# Missionaries and Cannibals Problem

## Python Code

from collections import deque  
  
def is\_valid\_state(m\_left, c\_left, m\_right, c\_right):  
 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) or (m\_right > 0 and m\_right < c\_right):  
 return False  
 return True  
  
def get\_successors(state):  
 m\_left, c\_left, boat\_pos = state  
 successors = []  
 moves = [(2, 0), (0, 2), (1, 1), (1, 0), (0, 1)]  
 for m, c in moves:  
 if boat\_pos == 0:  
 new\_state = (m\_left - m, c\_left - c, 1)  
 m\_right = 3 - new\_state[0]  
 c\_right = 3 - new\_state[1]  
 else:  
 new\_state = (m\_left + m, c\_left + c, 0)  
 m\_right = 3 - new\_state[0]  
 c\_right = 3 - new\_state[1]  
 if is\_valid\_state(new\_state[0], new\_state[1], m\_right, c\_right):  
 successors.append(new\_state)  
 return successors  
  
def solve\_missionaries\_cannibals():  
 start = (3, 3, 0)  
 goal = (0, 0, 1)  
 visited = set()  
 queue = deque([(start, [start])])  
 while queue:  
 state, path = queue.popleft()  
 if state == goal:  
 return path  
 if state in visited:  
 continue  
 visited.add(state)  
 for successor in get\_successors(state):  
 queue.append((successor, path + [successor]))  
 return None  
  
solution\_path = solve\_missionaries\_cannibals()  
for step in solution\_path:  
 print(step)

## Output (Solution Path)

(3, 3, 0)

(3, 1, 1)

(3, 2, 0)

(3, 0, 1)

(3, 1, 0)

(1, 1, 1)

(2, 2, 0)

(0, 2, 1)

(0, 3, 0)

(0, 1, 1)

(1, 1, 0)

(0, 0, 1)