5. Write the python program for Missionaries Cannibal problem

from collections import deque

# Define the goal state

goal\_state = (0, 0, 0) # (missionaries\_left, cannibals\_left, boat\_left)

# Check if a state is valid

def is\_valid(m, c):

return (m == 0 or m >= c) and (3 - m == 0 or 3 - m >= 3 - c)

# Generate possible next states

def get\_successors(state):

m, c, b = state

moves = [(1, 0), (0, 1), (2, 0), (0, 2), (1, 1)] # Possible moves: (missionaries, cannibals)

successors = []

for dm, dc in moves:

if b == 1: # Boat on the left side

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

else: # Boat on the right side

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

nm, nc, nb = new\_state

if 0 <= nm <= 3 and 0 <= nc <= 3 and is\_valid(nm, nc):

successors.append(new\_state)

return successors

# BFS to find the solution

def solve():

start\_state = (3, 3, 1) # (missionaries\_left, cannibals\_left, boat\_left)

queue = deque([(start\_state, [start\_state])])

visited = set()

while queue:

state, path = queue.popleft()

if state in visited:

continue

visited.add(state)

if state == goal\_state:

print("Solution found in", len(path) - 1, "moves:")

for step in path:

m, c, b = step

side = "Left" if b else "Right"

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

return

for succ in get\_successors(state):

if succ not in visited:

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

print("No solution found.")

# Run the solver

solve()

output :

