**EXP NO: 01**

**DATE: 03.08.2022**

**WATER JUG PROBLEM**

**NAME: NAVEENKUMAR M**

**ROLLNO: 1905097**

**AIM:**

To solve the water jug problem using BFS and DFS.

**Problem:**There are two jugs of volume A litre and B litre. Neither has any measuring mark on it.There is a pump that can be used to fill the jugs with water. How can you get exactly x litre of water into the A litre jug. Assuming that we have unlimited supply of water.

Note:Let’s assume we have  A=4 litre and B= 3 litre jugs. And we want exactly 2 Litre water into jug A (i.e 4 litre jug) how we will do this.

**BFS**

from collections import deque

def BFS(a, b, target):

    pathMap = {}

    isSolvable = False

    path = []

    q = deque()

    q.append((0, 0))

    while (len(q) > 0):

        curr = q.popleft()

        if ((curr[0], curr[1]) in pathMap):

            continue

        if ((curr[0] > a or curr[1] > b or

                curr[0] < 0 or curr[1] < 0)):

            continue

        path.append([curr[0], curr[1]])

        pathMap[(curr[0], curr[1])] = 1

        if (curr[0] == target or curr[1] == target):

            isSolvable = True

            if (curr[0] == target):

                if (curr[1] != 0):

                    path.append([curr[0], 0])

            else:

                if (curr[0] != 0):

                    path.append([0, curr[1]])

            sz = len(path)

            for i in range(sz):

                print("(", path[i][0], ",",

                      path[i][1], ")")

            break

        q.append([curr[0], b])

        q.append([a, curr[1]])

        for ap in range(max(a, b) + 1):

            c = curr[0] + ap

            d = curr[1] - ap

            if (c == a or (d == 0 and d >= 0)):

                q.append([c, d])

            c = curr[0] - ap

            d = curr[1] + ap

            if ((c == 0 and c >= 0) or d == b):

                q.append([c, d])

        q.append([a, 0])

        q.append([0, b])

    if (not isSolvable):

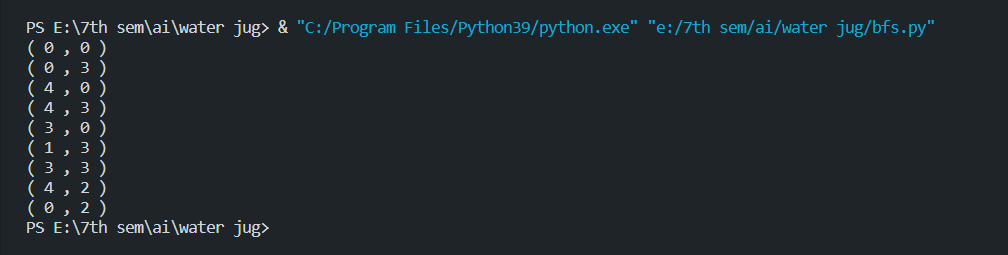
        print("No solution")

if \_\_name\_\_ == '\_\_main\_\_':

    Jug1, Jug2, target = 4, 3, 2

    BFS(Jug1, Jug2, target)

Output:



**DFS**

from collections import deque

def DFS(a, b, target):

    pathMap = {}

    isSolvable = False

    path = []

    stack = []

    stack.append((0, 0))

    while (len(stack) > 0):

        curr = stack.pop()

        if ((curr[0], curr[1]) in pathMap):

            continue

        if ((curr[0] > a or curr[1] > b or

                curr[0] < 0 or curr[1] < 0)):

            continue

        path.append([curr[0], curr[1]])

        pathMap[(curr[0], curr[1])] = 1

        if (curr[0] == target or curr[1] == target):

            isSolvable = True

            if (curr[0] == target):

                if (curr[1] != 0):

                    path.append([curr[0], 0])

            else:

                if (curr[0] != 0):

                    path.append([0, curr[1]])

            sz = len(path)

            for i in range(sz):

                print("(", path[i][0], ",",

                      path[i][1], ")")

            break

        stack.append([curr[0], b])

        stack.append([a, curr[1]])

        for ap in range(max(a, b) + 1):

            c = curr[0] + ap

            d = curr[1] - ap

            if (c == a or (d == 0 and d >= 0)):

                stack.append([c, d])

            c = curr[0] - ap

            d = curr[1] + ap

            if ((c == 0 and c >= 0) or d == b):

                stack.append([c, d])

        stack.append([a, 0])

        stack.append([0, b])

    if (not isSolvable):

        print("No solution")

if \_\_name\_\_ == '\_\_main\_\_':

    Jug1, Jug2, target = 4, 3, 2

    DFS(Jug1, Jug2, target)

Output:

