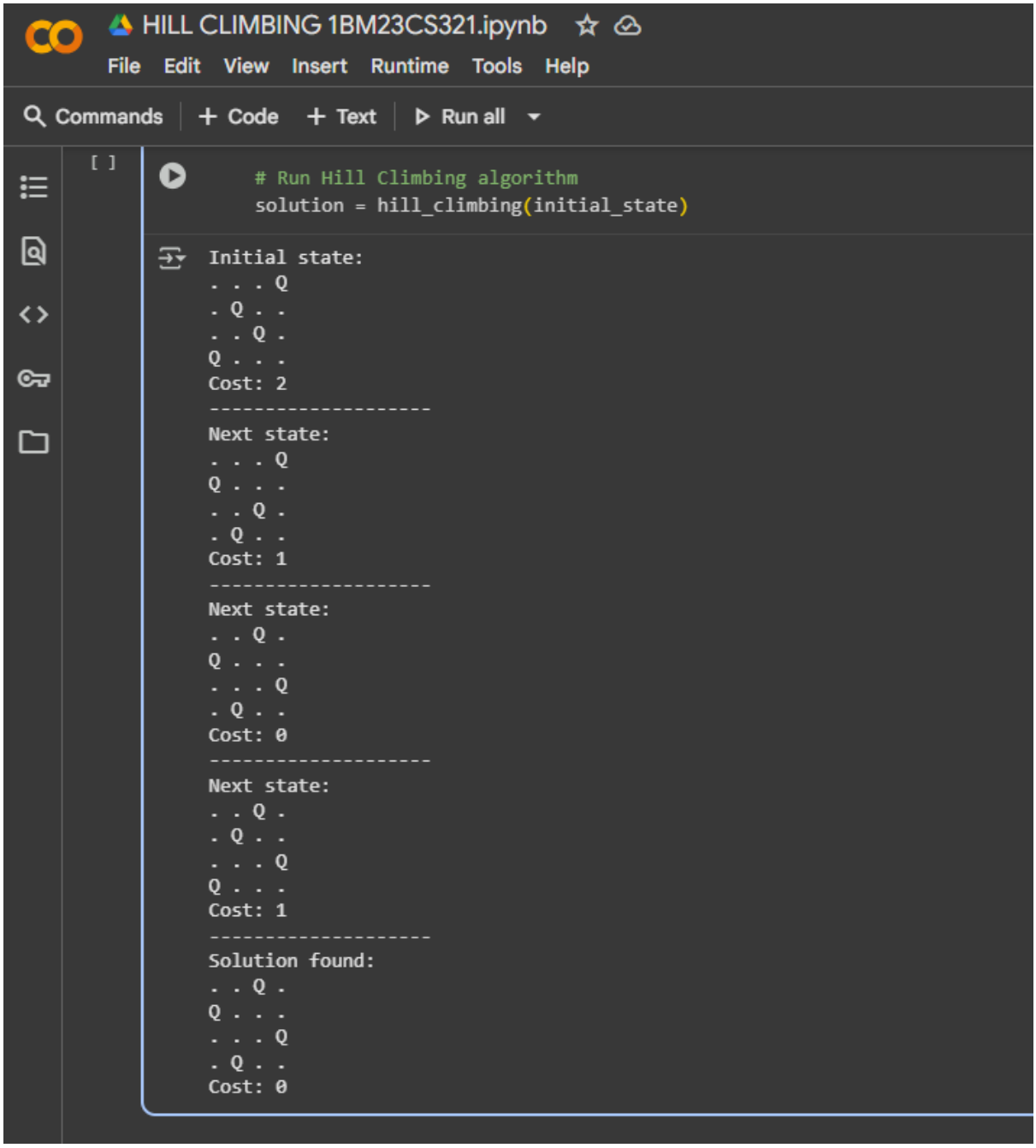
```python
import random

def cost(state):

    attacking_pairs = 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):
                attacking_pairs += 1
    return attacking_pairs

def print_board(state):

    n = len(state)
    board = [['.' for _ in range(n)] for _ in range(n)]
    for i in range(n):
        board[state[i]][i] = 'Q'

    for row in board:
        print(" ".join(row))

def get_neighbors(state):

    neighbors = []
    n = len(state)
    for i in range(n):
        for j in range(i + 1, n):
            neighbor = list(state)
            neighbor[i], neighbor[j] = neighbor[j], neighbor[i]
            neighbors.append(tuple(neighbor))
    return neighbors

def hill_climbing(initial_state):

    current = initial_state
    print(f"Initial state:")
    print_board(current)
    print(f"Cost: {cost(current)}")
    print('-' * 20)

    while True:
        neighbors = get_neighbors(current)

        next_state = min(neighbors, key=lambda x: cost(x))
        print(f"Next state:")
        print_board(next_state)
```

```
        print(f"Cost: {cost(next_state)}")
        print('-' * 20)

        if cost(next_state) >= cost(current):

            print(f"Solution found:")
            print_board(current)
            print(f"Cost: {cost(current)}")
            return current
        current = next_state

if __name__ == "__main__":

    initial_state = (3, 1, 2, 0)


    solution = hill_climbing(initial_state)
```


