Lab-05

Implementing Hill Climb Searching Algorithm for n queens problem.

function HILL-CLIMBING (problem) returns a state that is a local maximum
current ← MAKE-NODE (problem : INITIAL-STATE
loop do
neighbor ← a highest-valued successor of current
if neighbor. VALUE ≤ current. VALUE then
return current. STATE
current ← neighbor

Pseudocode

function hillClimb(n):
Step 1 : Initialize
step 2: board = random_board(n)
curr_conflicts = calc-conflicts(board)
loop:
step 2: Generate neighbors
best_neighbors = None
lowest-conflicts = curr.conflicts
for each column in board:
for each row in 0 to n-1:
if row != board[column]:
new_board = copy(board)
new_board [column] = row_conflicts
calc-conflicts (new_board)
if conflicts < lowest_conflicts:
best_neighbor = new_board
lowest_conflicts = conflicts
step 3: check if best neighbor is an improvement
if lowest-conflicts >= current_conflicts
return board

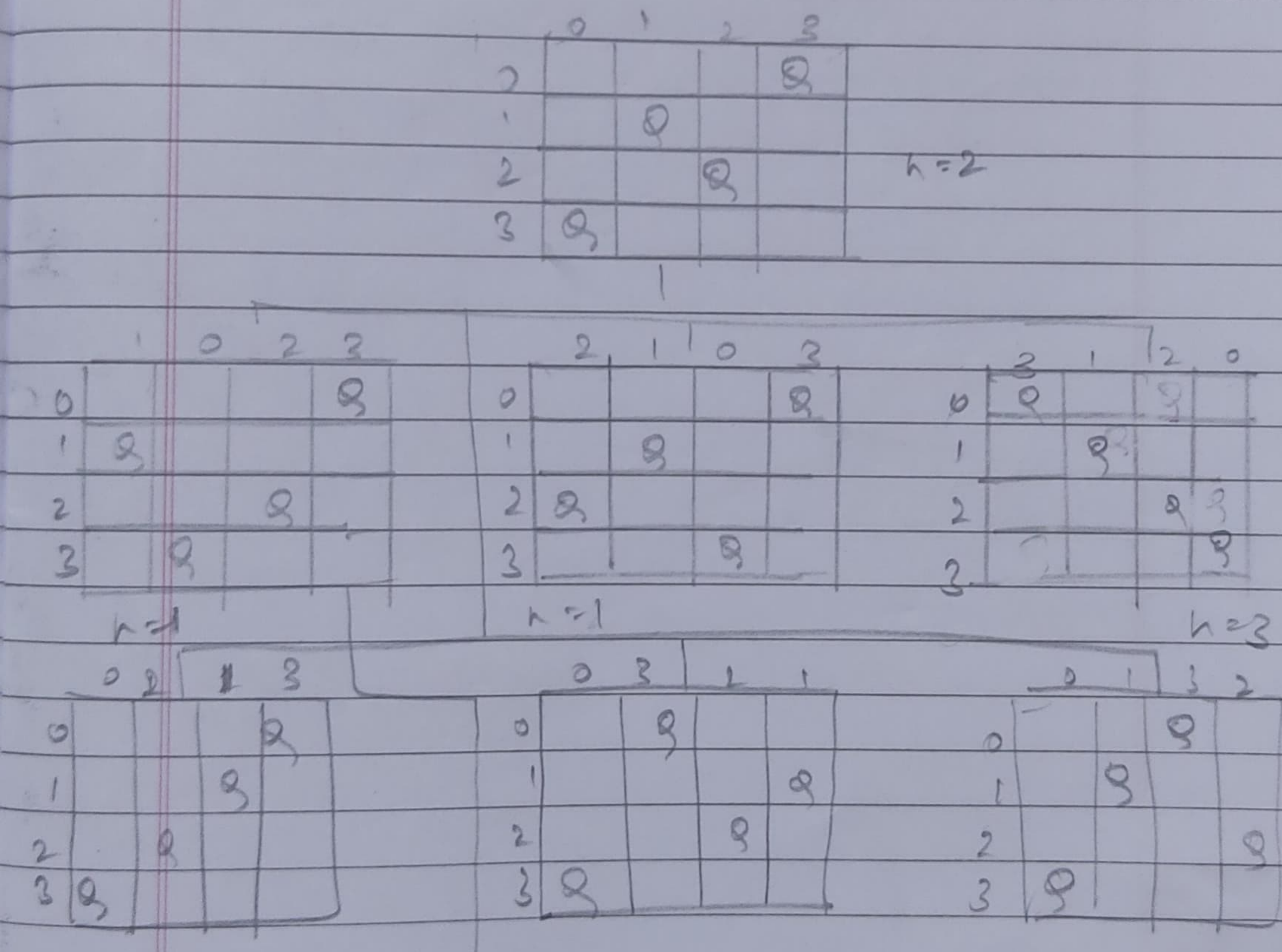
0
1
2
3

0
1
2
3

Step 4: Update to best neighbor

```
board = best-neighbor
current conflicts = lowest conflicts
function random board(n):
    board = array of size n
    for each column: board[column] = random row

function calculate_conflicts(board):
  conflicts = 0
  for each i and j:
    if board[i] == board[j] or
      abs(board[i] - board[j]) == abs(i-j)
      conflicts += 1
  return conflicts
```



	1	0	3	2
0			Q	
1	Q			
2				Q
3		Q		

final state