PAGE :
DATE : 29/10/24

1. 8 queens using Hill climbing :-

Step 1 : Initialization :-
Generate a random initial state, [list of integers] representing the positions of the queens on the chessboard. Each index of the index.

Step 2 : Calculate Attack :
calculate_attacks(state)
This counts how many pairs of queens are attacking each other.
2 queens attack each other if
→ state[i] == state[j] [same row & column]
→ (abs(state[i] - state[j]) == j - 1) [diagonal]

Step 3 : Loop : > Generate all possible neighbor states by moving each queen to every column in its row except its column.
> Calculate the no of attacks for each neighbor state
> Select the neighbor with the fewest attack.
if not :
exit

Track : Keep checking
If a soln is found with zero attack, exit & display the board state.

PAGE :
DATE : / /

2. 8 queens problem using A* search algorithm

Step 1 Create an initial state with all queens placed.
> Initialize the open set & push the initial node onto it.
> Create a visited set to track explored states.

Step 2. main loop:
> when nodes are in the open set:
1. pop the node with the lowest f value (best node)
2. check if the current node represents a goal state (no queens attacks each other.)
if it is a goal state
return the current state
if the state has already been visited, skip to the next iteration.
↳ Add the current state to visited set

Determine the next row to place a queen.
1. Generate new states by placing queens in each col of the next row.
calculate g, h & f values & push it into open state.

Step3: Display function:
Call the function to find a sol^n
if a sol^n is found
display the board
otherwise, print no solution exit

o/p
. . . . . Q
. Q . . . . . .
. . . Q . . . .
Q . . . . . .
. . . . . Q .
. . . . Q . . .
. . Q . . . . .
. . . Q . .

24/10/24