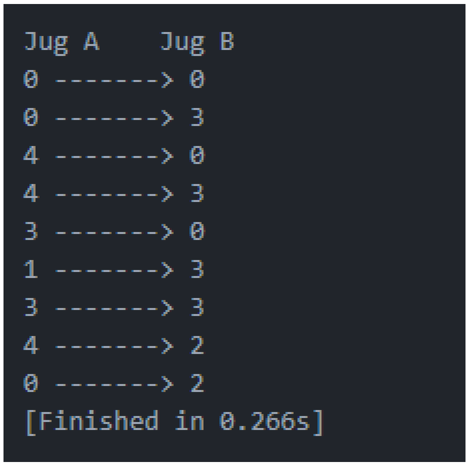
Ex. No: 1 Problem Solving- Using State Space Search

27/07/2022 Uninformed Search Strategies

1)Using BFS

Code:

```
from collections import deque
def pour_water_bfs(jug1, jug2, target):
    visited = {}
    pathAvailable = False
    ways = []
    queue = deque()
    queue.append((0, 0))
    while (len(queue) > 0):
        cur = queue.popleft()
        if ((cur[0], cur[1]) in visited):
            continue
        if ((cur[0] > jug1 or cur[1] > jug2 or cur[0] < 0 or cur[1] < 0)):
            continue
        ways.append([cur[0], cur[1]])
        visited[(cur[0], cur[1])] = 1
        if (cur[0] == target or cur[1] == target):
            pathAvailable = True
            if (cur[0] == target):
                if (cur[1] != 0):
                    ways.append([u[0], 0])
            else:
                if (cur[0] != 0):
                    ways.append([0, cur[1]])
            for i in ways:
                print( i[0], "------->",i[1] )
            break
        queue.append([cur[0], jug2])
        queue.append([jug1, cur[1]])
        for ap in range(max(jug1, jug2) + 1):
            c = cur[0] + ap
            d = cur[1] - ap
            if (c == jug1 or d==0 ):
                queue.append([c, d])
            c = cur[0] - ap
            d = cur[1] + ap
            if (c==0 or d == jug2):
                queue.append([c, d])
        queue.append([jug1, 0])
        queue.append([0, jug2])
    if (not pathAvailable):
        print("Cannot be measured")
Jug1, Jug2, target = 4,3,2
print("Jug A    Jug B")
pour_water_bfs(Jug1, Jug2, target)
```



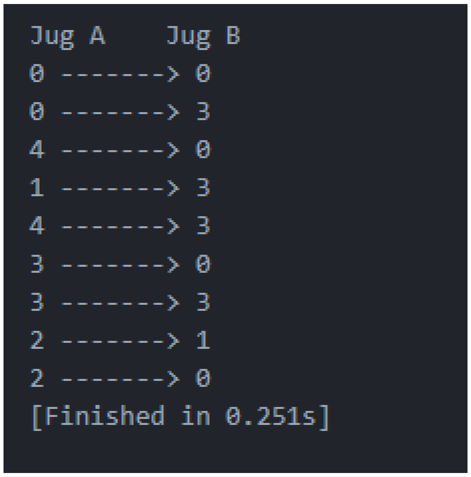
```
Jug A    Jug B
0 -------> 0
0 -------> 3
4 -------> 0
4 -------> 3
3 -------> 0
1 -------> 3
3 -------> 3
4 -------> 2
0 -------> 2
[Finished in 0.266s]
```

2)Using DSF
Code:

```
from collections import deque
def pour_water_dfs(a, b, target):
    visited = {}
    pathAvailable = False
    ways= []
    stack = []
    stack.append((0, 0))
    while (len(stack) > 0):
        cur = stack.pop()
        if ((cur[0], cur[1]) in visited):
            continue
        if ((cur[0] > a or cur[1] > b or cur[0] < 0 or cur[1] < 0)):
            continue
        ways.append([cur[0], cur[1]])
        visited[(cur[0], cur[1])] = 1
        if (cur[0] == target or cur[1] == target):
            pathAvailable = True
            if (cur[0] == target):
                if (cur[1] != 0):
                    ways.append([cur[0], 0])
            else:
                if (cur[0] != 0):
                    ways.append([0, cur[1]])
            for i in ways:
                print( i[0], "------->",i[1] )
            break
        stack.append([cur[0], b])
        stack.append([a, cur[1]])
        for a in range(max(a, b) + 1):
            c = cur[0] + a
            d = cur[1] - a
            if (c == a or (d == 0 and d >= 0)):
                stack.append([c, d])
            c = cur[0] - a
            d = cur[1] + a
            if ((c == 0 and c >= 0) or d == b):
                stack.append([c, d])
        stack.append([a, 0])
        stack.append([0, b])
    if (not pathAvailable):
        print("No solution")
```

```
Jug1, Jug2, target = 4,3,2
print("Jug A    Jug B")
pour_water_dfs(Jug1, Jug2, target)
```

Output:



```
Jug A    Jug B
0 -------> 0
0 -------> 3
4 -------> 0
1 -------> 3
4 -------> 3
3 -------> 0
3 -------> 3
2 -------> 1
2 -------> 0
[Finished in 0.251s]
```