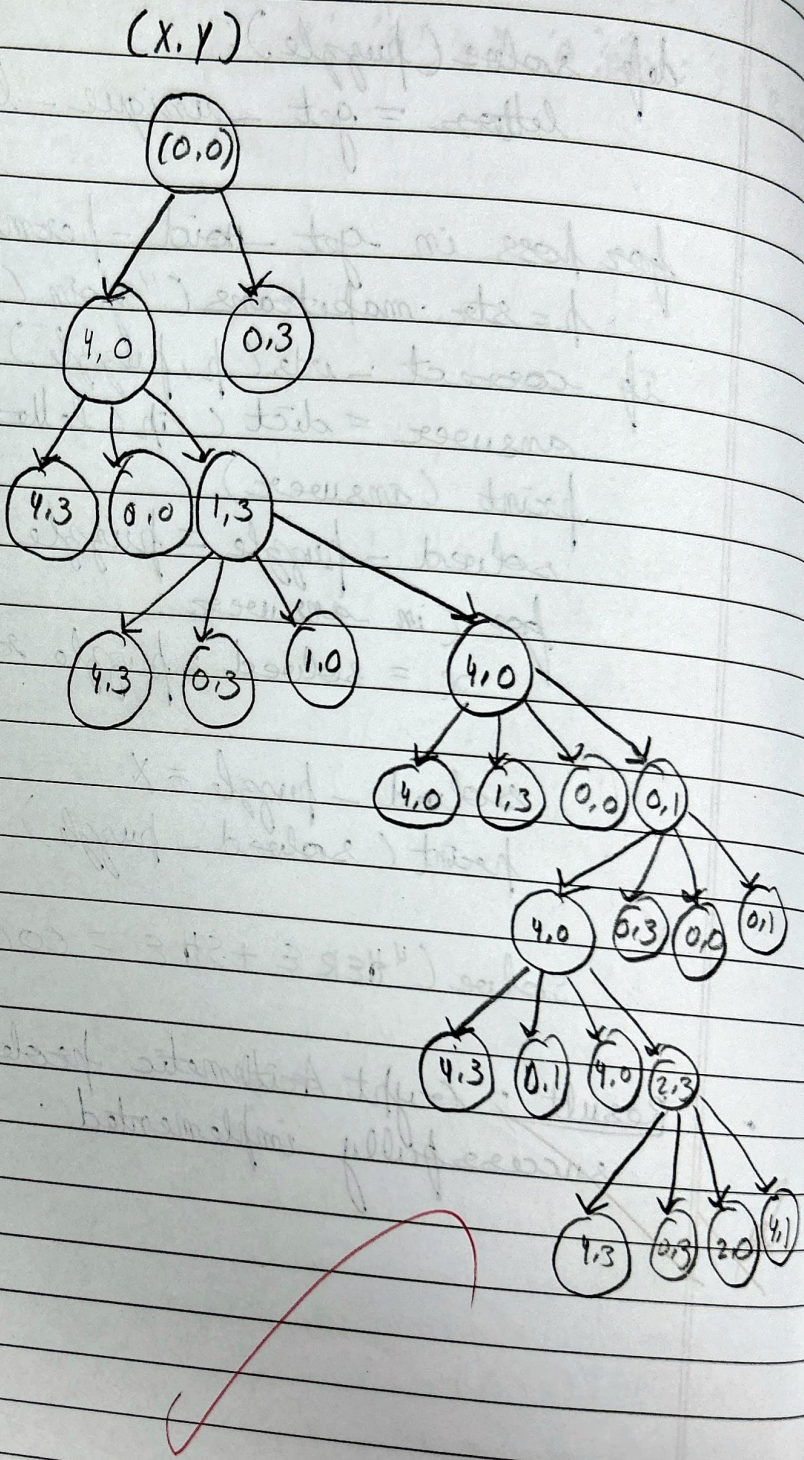
INDEX

NAME: Sulkit Shringi STD.: ____ SEC.: I I ROLL NO.: 596 SUB.: Artificial Intelligence

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1	22/1/24	Implementation of toy problem.	10	
2	31/1/24	Implementation of real world problem	10	31/01/24
3	~~14~~ 07/2/24	Implementation of constraint Satisfaction problem	10	7/2/24
4a	14/2/24	Breadth first search	10	
4b	14/2/24	Depth first search	10	21/2/24

State space tree

X = 4, Y = 3, Z = 2



14/2/24

Exc - 4

Breadth - First Search

- **Aim:**

To implement BFS and elaborate the search space tree for water jug problem.

- **Water jug:**

Jug 1 (x) = 5 litres
Jug 2 (y) = 3 litres
Target = 4 litres :: state (4,0)

- **Operations:**

1. Fill (y)
   current state (0,3)
2. Pour (y → x)
   current state (3,0)
3. Fill (y)
   current state : (3,3)
4. Pour (y → x)
   current state: (5,1)
5. Empty (y)
   current state (0,1)
6. Pour (y → x)
   current state : (1,0)
7. Fill (y)
   current state : (1,3)
8. Pour (y → x)
   current state : (4,0)
   Target reached.

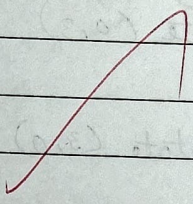
classmate
Date
Page

- Output:

Path from initial state to solution state ::

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

Observed.
Good
Well done.



code:

```
from collections import deque

def BFS(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
    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[0] == target):
                if (u[1] != 0):
                    path.append([u[0], 0])
        else
            if (u[0] != 0):
                path.append([0, u[1]])
```

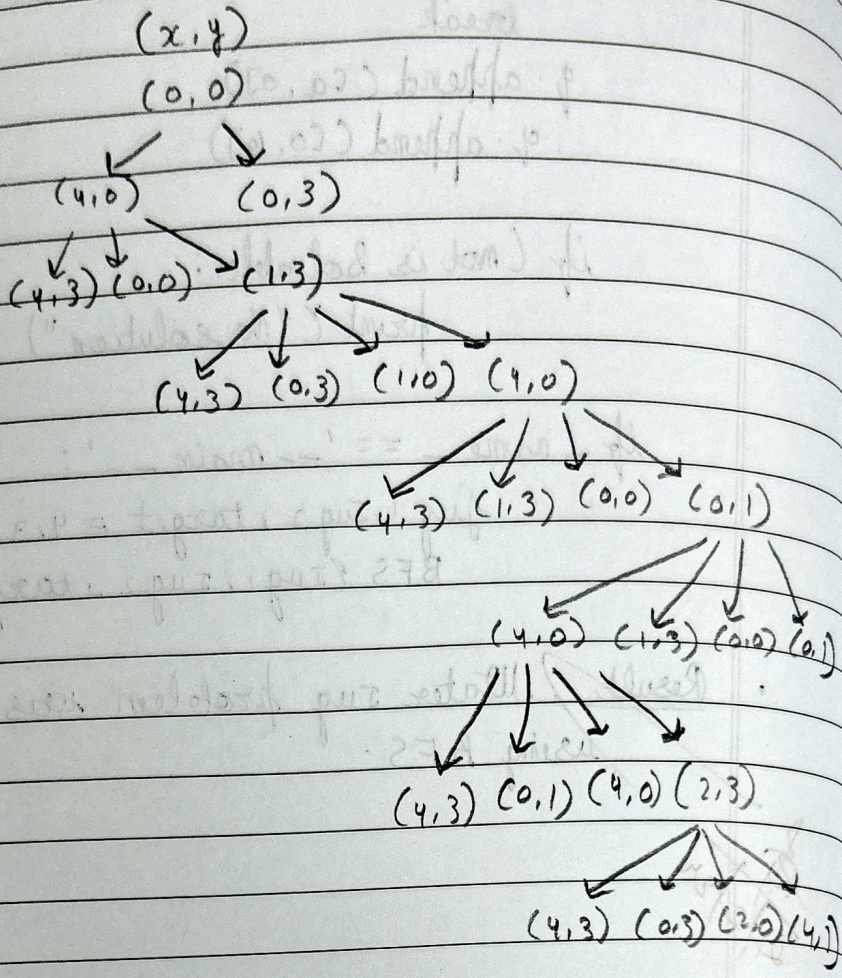
```
        sz = len(path)
        for i in range(sz):
            print("(", path[i][0]
            break
        q.append([a, 0])
        q.append([0, b])

    if (not isSolvable):
        print("No solution")

if __name__ == '__main__':
    jug1, jug2, target = 4, 3, 2
    BFS(jug1, jug2, target)
```

- **Result:** Water jug problem was executed using BFS.

State Space Tree : x=4 ; y=3 , z=2



14/2/24

Ex - 4b

Depth first search

- **Aim:**

To implement the DFS and elaborate the search space tree for water jug problem

- **Code:**

```
def print_path (mp, u):
    if u[0] == 0 and u[1] == 0:
        print ("0,0")
        return
    print_path (mp, mp[u])
    print (u[0], u[1])

def DFS (a, b, target):
    m = {}
    is_solvable = False
    mp = {}
    stack = [(0, 0)]

    while stack:
        u = stack.pop()
        if m.get (u, 0) == 1:
            continue:
        if u[0] == target or u[1] == target:
        is_solvable == True
            print_path (mp, u)
if u[0] != 0:
    print (u[0], 0)
else
```

Output :-

Jug1 = 4, Jug2 = 3, Target = 2

Path from initial state to solution state using DFS:

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

o/p verified.
Gopi
21/2/24

```
if u[0] != 0:
        print(0, u[i])
    return
    m[u] = 1
    if (u[0], 0) not in m:
      stack.append((u[0], 0))
      m[(u[0], 0)] = u
    if (0, u[i]) not in m:
      stack.append((0, u[i]))
      m[(0, u[i])] = u
    if (a, u[i]) not in m:
      stack.append((a, u[i]))
      m[(0, u[i])] = u
    x = min(u[0], b - u[1])
    if (u[0] - x, u[1] + x) not in m
      stack.append((a[0] - x, u[1] + x))
      m[(u[0] - x, u[1] + x)] = u
    if it is not is_solvable:
        print("No Solution")
Jug1, Jug2, target = 4, 3, 2;
    DFS(Jug1, Jug2, target);
```

Result: DFS was successfully implemented.