated Annealing

Algorithm:

1) Initialize
   - Generate a random current-state (queen positions)
   - Set temperature T = initial_temperature.

2) Repeat while T > min_temperature.
   - Pick a random neighbor by moving one queen
   - Calculate its attacking pairs (neighbor_cost)
   - If neighbor is better accept it.
   - If worse accept it with probability exp(-(neighbor_cost - current_cost)/T).
   - Reduce temperature : T = T * alpha.
   - Stop if attacking pairs = 0 (solution found)

3) Return the final state (solution or best found)

Output:

Initial state : [0, 0, 3, 2]    Attacking pairs : 3

| | | | |
|---|---|---|---|
| Q | | | |
| Q | | | |
| | | | Q |
| | | Q | |

Step 1: Current state : [0, 0, 2, 2]    Attacking pairs : 4
Temperature : 100.00

| | | | |
|---|---|---|---|
| Q | | | |
| Q | | | |
| | | Q | |
| | | Q | |

Step 2: Current state : [0, 0, 2, 1]    Attacking pairs : 3
Temperature : 95.00

| | | | |
|---|---|---|---|
| Q | | | |
| Q | | | |
| | | Q | |
| | Q | | |

Step 3: Current State : [2, 0, 2, 1] , Attacking pairs 2.
Temperature 90.25

| | | | |
|---|---|---|---|
| | | Q | |
| Q | | | |
| | | Q | |
| | Q | | |

step 4: Current State: [2, 0, 3, 1] Attacking pairs: 0

Temperature: 85.74

| | | | |
|---|---|---|---|
| | | Q | |
| Q | | | |
| | | | Q |
| | Q | | |

Final state: [2, 0, 3, 1], Attacking pairs: 0

22/9/25