from collections import deque

def is\_goal(a, b, target):

return a == target or b == target

def bfs(x, y, target):

visited = set()

queue = deque()

queue.append((0, 0, [])) # (jug A, jug B, path)

while queue:

a, b, path = queue.popleft()

if (a, b) in visited:

continue

visited.add((a, b))

path = path + [(a, b)]

if is\_goal(a, b, target):

return path

# Possible operations

next\_states = [

(x, b), # Fill A

(a, y), # Fill B

(0, b), # Empty A

(a, 0), # Empty B

(min(a + b, x), b - (min(a + b, x) - a)), # Pour B → A

(a - (min(a + b, y) - b), min(a + b, y)) # Pour A → B

]

for state in next\_states:

if state not in visited:

queue.append((state[0], state[1], path))

return None

# Example usage

x, y, target = 3, 5, 4

solution = bfs(x, y, target)

if solution:

for step in solution:

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

else:

print("No solution found.")

