220701077
classmate
Date
Page

EXP NO : 3 - WATER JUG

AIM :

To implement DFS Algorithm for water Jug problem.

ALGORITHM :

1) start

2) Initiate the start with both Jugs empty.

3) Mark current state or visited

4) If both Jugs have equal amount of water then print the solution and terminate.

5) Generate all possible next states

6) Recursively apply DFS to each unvisited state.

7) stop.

CODE :

```
def waterJugDFS ( jug1-capacity , jug2-capacity , target ,
                      current-state = None , visited = None ):

        if visited is None :
              visited = set ()
        if current-state is None :
              current-state = (0,0)
        if current-state is visited :
              return False.

        visit.add ( current-state)
```

2207101077

```
Jug1, Jug2 = current_-state
printf( "Jug1 : { Jug1 }, Jug2: {Jug2}")
if Jug1== target or Jug2 == target:
    print (" solution found")
    return True

Possible- moves = [
        ( Jug1 - capacity , Jug2)
        ( Jug1 , Jug2- capacity
        ( 0 , Jug2)
        ( Jug1, 0)
        ( min ( Jug1-capacity, Jug1 + Jug2),
        max (0, Jug2- ( Jug1 - capacity - Jug1))),
       ( max (0, Jug1- ( Jug2- capacity - Jug2),
          min ( Jug2- capacity , Jug1 + Jug2)))
]
for next- state in possible - moves:

    if water JugDFS ( Jug1-capacity , Jug2- capacity,
                  target , next- state , visited ):

            return True


if --name == " main "
    Jug1 - capacity = 4

    Jug2 - capacity = 3

    target = 2

    print(" DFS Traversal")
```

```
if (not waterJugDFS (Jug1-capacity , Jug2-capacity,
                     target):

    print ("No solution is found").
```

OUTPUT :

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

RESULT :

Thus the program to implement DFS algorithm is successfully executed and output is verified.