

=> Implement Hill climbing search algorithm to solve N-Queens problem.

	1	2	3	4
1		Q		Q
2			Q	
3				Q
4	Q			

=> Initial input board.

(row, col)

(1,4) (2,2) (3,3) (4,1)

Step 1: Initial State

			Q
	Q		
		Q	
Q			

Attacking pairs

(2,2) \leftrightarrow (3,3)

(1,4) \leftrightarrow (4,1)

both diagonal

Cost = 2

Step 2:

			Q
Q	Q		Q
		Q	
Q			

Attacking pairs:

(1,4) \leftrightarrow (1,2)

(2,2) \leftrightarrow (2,4)

(2,2) \leftrightarrow (3,3)

(2,4) \leftrightarrow (3,3)

Cost = 3

Step 3:

		Q	
	Q		Q
Q			

A.P =>

(1,3) \leftrightarrow (2,2)

(1,3) \leftrightarrow (2,4)

(2,2) \leftrightarrow (2,4)

Cost = 3

Step 4:

			Q
		Q	
	Q	Q	
Q			

A.P. \Rightarrow

$(2,4) \leftrightarrow (3,3)$

$(3,2) \leftrightarrow (4,1)$

$(3,2) \leftrightarrow (3,3)$

Cost = 3

Step 5:

	Q		
Q			Q
Q			
		Q	

A.P.

no attacking pairs
(no diagonal, no same
row & column)

Algorithm

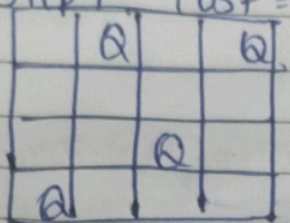
1. start with one queen in each column (initial board)
2. calculate cost(h) = no of attacking queen pairs
3. for each column, move the queen to every other row & compute new cost
4. choose the move that gives the lowest cost (best neighbor)
5. if best cost < cur cost, move queen there & repeat step 2.
6. if no neighbor has lowest cost, STOP.

o/p: $\leftrightarrow (6,8)$

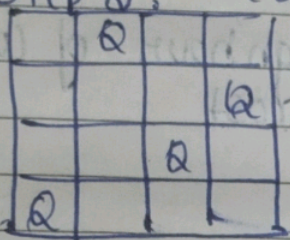
initial state (cost = 2):

			Q
	Q		
		Q	
Q			

Step 1: Cost = 2



Step 2: Cost = 1



Step 3: Cost = 1



Step 4: Cost = 0



Soln found in 4 steps.

Initial State (cost=2):

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

Step 1 (cost=2):

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

Step 2 (cost=1):


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

Step 3 (cost=1):

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

Step 4 (cost=0):

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

Solution found in 4 steps 

Simulated Annealing

Algorithm:

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

Output:

~~The best position found is: (0 8 5 2 6 3 7 4)
The no of queens that are not attacking each other is: 8~~

```
The best position found is: [6, 3, 1, 7, 4, 2, 0, 5]  
The number of queens that are not attacking each other is: 8
```