**LAB-06**

The missionaries and cannibals problemusually stated as follows. Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one ortwo people. Find a way to get everyone to the other side without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place. This problem is famous inAI because it was the subject of the first paper that approached problem formulation from ananalytical viewpoint (Amarel, 1968).

a. Formulate the problem precisely, making only those distinctions necessary to ensure avalid solution. Draw a diagram of the complete state space.

a. **Problem Formulation**:

* **Initial State**: All three missionaries and three cannibals are on one side of the river, with the boat on the same side.
* **Goal State**: All three missionaries and three cannibals are on the opposite side of the river.
* **Actions**: Move one or two people (missionaries or cannibals) from one side of the river to the other.
* **Constraints**:
  1. At no point can there be more cannibals than missionaries on either side of the river.
  2. The boat has a maximum capacity of two people.
  3. All individuals must eventually reach the other side.

**State Space Diagram 1**:

Left Bank | Boat | Right Bank

(M=3, C=3) | (B) | (M=0, C=0)

| M | C | B | | M | C | B | | M | C | B |

----------------------------------------------- --------------------------------------------

| 3 | 3 | 1 | | 3 | 1 | 0 | | 0 | 2 | 1 |

------------------------------------------- --------------------------------------------

| 3 | 2 | 0 | | 3 | 0 | 1 | | 0 | 3 | 0 |

------------------------------------------- --------------------------------------------

| 3 | 1 | 1 | | 3 | 2 | 0 | | 0 | 1 | 1 |

------------------------------------------- --------------------------------------------

| 3 | 0 | 0 | | 3 | 1 | 0 | | 0 | 2 | 0 |

------------------------------------------- --------------------------------------------

| 2 | 0 | 1 | | 3 | 0 | 1 | | 1 | 1 | 0 |

------------------------------------------- --------------------------------------------

| 2 | 1 | 0 | | 1 | 0 | 0 | | 1 | 2 | 1 |

------------------------------------------- --------------------------------------------

| 0 | 1 | 1 | | 2 | 2 | 0 | | 3 | 0 | 1 |

------------------------------------------- --------------------------------------------

| 0 | 2 | 0 | | 0 | 2 | 1 | | 3 | 1 | 0 |

------------------------------------------- --------------------------------------------

| 1 | 1 | 1 | | 1 | 1 | 0 | | 2 | 2 | 1 |

------------------------------------------- --------------------------------------------

| 1 | 2 | 0 | | 0 | 0 | 1 | | 2 | 1 | 0 |

------------------------------------------- --------------------------------------------

| 0 | 0 | 1 | | 0 | 1 | 0 | | 3 | 3 | 1 |

------------------------------------------- --------------------------------------------

**State Space Diagram 2**:

Left Bank Right Bank

(M=3, C=3, B=1) (M=0, C=0, B=0)

| M | C | B | | M | C | B |

---------------------------------------------

| 3 | 3 | 1 | | 0 | 0 | 0 |

-------------------------------------------

| 3 | 2 | 0 | | 0 | 1 | 1 |

-------------------------------------------

| 3 | 1 | 0 | | 0 | 2 | 1 |

-------------------------------------------

| 3 | 1 | 1 | | 0 | 2 | 0 |

-------------------------------------------

| 3 | 0 | 0 | | 0 | 3 | 1 |

-------------------------------------------

| 2 | 2 | 1 | | 1 | 1 | 0 |

-------------------------------------------

| 2 | 1 | 0 | | 1 | 2 | 1 |

-------------------------------------------

| 2 | 0 | 1 | | 1 | 0 | 0 |

-------------------------------------------

| 1 | 1 | 0 | | 2 | 2 | 1 |

-------------------------------------------

| 1 | 0 | 1 | | 2 | 1 | 0 |

-------------------------------------------

| 0 | 2 | 1 | | 3 | 1 | 0 |

-------------------------------------------

| 0 | 1 | 0 | | 3 | 2 | 1 |

-------------------------------------------

| 0 | 0 | 1 | | 3 | 3 | 0 |

-------------------------------------------

| 0 | 0 | 0 | | 3 | 3 | 1 |

-------------------------------------------

These diagrams represent the valid states and transitions between the left and right banks of the river, considering the constraints of the problem

b. **Implementing the Problem using an Appropriate Search Algorithm**:

This problem can be solved using a breadth-first search (BFS) algorithm, as it can find the shortest path in an unweighted graph, which is our case. It's a good idea to check for repeated states to avoid loops and redundant paths.

Here's a Python-like pseudocode for BFS:

def bfs(initial\_state):

queue = [initial\_state]

visited = set()

while queue:

state = queue.pop(0)

visited.add(state)

if is\_goal(state):

return state

for neighbor in get\_neighbors(state):

if neighbor not in visited:

queue.append(neighbor)

In this pseudocode, initial\_state is the starting state, is\_goal checks if a state is the goal state, and get\_neighbors returns the neighboring states of a given state.

Code:

def bfs(initial\_state, is\_goal, get\_neighbors):

queue = [(initial\_state, [])] # Queue of states to explore, each state paired with its path

visited = set() # Set to keep track of visited states

while queue:

state, path = queue.pop(0) # Get the next state and its path from the queue

if state in visited:

continue # Skip this state if it has already been visited

visited.add(state) # Mark the state as visited

if is\_goal(state):

return path + [state] # Return the path to the goal state

neighbors = get\_neighbors(state) # Get neighboring states

if not neighbors:

continue # Skip if there are no valid moves from this state

# Add neighbors to the queue with their paths

for neighbor in neighbors:

queue.append((neighbor, path + [state]))

return None # If no goal state is found

def is\_goal(state):

return state == (0, 0, 0, 3, 3) # All missionaries and cannibals are on the other side

def get\_neighbors(state):

moves = []

m1, c1, b, m2, c2 = state

if b == 1: # Boat on initial side

for i in range(3):

for j in range(3):

if (m1 - i >= c1 - j >= 0 or m1 - i == 0) and (m2 + i >= c2 + j >= 0 or m2 + i == 0):

moves.append((m1 - i, c1 - j, 0, m2 + i, c2 + j))

else: # Boat on other side

for i in range(3):

for j in range(3):

if (m1 + i >= c1 + j >= 0 or m1 + i == 0) and (m2 - i >= c2 - j >= 0 or m2 - i == 0):

moves.append((m1 + i, c1 + j, 1, m2 - i, c2 - j))

return moves

# Define the initial state

initial\_state = (3, 3, 1, 0, 0)

# Call the bfs function

path\_to\_goal = bfs(initial\_state, is\_goal, get\_neighbors)

# Print the path to the goal state

print("Path to the goal state:")

for state in path\_to\_goal:

    print(state)

**(c) Why do you think people have a hard time solving this puzzle, given that the state space is so simple?**

Despite the simplicity of the state space, the Missionaries and Cannibals problem is challenging because it requires strategic planning and the ability to think ahead. The constraints of the problem mean that the solution is not straightforward and requires careful consideration of each move. Additionally, the problem is counter-intuitive in some ways (for example, sometimes it's necessary to move backwards to progress), which can lead to confusion. Finally, humans often struggle with problems that require considering many steps at once, as is required here.