Practical No. - 2

Name- Bhavesh Kewalramani

Roll No.- A-25

Section- A

Semester- 6th

Shift- 1st

Aim:

Write a program to implement Breadth First Search to solve Water Jug Problem.

Code:

```
import time


def gcd(a,b):
    if a == 0:
        return b
    return gcd(b%a, a)


jug1 = int(input("Enter the capacity of JUG-1       :"))
jug2 = int(input("Enter the capacity of JUG-2       :"))
goal = int(input("Enter the capacity to be achieved :"))


start=time.time()


path = []
queue = []
queue.append([0,0])
```

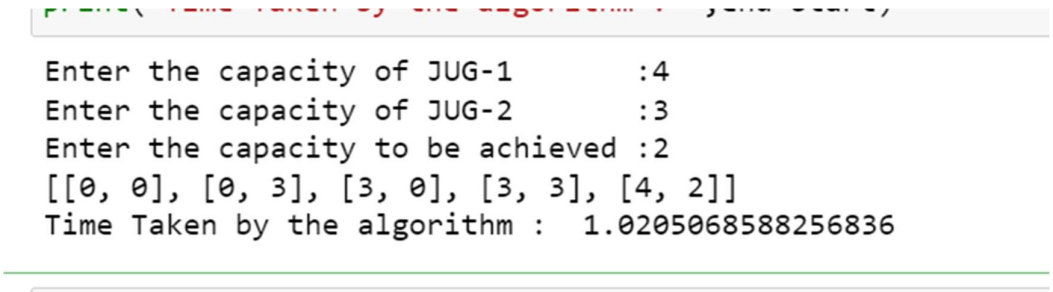
```
visited = []


time.sleep(1)
if goal % gcd(jug1,jug2) == 0:
    while(not (not queue)):
        state = queue.pop()
        x = state[0]
        y = state[1]
        path.append(state)
        if x == goal or y == goal:
            print(path)


        if x < jug1 and ([jug1, y] not in visited):
            queue.append([jug1, y])
            visited.append([jug1, y])


        if y < jug2 and ([x, jug2] not in visited):
            queue.append([x, jug2])
            visited.append([x, jug2])


        if x > jug1 and ([0, y] not in visited):
            queue.append([0, y])
            visited.append([0, y])


        if y > jug2 and ([jug1, 0] not in visited):
```

```
            queue.append([jug1, 0])
            visited.append([jug1, 0])


        if y > 0 and ([min(x + y, jug1), max(0, x + y - jug1)] not in visited):
            queue.append([min(x + y, jug1), max(0, x + y - jug1)])
            visited.append([min(x + y, jug1), max(0, x + y - jug1)])


        if x > 0  and ([max(0, x + y - jug2), min(x + y, jug2)] not in visited):
            queue.append([max(0, x + y - jug2), min(x + y, jug2)])
            visited.append([max(0, x + y - jug2), min(x + y, jug2)])
else:
    print("No solution found.")
end=time.time()
print("Time Taken by the algorithm : ",end-start)
```

Output:

1.

```
Enter the capacity of JUG-1       :4
Enter the capacity of JUG-2       :3
Enter the capacity to be achieved :2
[[0, 0], [0, 3], [3, 0], [3, 3], [4, 2]]
Time Taken by the algorithm :  1.0205068588256836
```



2.

```
Enter the capacity of JUG-1       :7
```

Enter the capacity of JUG-2 :9
Enter the capacity to be achieved :5
[[0, 0], [0, 9], [7, 2], [7, 9], [7, 0], [0, 7], [7, 7], [5, 9]]
Time Taken by the algorithm : 1.0100607872009277

```
print( Time Taken by the algorithm :  ,end-start)
```

```
Enter the capacity of JUG-1       :7
Enter the capacity of JUG-2       :9
Enter the capacity to be achieved :5
[[0, 0], [0, 9], [7, 2], [7, 9], [7, 0], [0, 7], [7, 7], [5, 9]]
Time Taken by the algorithm :  1.0100607872009277
```