Exp. No: 2
Date:

Depth First Search

Water Jug

Aim:

To implement depth first search (DFS) for traverse a graph & explore all vertices by visiting as far along each branch as possible ~~before~~ ~~backtracking~~ in watejug.

Algorithm:

```
Step 01: Start
Step 02: Initialize an empty ~~stack and~~ a list to keep
         track of visited nodes.
Step 03: Push the starting node onto stack & mask
Step 04: While the stack is not empty, repeat 5 to 7
Step 05: Pop the top node from the stack
Step 06: Print or process the popped node.
Step 07: For each adjacent unvisited neighbour of the
         popped node.
Step 08: Stop
```

Program:

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

```
if u[0]>a or u[1]>b or u[0]<0 or u[1]<0
    continue
  path.append([u[0], u[1]])
  m[(u[0], u[1])]=1
  if u[0]==target or u[1]==target:
     is solvable = true
     if u[0]==target:
        if u[1]!=0:
           path.append([u[0],0])
     s1=len(path)
     for i in range(s1):
       print("(".path[i][0], ",",path[i][1]")")
     break
  q.append([u[0], b])
  q.append([u[1],a])
  for ap in range(max(a,b)+1):
      c=u[0]+ap
      d=u[1]-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 jug1"))
    jug2=int(input("Enter capacity of jug2: "))
    jug3=int(input("Enter the target amount"))
    print("path from initial state to
             solution state")
```

solution(jug1, jug2, target)

Output:

```
Enter the capacity of jug1 : 4
Enter the capacity of jug2: 3
Enter the target : 2
Path from initial state to solution state.
(0,0)       (1,3)
(0,3)       (3,3)
(4,0)       (4,2)
(4,3)       (0,2)
(3,0)
```

Result:

Thus the water jug program is executed and output is verified.