HILL CLIMBING 1BM23CS321.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
[ ]
# Run Hill Climbing algorithm
solution = hill_climbing(initial_state)
Initial state:
. . . Q
. Q . .
. . Q .
Q . . .
Cost: 2
Next state:
. . . Q
Q . . .
. . Q .
. Q . .
Cost: 1
Next state:
. . Q .
Q . . .
. . . Q
. Q . .
Cost: 0
Next state:
. . Q .
. Q . .
. . . Q
Q . . .
Cost: 1
Solution found:
. . Q .
Q . . .
. . . Q
. Q . .
Cost: 0

LAB - V

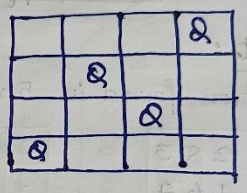
Hill Climbing Search Algorithm.

Function Hill climbing (problem) returns a state that is a local maximum

curr ← Make Node (problem INITIAL-STATE)
loop do
  neighbours ← a highest valued successor
  if neighbour value ≤ curr.value then
    return curr.state
  curr ← neighbour

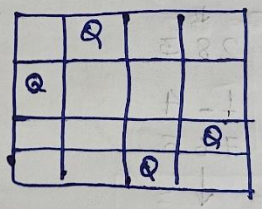
- $x_0 = 3,\ x_1 = 1,\ x_2 = 2,\ x_3 = 0$

  cost = 2.



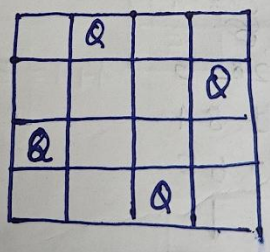
- $x_0 = 1,\ x_1 = 0,\ x_2 = 3,\ x_3 = 2.$

  cost = 2+1+1 = 4



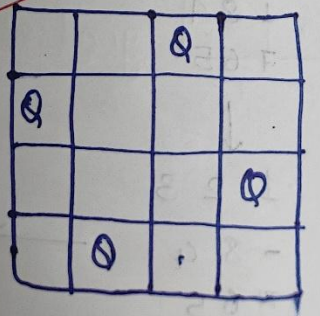
- $x_0 = 1,\ x_1 = 3,\ x_2 = 0,\ x_3 = 2.$

  cost = 0.



- $x_0 = 3,\ x_1 = 2,\ x_2 = 0,\ x_3 = 1$

  cost = 2.



Output ⇒

Enter no. of queens : 8

Solution found at step 623

Position format :

3 1 7 4 6 0 2 5

Hueristic 0

Output :=

Enter the no queens (N) : 4

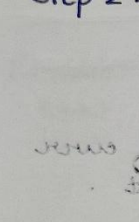
Initial state (hueristic 4) :

Q . Q .
. . . Q
. . . .
. . Q .

Step 1 : (h = 1)

Q . . .
. . . Q
Q . . .
. . Q .

Step 2 : (h = 0)



Reached local min at step 2, h = 0

Solution found after restart.