Exp. No: 3 DFS - Depth First Search [Water Jug]

Aim:

Create a DFS program to solve the Water Jug problem using python code

Algorithm:

1. Initialize the queue:

Step 1 - Create a queue 'q' for BFS
Step 2 - Create a set visited to keep track of visited states to avoid cycles
Step 3 - Enqueue the initial state (0,0) where the both jugs are empty.

2. BFS loop:

Step-4 - While queue is not empty

* Dequeue the front state (X, Y) where X is the amount of water in jug 1 and Y is the amount of water in Jug 2
* If either X == target or Y == target then solution is found
* If the state X, Y has been visited before skip to the next iteration
* Mark the state (X, Y) as visited
* For the current state (X, Y) generate all possible next states by applying

Fill jug 1 (jug1, x)
Fill jug 2 (jug2, x)
Empty jug 1 : (0, y)
Empty jug 2 : (x, 0)
pour water from jug1 to jug2 with capacity of jug2 :
pour water from jug2 to jug1 with capacity of jug1 :

3. Check for solution

step 5 - If the queue is exhausted and the target been reached. print "solution is not possible".

step 6 - otherwise print the sequence of operation leading to the solution.

Program:

```
From collection import deque
def solution (a, b, target)
    m = {}
    is solvable = False
    Path = []
    q = deque()
    q.append ((0,0))
    while len(q) > 0:
    u = q.popleft()
    if (u[0], u[1]) in m:
        continue
```

```
continue
path.append([u[0], u[i]])
m[[u[0], u[i]]] = 1

if u[0] == target or u[i] == target
    is solvable = true
    if u[0] == target:
        if u[i] != 0
    path.append([u[0], 0]).
    sl = len(path).
    for i in range(sl):
        print("(", path[i][0], ",", path[i][i])
    break
q.append([u[0], b])
q.append([u[i], a])
for ap in range(max(a,b)+1):
    c = u[0] + ap
    d = u[i] - ap
if c == a or (d == 0 and d >= 0):
    q.append([c, d])
q.append([a, 0])
q.append([0, b])
if not is solvable
print("solution not possible")
if __name__ == "__main__":
jug1 = int(input("Enter the capacity of jug 1:"))
```

```
jug2 = int(input("Enter the capacity of jug 2:"))
target = int(input("Enter the target amount:"))
print("path from initial state to solution state")
solution(jug1, jug2, target)
```

Output

```
Enter the capacity of jug 1 : 4
Enter the capacity of jug 2 : 3
Enter the target amount : 2
```

Path from initial state to solution

```
(0,0)   (1,3)
(0,3)   (3,3)
(4,0)   (4,2)
(4,3)   (0,2)
(3,0)
```

Result

Thus the jug program is executed and output is verified successfully