8/11/24

LAB-4

Hill-climbing search Algorithm for N-Queens:

function Hill-climbing (problem) returns a state that is a local minima

```
While (true)
    if calculate_heuristic(board) == 0
            return board
    for each row in board:
        for position in row:
            neighbor = makemove(board, row, position)

            heuristic = calculateheuristic (neighbour)
            if heuristic < lowest heuristic
                best_neighbour, lowest heuristic =
                                neighbour, heuristic
            if lowest heuristic >= calculate_heuristic
                                                (board)
                return "local minima reached", board
    board = bestneighbour
```

cost calculation:

			Q_0
	Q_1		
		Q_2	
Q_3			

Q_0	Q_1	Q_2	Q_3	
3	1	2	0	
1	3	2	0	Q_0 Q_1
2	1	3	0	Q_0 Q_2
0	1	2	3	Q_0 Q_3
3	0	2	1	Q_1 Q_3
3	2	1	0	Q_1 Q_2
3	1	0	2	Q_2 Q_3

	Q_0		
			Q_1
		Q_2	
Q_3			

Q_0 -
Q_1 Q_2
Q_2 Q_1
Q_3 -

cost = 1

		Q_0	
	Q_1		
			Q_2
Q_3			

Q_0 - Q_1
Q_1 - Q_0
Q_2 -
Q_3 -

cost = 1

Q_0			
	Q_1		
		Q_2	
			Q_3

Q_0 Q_1 Q_2 Q_3
Q_1 Q_0 Q_1 Q_3
Q_2 Q_0 Q_1 Q_3
Q_3 Q_0 Q_1 Q_2

cost = 3

			Q_0
Q_1			
		Q_2	
	Q_3		

Q_0 -
Q_1 -
Q_2 - Q_3
Q_3 - Q_2

cost = 1

		Q_2	
Q_0			
			Q_3
	Q_1		

cost : 0

Q_0	Q_1	Q_2	Q_3
1	3	2	0
3	1	2	0
2	3	1	0
0	3	2	1
1	2	3	0
1	0	3	2
1	3	0	2

OUTPUT:

0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0

```python
#Hill clibing search for N-Queens
import random

def get_user_board(n):
    board = []
    print(f"Enter the initial row positions for each column (0 to {n-1}):")
    for col in range(n):
        row = int(input(f"Column {col + 1}: "))
        if 0 <= row < n:
            board.append(row)
        else:
            print("Invalid input. Row must be between 0 and", n - 1)
            return None
    return board

def heuristic(board):
    n = len(board)
    attacks = 0
    for i in range(n):
        for j in range(i + 1, n):
            if board[i] == board[j] or abs(board[i] - board[j]) == j - i:
                attacks += 1
    return attacks

def get_neighbors(board):
    neighbors = []
    n = len(board)
    for col in range(n):
        for row in range(n):
            if board[col] != row:
                neighbor = board[:]
                neighbor[col] = row
                neighbors.append(neighbor)
    return neighbors
```

```python
def print_board(board):
    n = len(board)
    for row in range(n):
        line = ""
        for col in range(n):
            if board[col] == row:
                line += "Q "
            else:
                line += ". "
        print(line)
    print("\n")

def hill_climbing_with_restarts(n, initial_board):
    current = initial_board
    restarts = 0

    while True:
        while True:
            current_heuristic = heuristic(current)

            if current_heuristic == 0:
                return current  # Return the solution when no attacks are found

            neighbors = get_neighbors(current)
            best_neighbor = min(neighbors, key=heuristic)
            best_neighbor_heuristic = heuristic(best_neighbor)

            if best_neighbor_heuristic >= current_heuristic:
                break  # Local minimum reached, restart

            current = best_neighbor

        current = generate_board(n)  # Restart with a new random board
        restarts += 1

```

```
# Main execution
print("Tanish M V")
print("1BM22CS302")
print("Hill climbing search for N-Queens")
n = int(input("Enter the number of queens: "))
initial_board = get_user_board(n)

if initial_board:
    solution = hill_climbing_with_restarts(n, initial_board)
    print("Final Solution:")
    print_board(solution)
else:
    print("Invalid initial board configuration.")
```

```
Tanish M V
1BM22CS302
Hill climbing search for N-Queens
Enter the number of queens: 8
Enter the initial row positions for each column (0 to 7):
Column 1: 5
Column 2: 4
Column 3: 7
Column 4: 2
Column 5: 1
Column 6: 0
Column 7: 4
Column 8: 3
Final Solution:
. . . Q . . . .
. . . . . . Q .
Q . . . . . . .
. . . . . . . Q
. . . . Q . . .
. Q . . . . . .
. . . . . Q . .
. . Q . . . . .
```