Write the python program for Missionaries Cannibal problem

AIM

To implement a Python program that solves the **Missionaries and Cannibals problem** using **Breadth-First Search (BFS)** to safely transfer all missionaries and cannibals across the river without violating constraints.

ALGORITHM

1. Represent each state as (M, C, B) where:
    a. M = missionaries on the left bank
    b. C = cannibals on the left bank
    c. B = boat position (1 = left, 0 = right)
2. Initialize the **start state** (3,3,1) and the **goal state** (0,0,0).
3. Use a queue to explore states in BFS order.
4. For each state (M,C,B):
    a. If it equals the goal, print the solution path and stop.
    b. Otherwise, generate all possible moves:
    (1,0), (2,0), (0,1), (0,2), (1,1) representing the number of missionaries and cannibals to move.
    c. Compute new state based on the boat position.
    d. Check **validity**: number of missionaries >= number of cannibals on each side (unless missionaries = 0).
    e. Enqueue valid new states not already visited.
5. Repeat until the goal state is reached.

8 PUZZLE AI.py - C:/Users/gayathri/Downloads/8 PUZZLE AI.py (3.8.2)

File Edit Format Run Options Window Help

```
from collections import deque
def is_valid(m,c):
    return (m==0 or m>=c) and (3-m==0 or 3-m>=3-c)
def mc_problem():
    start=(3,3,1)
    goal=(0,0,0)
    q=deque([(start,[])])
    visited=set()
    moves=[(1,0),(2,0),(0,1),(0,2),(1,1)]
    while q:
        (m,c,b),path=q.popleft()
        if (m,c,b)==goal:
            for step in path+[(m,c,b)]: print(step)
            return
        if (m,c,b) in visited: continue
        visited.add((m,c,b))
        for dm,dc in moves:
            if b==1:
                nm, nc, nb=m-dm, c-dc, 0
            else:
                nm, nc, nb=m+dm, c+dc, 1
            if 0<=nm<=3 and 0<=nc<=3 and is_valid(nm,nc):
                q.append(((nm,nc,nb),path+[(m,c,b)]))
mc_problem()
```

```
============================
(3, 3, 1)
(3, 1, 0)
(3, 2, 1)
(3, 0, 0)
(3, 1, 1)
(1, 1, 0)
(2, 2, 1)
(0, 2, 0)
(0, 3, 1)
(0, 1, 0)
(1, 1, 1)
(0, 0, 0)
>>>
```

RESULT

The program successfully solved the **Missionaries and Cannibals problem** using BFS.
It generated a sequence of safe moves transferring all 3 missionaries and 3 cannibals from the left bank to the right bank without any missionary being eaten.