Name: DEVASHISH TADASKA

Class: B

Batch: B2

Roll No: 21

AI LAB

Date:02/02/26

```
# CODE FOR WATER JUG
# BFS
```

In [10]:

```
from collections import deque

def water_jug():
    q = deque([(0, 0)])
    v = {(0, 0)}

    A, B, goal = 3, 5, 4

    while q:
        a, b = q.popleft()

        print(a, b)

        if a == goal or b == goal:
            print("Goal reached")
            return

        for x, y in [
            (A, b), (a, B), (0, b), (a, 0),
            (a - min(a, B - b), b + min(a, B - b
            (a + min(b, A - a), b - min(b, A - a

        ]:
            if (x, y) not in v:
                v.add((x, y))
                q.append((x, y))
water_jug()
```

```
0 0
3 0
0 5
3 5
0 3
3 2
3 3
0 2
1 5
2 0
1 0
2 5
0 1
3 4
Goal reached
```

CODE FOR WATER JUG

DFS

In [17]:

```
def water_jug_dfs(capacity1, capacity2, target):

    visited = set()
    path = []

    def dfs(jug1, jug2):

        if (jug1, jug2) in visited:
            return False

        visited.add((jug1, jug2))
        path.append((jug1, jug2))

        if jug1 == target or jug2 == target:
            return True

        # Fill Jug1
        if dfs(capacity1, jug2):
            return True

        # Fill Jug2
        if dfs(jug1, capacity2):
            return True

        # Empty Jug1
        if dfs(0, jug2):
            return True

        # Empty Jug2
        if dfs(jug1, 0):
            return True

        # Pour Jug1 → Jug2
        if dfs(max(0, jug1 - (capacity2 - jug2))
               min(capacity2, jug1 + jug2)):
            return True

        # Pour Jug2 → Jug1
        if dfs(min(capacity1, jug1 + jug2),
               max(0, jug2 - (capacity1 - jug1))
            return True

        # Backtrack
        path.pop()
        return False


    # Start from (0,0)
```

```
    if dfs(0, 0):
        return path
    else:
        return None

capacity1 = 3
capacity2 = 5
target = 4

solution = water_jug_dfs(capacity1, capacity2, t

if solution:
    print("Solution steps:")
    for step in solution:
        print(step)
else:
    print("No solution found.")
```

```
Solution steps:
(0, 0)
(3, 0)
(3, 5)
(0, 5)
(3, 2)
(0, 2)
(2, 0)
(2, 5)
(3, 4)
```

In [ ]: