LAB-6

Algorithm - A* for 8 queens

1) Initialize the priority queue with the start state, and queens are placed randomly in each column.
2) → Dequeue the node with lowest priority value.
→ If node has no (attacks) problems, return as solution.
3) Generate new states
→ for each row generate new states by moving each queen within a column
→ for each, generate g (increase by 1) and h (heuristic, no. of conflicting pairs)
4) Push into priority queue:
→ set f = g+h and add the successor to the priority queue.
5) Repeat until a solution is found or if all the ~~three~~ nodes are visited.

* Heuristic function:
→ Calculate the no. of queens attacking each other.

f = 3.600
= 1.0800
= 0.3240.

Algorithm - Hill climbing

1) Initialize the state with queens placed randomly in each column.

2) Iterative calculation.
   → Calculate the heuristic for the current state.
   → If the heuristic is 0, the solution is found.

3) Generate near states.
   → For each queen, move it to all possible row in it's column and calculate heuristic for each state.

4) Choose the best state.
   → Select the neighbor state with the lowest heuristic.
   → If a better neighbor is not found, terminate.

5) Repeat until solution is found or there is no improvement.

If a solution is not found, restart with a new random configuration.

Proceed

Jussnote
Date
Page

A* pseudo code

```
function Astar(start):
    open_set = Priority Queue()
    open_set.enqueue(start, priority = heuristic(start))

    while open_set is not empty:
        cursent = open_set.dequeue()
        if heuristic(cursent) == 0:
            return cursent
        for neighbor in neighbors(cursent):
            cost = heuristic(neighbor)
            open_set.enqueue(neighbor, priority = cost)

function heuristic(state):
    return no. of conflicting queen pairs in state
```

Output

A* Solution: [7, 0, 6, 3, 1, -1, 4, 2]

Hill climbing Pseudo code

```
function Hill(start):
    current = start
    while true:
        next = best_neigh(current)
        if heuristic(next) >= heuristic(current):
            return current if heuristic(current)==0
            else failure
        current = next
function best_neigh(state):
    return neighbor of state with the lowest
    heuristic
function heuristic(state):
    return no-of conflicting queen pairs in
```

Output:

Hill climbing solution: [2, 4, 7, 3, 0, 6, 1, 5]

Hill climbing Solution: None.

29/10/24