In [5]:

```python
#KETHARNATH R 111723102089 CSE C EX.NO. 3

from collections import deque

def Solution(a, b, target):
    # Dictionary to store visited states
    visited = {}
    isSolvable = False
    path = []  # To store the path to the solution

    # Queue for BFS
    q = deque()

    # Initial state with both jugs empty
    q.append((0, 0))

    while q:
        # Get the current state
        u = q.popleft()

        # If already visited, continue
        if (u[0], u[1]) in visited:
            continue

        # If the state is out of bounds, continue
        if u[0] > a or u[1] > b or u[0] < 0 or u[1] < 0:
            continue

        # Store the path
        path.append([u[0], u[1]])
        visited[(u[0], u[1])] = True  # Mark as visited

        # If we have reached the target in either jug
        if u[0] == target or u[1] == target:
            isSolvable = True

            # If Jug1 contains the target, empty Jug2
            if u[0] == target and u[1] != 0:
                path.append([u[0], 0])

            # If Jug2 contains the target, empty Jug1
            elif u[1] == target and u[0] != 0:
                path.append([0, u[1]])

            # Print the solution path
            print("Path from initial state to solution state:")
            for step in path:
                print(f"({step[0]}, {step[1]})")
            return

        # Add all possible next states to the queue

        # Fill Jug1
        q.append((a, u[1]))

        # Fill Jug2
        q.append((u[0], b))
```

```
        # Empty Jug1
        q.append((0, u[1]))

        # Empty Jug2
        q.append((u[0], 0))

        # Pour water from Jug1 to Jug2
        transfer = min(u[0], b - u[1])
        q.append((u[0] - transfer, u[1] + transfer))

        # Pour water from Jug2 to Jug1
        transfer = min(u[1], a - u[0])
        q.append((u[0] + transfer, u[1] - transfer))

    if not isSolvable:
        print("Solution not possible")

# Driver code
if __name__ == '__main__':
    Jug1, Jug2, target = 4, 3, 2
    Solution(Jug1, Jug2, target)
```

```
Path from initial state to solution state:
(0, 0)
(4, 0)
(0, 3)
(4, 3)
(1, 3)
(3, 0)
(1, 0)
(3, 3)
(0, 1)
(4, 2)
(0, 2)
```

In [ ]: