# Name: Shantanu Rohile

# Roll No: 53

*# ==============================================*

# AI Practical 02-A

# Problem Statement:

# Write a program to solve the N-Queens Problem

# using the Hill Climbing algorithm as a heuristic-based local search technique # ==============================================

import random #

# Function to check if a board configuration is valid (no queens attack each other)

*#*

*def* ***is\_valid****(board):*

n = len(board) for i in range(n):

for j in range(i + 1, n):

if board[i] == board[j] or abs(board[i] - board[j]) == abs(i - j): return False

return True

*#*

# Heuristic function: number of conﬂicting pairs of queens #

*def* ***calculate\_heuristic****(board):*

n = len(board) conﬂicts = 0

for i in range(n):

for j in range(i + 1, n):

if board[i] == board[j] or abs(board[i] - board[j]) == abs(i - j): conﬂicts += 1

return conﬂicts

*#*

# Hill Climbing Algorithm for N-Queens Problem #

*def* ***hill\_climbing****(n):*

current\_board = [random.randint(0, n - 1) for \_ in range(n)] # Initial random board current\_heuristic = calculate\_heuristic(current\_board)

while current\_heuristic > 0:

found\_better = False

next\_board = list(current\_board)

# Try moving each queen to every other row in its column for i in range(n):

for j in range(n):

if current\_board[i] != j:

test\_board = list(current\_board) test\_board[i] = j

test\_heuristic = calculate\_heuristic(test\_board)

if test\_heuristic < current\_heuristic: next\_board = test\_board

current\_heuristic = test\_heuristic found\_better = True

if not found\_better:

break # No better neighbor found — local minimum current\_board = next\_board

return current\_board

*#*

# Example Usage #

n = 6

solution = hill\_climbing(n)

# Output the result

print("N-Queens Solution:", solution) print("Is solution valid?", is\_valid(solution))

*#*

# Sample Output:

# N-Queens Solution: [1, 3, 5, 0, 2, 4] # Is solution valid? True

*#*