29/10/24 Lab-6 Date / / Page

Algorithm for 8 queens using A*

1. Initialisation
   - place queens randomly with one queen per column
   - Initialise priority queue with the start node and its heuristic

2. Expand nodes:
   - Remove the node with the lowest f value from priority queue.
   - If the node has no conflicts, return possible solutions.

3. Generate successors:
   - For each queen, move it to any row within its column
   - For each new configuration compute the g value and h value (h is the heuristic which is the conflicting positions)

4. Push the successors into priority queue:
   f = g+h for each successor and add to priority queue
   g = cost for each current state from start state

5. Repeat until solution
   Continue expanding and exploring successors until a solution is found

Date __/__/__
Page _____

Algorithm for 8 queens using Hill climbing

1. Initialization
   start with a random configuration where one queen is placed in each column

2. Iteratively Improve:
   - Compute the heuristic for the current state
   - If heuristic value is zero, the solution is found

3. Generate neighbouring states:
   - For each queen generate the neighbouring states by moving it to every possible row within its column and calculate the heuristic for each new state

4. Choose the best neighbor:
   - Select the neighbor with least heuristic
   - If no neighbors state improves the current state, terminate.

5. Repeat until solution is achieved

Date
Page

output for A* :

Solution: [1, 6, 4, 7, 0, 3, 5, 2]

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

output for Hillclimbing:

One of the solutions for 8-Queens problem:

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