ARTIFICIAL INTELLIGENCE

Assignment 1

Name : S.A.M.A.T.N. Senanayake

Reg. No. : ENG/15/102

Stream : Electrical & Electronic Engineering (EE)

Intake 32

Problem : Create a program to solve the 8-Queen problem.

GitHub URL : https://github.com/thilanse/8-Queen-Hill-Climbing---Thilan.git

---

The hill climbing approach was used in the program to solve the 8 Queen problem. In this, a heuristic cost (h cost) was introduced, which is the number of queens attacking each other for a specific board state. As we know, the Queen can move in all directions, hence, if any other Queen is placed in its path, the Queen can attack.

The way I approached this was by creating a function which determines this h cost by checking whether other Queens are placed on the same row, or on the diagonals of the Queen being checked. It was assumed that all 8 Queens are placed on different columns, which reduces the complexity of the problem. Therefore, initially h is set to 0, and then increased by 1 every time another Queen is in the line-of-attack of the Queen being checked. This function was used to determine the h cost for all the relevant states of the board.

Since only one Queen can be moved to reach the next state, there are 56 possible states which could be achieved from one move (8 Queens can be placed on any of the 7 other rows in its column). Therefore, the h cost was determined for all 56 neighboring states as well as for the current state. This will give some states which will have a lower h cost than other states. However, there can be many states with the same low h cost. Therefore, the program will choose randomly from all the next best states, and commit to the move.

This process will repeat until no better state could be reached. However, there is no guarantee that the state reached would have an h cost of 0 (which is the global maximum that needs to be achieved). The final state could have an h cost of 0, 1, 2, or other local maximum states depending on the path the program had decided to take.

In order to rectifying this problem, I made a small change to the code, such that if the final state is a local maximum, the board will be reinitialized and run through the program

again. This process will repeat until a global maximum solution has been achieved. In this way, the final solution provided by the program will definitely be a global maximum.

These are some of the results. More example results are included in the GitHub repository.

```
[[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 0]]
Q Q Q Q Q Q Q Q
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .

h = 28

[[0, 1], [1, 4], [2, 6], [3, 3], [4, 0], [5, 7], [6, 5], [7, 2]]
. . . . Q . . .
Q . . . . . . .
. . . . . . . Q
. . . Q . . . .
. Q . . . . . .
. . . . . Q . .
. . Q . . . . .
. . . . . Q . .

h = 0
Reached Global Maximum!

Steps taken:
[[0, 0, 0, 0, 0, 0, 0, 0], 28]
[[0, 0, 0, 0, 0, 7, 0, 0], 21]
[[0, 4, 0, 0, 0, 7, 0, 0], 15]
[[0, 4, 0, 0, 0, 7, 5, 0], 10]
[[0, 4, 0, 0, 0, 7, 5, 2], 6]
[[1, 4, 0, 0, 0, 7, 5, 2], 3]
[[1, 4, 0, 3, 0, 7, 5, 2], 1]
[[1, 4, 6, 3, 0, 7, 5, 2], 0]

Number of steps: 7
```

```
[[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 0]]
Q Q Q Q Q Q Q Q
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .

h = 28

[[0, 4], [1, 2], [2, 0], [3, 6], [4, 3], [5, 7], [6, 5], [7, 1]]
. . Q . . . . .
. . . . . . . Q
. Q . . . . . .
. . . . Q . . .
Q . . . . . . .
. . . . . Q . .
. . . Q . . . .
. . . . . Q . .

h = 1
Stuck at Local Maximum

Steps taken:
[[0, 0, 0, 0, 0, 0, 0, 0], 28]
[[0, 0, 0, 6, 0, 0, 0, 0], 21]
[[0, 2, 0, 6, 0, 0, 0, 0], 15]
[[0, 2, 0, 6, 0, 0, 5, 0], 10]
[[0, 2, 0, 6, 0, 0, 5, 1], 6]
[[0, 2, 0, 6, 0, 7, 5, 1], 3]
[[0, 2, 0, 6, 3, 7, 5, 1], 2]
[[4, 2, 0, 6, 3, 7, 5, 1], 1]

Number of steps: 7
```

As mentioned previously, there is no guarantee that the final solution would be global maximum. The program can get stuck at a local maximum, which is not the ideal solution that we are expecting. However, since the program decides on the next best move stochastically out of several best moves, the program may reach the global maximum based on the path taken. This is evidence of the drawback of the hill-climbing approach.

The second version of the program, which reinitializes the board if it reaches a local maximum, provides a guaranteed global maximum solution. However, the number of steps taken to reach the goal, might be very high on certain occasions.

Other approaches such as simulated annealing, might provide a better solution to reach the global maximum.