Write the python program for Missionaries Cannibal problem

Program:

from collections import deque

# Check if state is valid

def is\_valid(m\_left, c\_left, boat, m\_total=3, c\_total=3):

m\_right = m\_total - m\_left

c\_right = c\_total - c\_left

# Missionaries and cannibals can't be negative

if m\_left < 0 or c\_left < 0 or m\_right < 0 or c\_right < 0:

return False

# Missionaries eaten condition

if (m\_left > 0 and m\_left < c\_left) or (m\_right > 0 and m\_right < c\_right):

return False

return True

# BFS to solve the problem

def missionaries\_cannibals():

# Each state: (missionaries\_left, cannibals\_left, boat\_position, path)

initial\_state = (3, 3, 'left', [])

visited = set()

queue = deque()

queue.append(initial\_state)

while queue:

m\_left, c\_left, boat, path = queue.popleft()

if (m\_left, c\_left, boat) in visited:

continue

visited.add((m\_left, c\_left, boat))

path = path + [(m\_left, c\_left, boat)]

# Goal state

if m\_left == 0 and c\_left == 0 and boat == 'right':

print("✅ Solution Found:")

for step in path:

print(f"Missionaries Left: {step[0]}, Cannibals Left: {step[1]}, Boat: {step[2]}")

return

# Possible moves: (missionaries, cannibals)

moves = [(2, 0), (0, 2), (1, 1), (1, 0), (0, 1)]

for m\_move, c\_move in moves:

if boat == 'left':

new\_m\_left = m\_left - m\_move

new\_c\_left = c\_left - c\_move

new\_boat = 'right'

else:

new\_m\_left = m\_left + m\_move

new\_c\_left = c\_left + c\_move

new\_boat = 'left'

if is\_valid(new\_m\_left, new\_c\_left, new\_boat):

queue.append((new\_m\_left, new\_c\_left, new\_boat, path))

print("❌ No solution found.")

# Run the program

missionaries\_cannibals()

