A\*-8 queens

import heapq

def calculate\_heuristic(board):

attacking\_pairs = 0

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):

attacking\_pairs += 1

return attacking\_pairs  
#Count how many pairs of queens are attacking each other.

#Same column → board[i] == board[j]

#Same diagonal → abs(board[i] - board[j]) == abs(i - j) low heuristic==low attackers

def print\_board(board):

n = len(board)

# Print the current state of the board

for row in range(n):

board\_row = ['Q' if col == board[row] else '.' for col in range(n)]

print(' '.join(board\_row))

print()

def a\_star(start\_board):

n = 8

open\_list = []

heapq.heappush(open\_list, (0 + calculate\_heuristic(start\_board), 0, start\_board))

closed\_list = set()

step\_count = 0 # To track number of steps taken

while open\_list:

f, g, board = heapq.heappop(open\_list)

step\_count += 1

if calculate\_heuristic(board) == 0:

print("✅ Solution Found!")

print\_board(board)

print(f"Total steps taken: {step\_count}")

return board

# Open list: Priority queue (min-heap) of boards to explore, prioritized by f = g + h.Closed list: Keeps track of already visited board states.

for i in range(n):

for j in range(n):

if board[i] != j:

new\_board = board[:]

new\_board[i] = j # Moves the queen in ith row to a new column j.

if tuple(new\_board) not in closed\_list:

closed\_list.add(tuple(new\_board)) # Avoids redundant

print(f"Placing queen at row {i}, column {j} (step: {g + 1}):")

print\_board(new\_board)

heapq.heappush(open\_list, (g + 1 + calculate\_heuristic(new\_board), g + 1, new\_board))

print("❌ No solution found.")

return None

def get\_user\_input():

print("Please enter the initial state for the 8 queens (values between 0 and 7 for each row).")

print("For example, a valid input could be: [0, 1, 2, 3, 4, 5, 6, 7]")

while True:

try:

user\_input = input("Enter the initial state (a list of 8 integers from 0 to 7): ")

initial\_state = list(map(int, user\_input.strip('[]').split(',')))

if len(initial\_state) == 8 and all(0 <= x < 8 for x in initial\_state):

print(f"Initial state: {initial\_state}")

return initial\_state

else:

print("Invalid input. Please enter a list of 8 integers, each between 0 and 7.")

except ValueError:

print("Invalid input. Please enter a valid list of integers.")

# Get the initial state from the user

initial\_state = get\_user\_input()

# Run the A\* algorithm and display each step

a\_star(initial\_state)