DATE:

PAGE:

Lab-6

Implement A* search algorithm for 8 Queens:

(1) Define the initial state:

⇒ Start with an empty board where no queens are placed

⇒ Initialize the open list (priority queue) with this initial state.

(2) Define the Heuristic Function (h):

⇒ For each state, calculate the number of conflicts between queens already placed.

⇒ This will serve as the heuristic estimate h(n), indicating how far the current state is from a solution.

(3) Define the Cost function (g):

⇒ Set the cost g(n) as the number of queens placed so far. In this setup, each placement has a cost of 1.

(4) Calculate the Evaluation Function (f):

⇒ For each state, calculate $f(n) = g(n) + h(n)$, where:

(i) g(n) is the cost to reach the current state.

(ii) h(n) is the heuristic estimation of conflicts.

⇒ The state with the lowest f(n) value is considered the most promising ~~to~~ expand next.

(5) Expand Nodes :

=> Pop the state with the lowest $f(n)$ from the open list.

=> If it has no conflicts (i.e, $h(n) = 0$) and all queens are placed, the goal is achieved, and the solution is found.

=> Otherwise, generate child states by placing a queen in the next row in each possible column.

(6) Calculate Heuristic for each child State :

=> For each child state, recalculate $f(n) = g(n) + h(n)$ based on the new queen placements.

=> Add each child state to the open list.

(7) Repeat :

=> Continue expanding nodes with the lowest $f(n)$ until a goal state (no conflicts) is found or the open list is empty.

(8) Goal check :

If a state where all queens are placed without conflicts is found, return it as the solution.

If no solution is found ~~[illegible]~~ report failure.

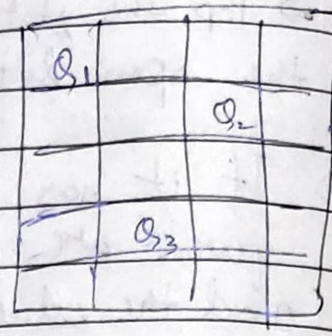
no. of queens present in board

no. of conflicts

$f(n) = g(n) + h(n)$

OL : node value ~~Q_1~~ ~~Q_2~~ ~~Q_3~~
~~1~~ ~~2~~ ~~3~~

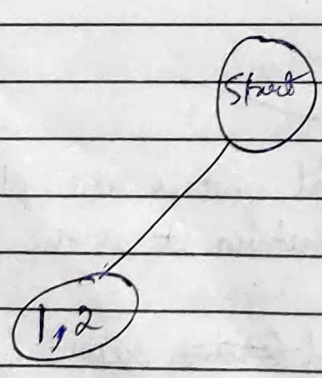
ans → CL : Q_1 Q_2 Q_3



Again backtrack

Output :
Best solution found

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



Implement Hill climbing Search for 8-queens:

Step 1: Create an array of where each index represents a column & the value represents the row-position of the queen in that row.

int[] q = new int[8]

q =							
0	1	2	3	4	5	6	7

~~Step2~~:

Step2: Initiate a random state where 8 queens are placed in a different column, but in a randomly choosen row.

~~Step 3 ? Store a heuristic value h(n) where h represents~~

Step 3: Evaluate the current state by using h(n) to evaluate how many pairs of queens are conflicting each other.

Step 4: ~~Select~~. Generate neighbors.
For each columns, try moving the queen to every possible row (except its current one) and generate neighboring states.

Step 5: ~~Mo~~ Select the ~~b~~ best neighbor with lowest heuristic value (h(n)) → total conflicts pairs

Step 6: Move to the new state if the new state's h(n) value is lower than the current one.

Step 7: If stuck, restart by backtracking.

Output : Best Solution found :

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