

**Sehrish Mubarik**

**BSAI-F23-001(4A)**

**Assignment no 01**

**Submitted to:**

**Sir Rasikh Ali**

**Lab task 01**

**Question no 01:**

**Water jug problem**

There are some rules for water jug problem

1. (x,y) is X<4 -> (4, Y) , Fill the 4-litre jug
2. (x, y) if Y<3 -> (x, 3) ,Fill the 3-litre jug
3. (x, y) if x>0 -> (x-d, d) , Pour some water from a 4-litre jug
4. (x, y) if Y>0 -> (d, y-d) , Pour some water from a 3-litre jug
5. (x, y) if x>0 -> (0, y) , Empty 4-litre jug on the ground
6. (x, y) if y>0 -> (x,0) , Empty 3-litre jug on the ground
7. (x, y) if X+Y >= 4 and y>0 -> (4, y-(4-x , Pour water from a 3-litre jug into a 4-litre jug until it is full
8. (x, y) if X+Y>=3 and x>0 -> (x-(3-y), 3)) , Pour water from a 3-litre jug into a 4-litre jug until it is full
9. (x, y) if X+Y <=4 and y>0 -> (x+y, 0), Pour all the water from a 3-litre jug into a 4-litre jug
10. (x, y) if X+Y<=3 and x>0 -> (0, x+) , Pour all the water from a 4-litre jug into a 3-litre jug
11. (0, 2) -> (2, 0), Pour 2-litre water from 3-litre jug into 4-litre jug
12. (2, Y) -> (0, y), Empty 2-litre in the 4-litre jug on the ground
13. **Code for BFS**

from collections import deque

def waterJugProblemBFS(capacity1, capacity2, goal):

queue = deque([(0, 0, [])])

visited = set()

while queue:

jug1, jug2, path = queue.popleft()

if (jug1, jug2) in visited:

continue

visited.add((jug1, jug2))

new\_path = path + [(jug1, jug2)]

if jug1 == goal or jug2 == goal:

print("Solution Found")

print("Steps:")

for action in new\_path:

print(action)

return True

rules = [

(capacity1, jug2),

(jug1, capacity2),

(0, jug2),

(jug1, 0),

(jug1 - min(jug1, capacity2 - jug2), jug2 + min(jug1, capacity2 - jug2)),

(jug1 + min(jug2, capacity1 - jug1), jug2 - min(jug2, capacity1 - jug1)),

]

for new\_state in rules:

if new\_state not in visited:

queue.append((\*new\_state, new\_path))

print("No Solution found")

return False

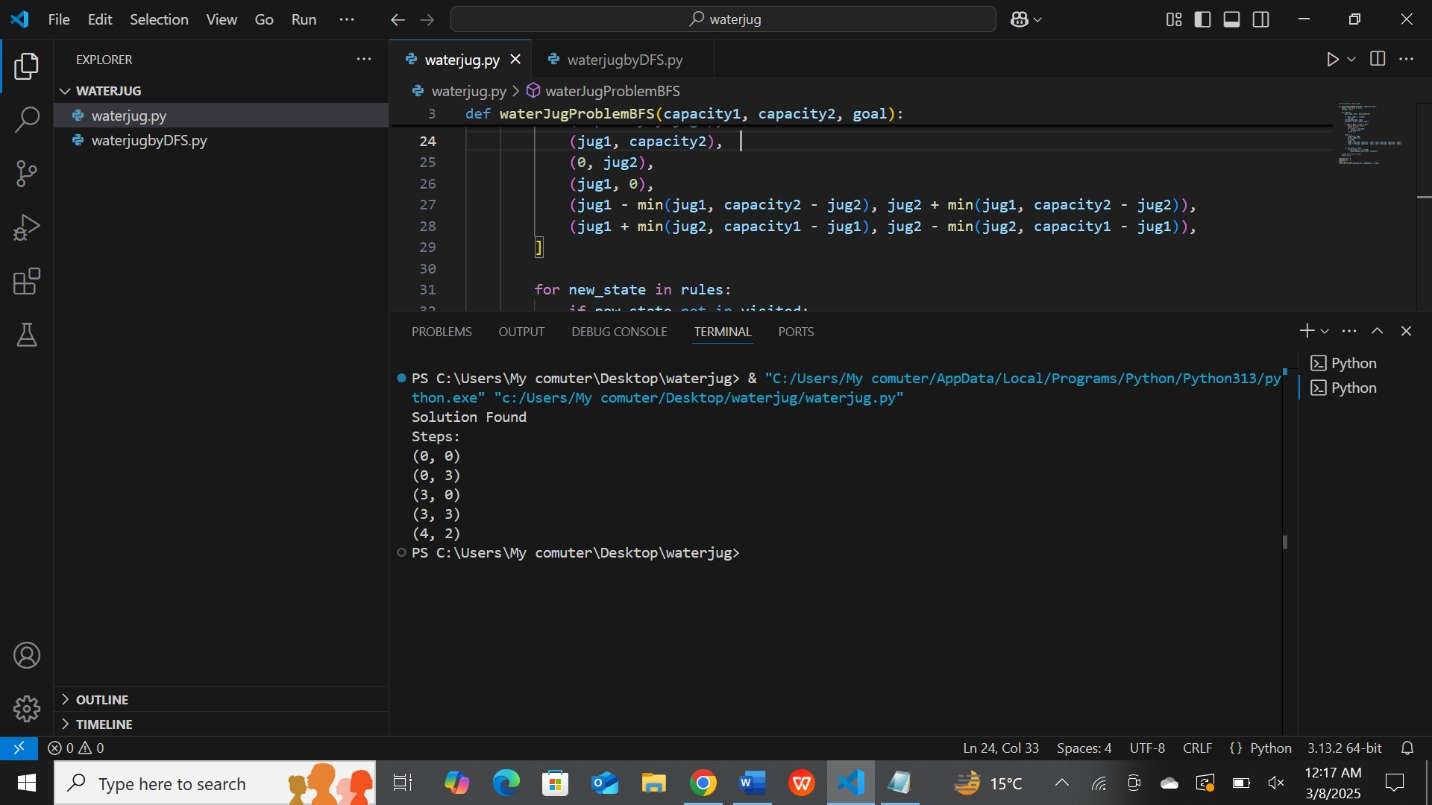
jug1Capacity = 4

jug2Capacity = 3

target = 2

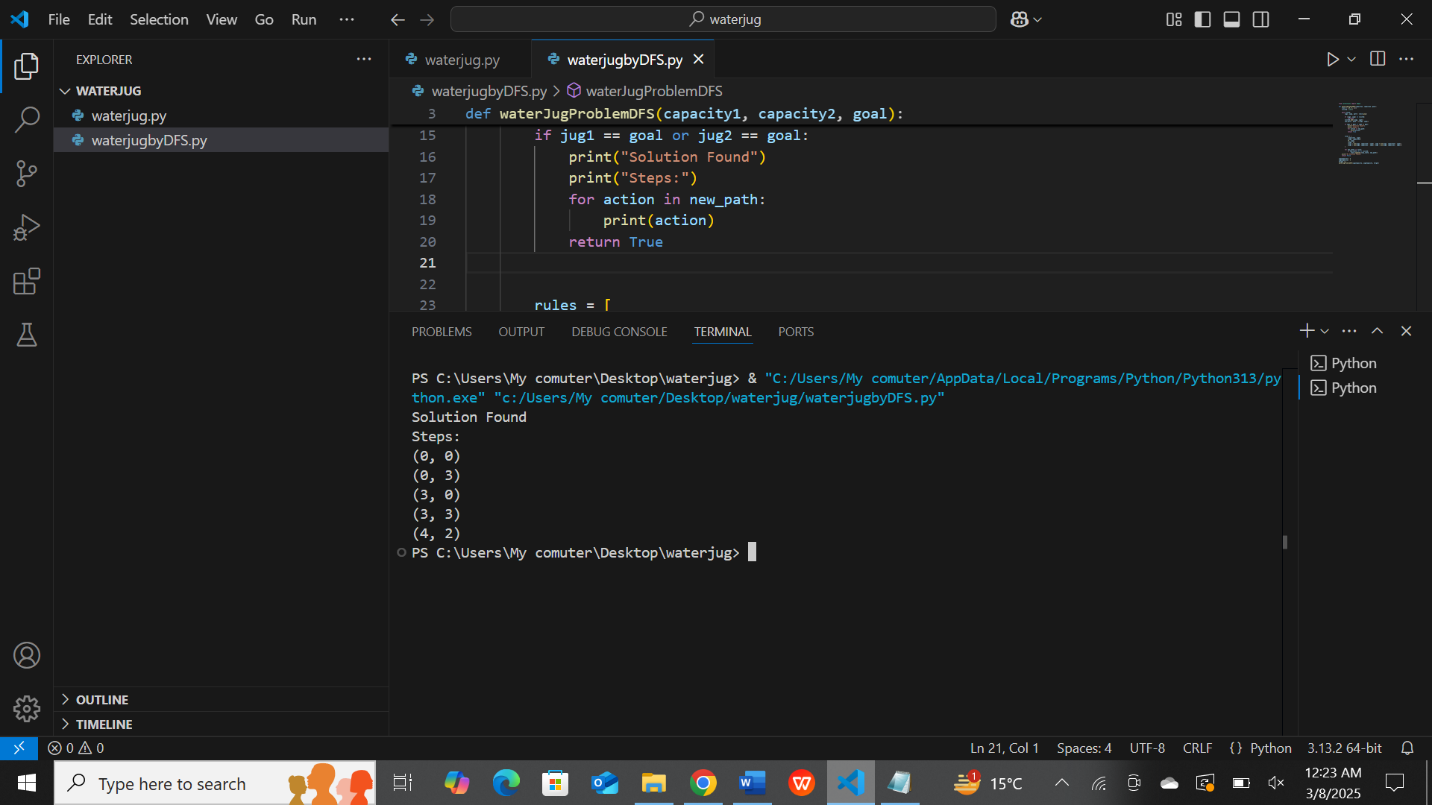
waterJugProblemBFS(jug1Capacity, jug2Capacity, target)

**output**



1. **DFS**
2. from collections import deque
3. def waterJugProblemDFS(capacity1, capacity2, goal):
4. stack = [(0, 0, [])]
5. visited = set()
6. while stack:
7. jug1, jug2, path = stack.pop()
9. if (jug1, jug2) in visited:
10. continue
11. visited.add((jug1, jug2))
12. new\_path = path + [(jug1, jug2)]
14. if jug1 == goal or jug2 == goal:
15. print("Solution Found")
16. print("Steps:")
17. for action in new\_path:
18. print(action)
19. return True
20. rules = [
21. (capacity1, jug2),
22. (jug1, capacity2),
23. (0, jug2),
24. (jug1, 0),
25. (jug1 - min(jug1, capacity2 - jug2), jug2 + min(jug1, capacity2 - jug2)),
26. (jug1 + min(jug2, capacity1 - jug1), jug2 - min(jug2, capacity1 - jug1)),
27. ]
28. for new\_state in rules:
29. if new\_state not in visited:
30. stack.append((\*new\_state, new\_path))
31. print("No Solution found")
32. return False
33. jug1Capacity = 4
34. jug2Capacity = 3
35. target = 2
36. waterJugProblemDFS(jug1Capacity, jug2Capacity, target)

**output**

****

.