DEPTH FIRST -WATER JUG PROBLEM

```
def water_jug_dfs(capacity_x, capacity_y, target):
def dfs(x, y, path):
if x == target or y == target:
path.append((x, y))
return True
if visited[x][y]:
return False
visited[x][y] = True
if x < capacity_x:
if dfs(capacity_x, y, path):
path.append((x, y))
return True
if y < capacity_y:
if dfs(x, capacity_y, path):
path.append((x, y))
return True
if x > 0:
if dfs(0, y, path):
path.append((x, y))
return True
if y > 0:
if dfs(x, 0, path):
path.append((x, y))
return True
if x > 0 and y < capacity_y:
pour = min(x, capacity_y - y)
if dfs(x - pour, y + pour, path):
path.append((x, y))
return True
if y > 0 and x < capacity_x:
pour = min(y, capacity_x - x)
if dfs(x + pour, y - pour, path):
path.append((x, y))
return True
return False

visited = [[False for _ in range(capacity_y + 1)] for _ in range(capacity_x + 1)]
path = []
if dfs(0, 0, path):
path.reverse()
return path
else:
return "No solution found."

capacity_x = 4
capacity_y = 3
target = 2
solution_path = water_jug_dfs(capacity_x, capacity_y, target)
if solution_path != "No solution found.":
for step, (x, y) in enumerate(solution_path):
print(f"Step {step}: Jug X: {x}, Jug Y: {y}")
else:
print("No solution found.")
```

OUTPUT:

```
Step 0: Jug X: 0, Jug Y: 0
Step 1: Jug X: 4, Jug Y: 0
Step 2: Jug X: 1, Jug Y: 3
Step 3: Jug X: 1, Jug Y: 0
Step 4: Jug X: 0, Jug Y: 1
Step 5: Jug X: 4, Jug Y: 1
Step 6: Jug X: 2, Jug Y: 3
```