**AI FOR LLM- CSA1704**

**5. Missionaries and Cannibals problem**

**CODE:**

from collections import deque

# Define the initial and goal states

initial\_state = (3, 3, 1) # (Missionaries, Cannibals, Boat on left side)

goal\_state = (0, 0, 0)

# Valid boat moves: (Missionaries, Cannibals)

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

def is\_valid(state):

m\_left, c\_left, \_ = state

m\_right = 3 - m\_left

c\_right = 3 - c\_left

# Missionaries can't be outnumbered on either side

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

return False

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

return False

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

return False

return True

def get\_successors(state):

successors = []

m, c, boat = state

for move in moves:

dm, dc = move

if boat == 1: # Boat on left side

new\_state = (m - dm, c - dc, 0)

else: # Boat on right side

new\_state = (m + dm, c + dc, 1)

if is\_valid(new\_state):

successors.append(new\_state)

return successors

def bfs():

queue = deque()

queue.append((initial\_state, [initial\_state]))

visited = set()

visited.add(initial\_state)

while queue:

current\_state, path = queue.popleft()

if current\_state == goal\_state:

return path

for successor in get\_successors(current\_state):

if successor not in visited:

visited.add(successor)

queue.append((successor, path + [successor]))

return None

# Run the solver

solution = bfs()

# Display the result

if solution:

print("Steps to solve the Missionaries and Cannibals problem:\n")

for step in solution:

m, c, b = step

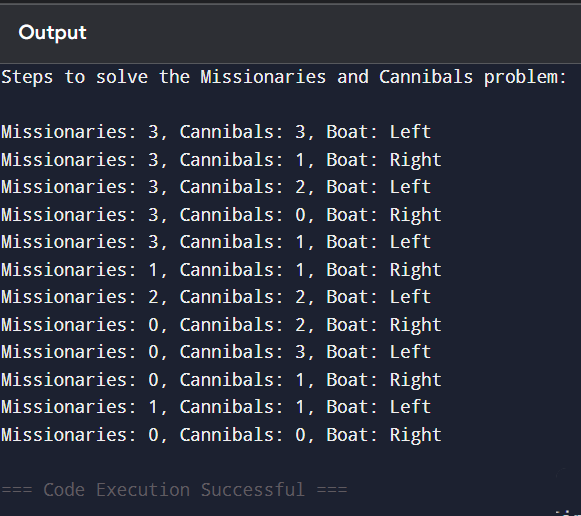
side = "Left" if b == 1 else "Right"

print(f"Missionaries: {m}, Cannibals: {c}, Boat: {side}")

else:

print("No solution found.")

**OUTPUT:**

****