A = 4 # Jug A capacity

B = 3 # Jug B capacity

goal = 2 # Goal is to get 2 liters

visited = set() # To keep track of visited states

# Recursive DFS function

def dfs(a, b, path):

# Base case: if goal is reached

if a == goal or b == goal:

path.append((a, b))

print("Solution found using DFS:")

for step in path:

print(f"Jug A: {step[0]}L, Jug B: {step[1]}L")

return True

# If already visited, skip

if (a, b) in visited:

return False

visited.add((a, b))

# Add current state to path

path.append((a, b))

# All possible operations

operations = [

(A, b), # Fill Jug A

(a, B), # Fill Jug B

(0, b), # Empty Jug A

(a, 0), # Empty Jug B

(min(a + b, A), max(0, b - (A - a))), # Pour B -> A

(max(0, a - (B - b)), min(a + b, B)) # Pour A -> B

]

# Try all operations recursively

for next\_a, next\_b in operations:

if dfs(next\_a, next\_b, path.copy()): # use path.copy() to keep separate paths

return True

return False

# Start from empty jugs

if not dfs(0, 0, []):

print("No solution found using DFS.")