**waterjugupdated**

from collections import deque

def Water\_Jug\_problem(X, Y, j1, j2, target):

queue = deque()

visited = set()

path = {}

actions = {}

queue.append((0, 0))

visited.add((0, 0))

path[(0, 0)] = None

actions[(0, 0)] = None

while queue:

current = queue.popleft()

current\_j1, current\_j2 = current

if current\_j1 == target or current\_j2 == target:

result\_path = []

while current:

result\_path.append(current)

current = path[current]

return result\_path[::-1], actions

next\_states = [

(X, current\_j2, "Fill Jug1"),

(current\_j1, Y, "Fill Jug2"),

(0, current\_j2, "Empty Jug1"),

(current\_j1, 0, "Empty Jug2"),

(current\_j1 - min(current\_j1, Y - current\_j2), current\_j2 + min(current\_j1, Y - current\_j2), "Pour Jug1 into Jug2"),

(current\_j1 + min(current\_j2, X - current\_j1), current\_j2 - min(current\_j2, X - current\_j1), "Pour Jug2 into Jug1")

]

for state in next\_states:

if state[:2] not in visited:

visited.add(state[:2])

queue.append(state[:2])

path[state[:2]] = current

actions[state[:2]] = state[2]

return None, actions

def positive(n):

while True:

try:

value = int(input(n))

if value <= 0:

raise ValueError("Please enter positive values only.")

return value

except ValueError as e:

print("Please enter positive values only.")

J1 = positive("Enter the volume of the first jug: ")

J2 = positive("Enter the volume of the second jug: ")

L = positive("Enter the target volume: ")

if J1 < L and J2 < L:

print("Not possible. Both jugs are smaller than the target volume.")

else:

print("Path is as follows:")

path, actions = Water\_Jug\_problem(J1, J2, J1, J2, L)

if path:

for state in path:

if state in actions and actions[state]:

print(f"Action: {actions[state]}")

print(f"Jug1: {state[0]}, Jug2: {state[1]}")

print("Solution found.")

else:

print("No solution found.")