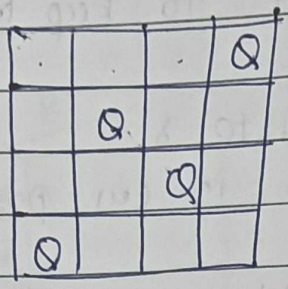
15/09/25

Lab-5.

Implementing Hill Climbing search algorithm to solve N-Queen problems.

Input:



=> ① - - - Q
- Q - -
- - Q -
Q - - -

total cost = 5.
(1,4) ↪ (2,2) (2,2) ↪ (3,3)
(4,4) ↪ (3,3) (2,2) ↪ (4,1)
(3,3) ↪ (4,1)

②
- - - -
- Q - Q
- - Q -
Q - - -

total cost = 2.
(2,2) ↪ (3,2)
(2,4) ↪ (3,2)

③ - - - -
- Q - Q
Q - Q -
Q - - -

(2,2) ↪ (3,3)
(2,4) ↪ (3,3)

① - - - Q
- Q - -
- - Q -
Q - - -

(2,2) ↪ (3,3) - ①
(1,4) ↪ (4,1) - ②

cost = 2.

② - - - -
- Q - Q
- - Q -
Q - - -

(2,2) ↪ (2,4) - ①
(2,2) ↪ (3,3) - ②
(3,3) ↪ (2,4) - ③

cost = 3

(3) - - Q -
- Q - Q
- - - -
Q - - -

(1,3) ↔ (2,2) - (1)
(2,2) ↔ (2,4) - (2)
(1,3) ↔ (2,4) - (3)
cost = 3.

(4) - - - -
- - - Q
- Q Q -
Q - - -

(2,4) ↔ (3,3) - (1)
(3,2) ↔ (3,1) - (2)
(3,2) ↔ (4,1) - (3)
cost = 3.

(5) - Q - -
- - - Q
Q - - -
- - Q -

cost = 0

Simulated Annealing

Algorithm

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 $p = e^{\Delta E/T}$
  end if
  decrease T
end while
return current

o/p
The best position found is: [0 8 5 2 6 3 7 1]
The number of queues that are not attacking each other is : 8

Algorithm of hill climbing.

1. start with one queen in each col (intial board)
2. Calculate cost (c) = no. of attacking queen pairs.
3. for each col, move the queen to every other row & compute next cost
4. Choose the move that gives the lowest cost (best neighbor)
5. If best cost < cur cost, move queen there & repeat step 2
6. if no neighbor has lowest cost, Stop

o/p :

1. initial state
- - - Q
- Q - -
- - Q -
Q - - -
cost = 2

2. - - - Q
- Q - -
- - - -
Q - Q -
cost 2

3. - - - Q
- Q - -
Q - - -
- - Q -
cost 1

4. - Q - Q
- - - -
Q - - -
- - Q -
cost=1

5. - Q - -
- - - Q
Q - - -
- - Q -
cost=

Sol. found in 4 steps.