

**The Superior University, Lahore**



**TASK (Fall 2023)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Course Title: | Programming for AI | | | | Course Code: | CAI601410 | Credit Hours: | 1 |
| Instructor: | Sir Rasikh | | | | Program Name: | BSDS | | |
| Semester: | 4th | Batch: | F23 | Section: | BSDSM-4A | Date: | Febrary 24 , 2024 | |
| Time Allowed: |  | | | | Maximum Marks: | |  | |
| Student’s Name: | Rafay Noor | | | | Reg. No. | 024 | | |
| **Task 1: Water Jug with DFS** | | | | | | | | |

jug1=int(input("Enter Capacity of Jug 1: "))

jug2=int(input("Enter Capacity of Jug 2: "))

goal=int(input('Enter Target: '))

print(f'Goal is {goal}')

def dfs(stack,visited):

    while stack:

        x,y=stack.pop()

        if (x,y) in visited:

            continue

        visited.add((x,y))

        print(f"Jug1: {x}, Jug2: {y}")

        if x==goal or y==goal:

            print("Goal reached!")

            return

        if x<jug1:

            stack.append((jug1,y))

        if y<jug2:

            stack.append((x,jug2))

        if x>0:

            stack.append((0,y))

        if y>0:

            stack.append((x,0))

        if x>0 and y<jug2:

            transfer=min(x, jug2 - y)

            stack.append((x-transfer,y+transfer))

        if y>0 and x<jug1:

            transfer=min(y, jug1 - x)

            stack.append((x+transfer,y-transfer))

    print("No solution found")

    return False

initial\_state=(0, 0)

stack=[initial\_state]

visited=set()

dfs(stack, visited)

This code is an implementation of the **Water Jug Problem** using a **Depth-First Search (DFS)** algorithm. The task is to find a sequence of steps that will allow you to measure a specific amount of water (the "goal") using two jugs with known capacities. Let’s break down the code and explain it step by step.

**1. Input and Goal Setup**

jug1=int(input("Enter Capacity of Jug 1: "))

jug2=int(input("Enter Capacity of Jug 2: "))

goal=int(input('Enter Target: '))

print(f'Goal is {goal}')

* The code first asks for the capacities of two jugs.
* It also asks for a target value which is the amount of water you need to measure using the two jugs.
* After the inputs, the goal is printed.

**2. DFS Function**

The main algorithm is contained in the dfs function, which implements a **Depth-First Search** on possible water states.

**Stack and Visited Set**

def dfs(stack,visited):

    while stack:

        x,y=stack.pop()

        if (x,y) in visited:

            continue

        visited.add((x,y))

        print(f"Jug1: {x}, Jug2: {y}")

* stack: This is the stack used for DFS. It starts with the initial state of the jugs, where both jugs are empty (0, 0).
* visited: This is a set that keeps track of visited states (pairs of water levels in jug1 and jug2) to avoid redundant calculations.
* In each iteration of the while loop:
  + The most recent state (x, y) (representing the amount of water in jug1 and jug2) is popped from the stack.
  + If the state has already been visited, it is skipped.
  + The state is added to the visited set, and the current state is printed.

**Goal Check**

if x==goal or y==goal:

            print("Goal reached!")

            return

If either of the jugs (jug1 or jug2) reaches the goal amount of water, a success message is printed, and the function returns, ending the DFS search.

**3. Generating New States**

If the goal hasn't been reached, new possible states are generated by performing the following actions:

        if x<jug1:

            stack.append((jug1,y))

        if y<jug2:

            stack.append((x,jug2))

        if x>0:

            stack.append((0,y))

        if y>0:

            stack.append((x,0))

These actions represent the following:

* **Filling the jugs**: If a jug is not full, it is filled to its capacity.
* **Emptying the jugs**: If a jug has water, it can be emptied.

**Transfer Between Jugs**

if x>0 and y<jug2:

            transfer=min(x, jug2 - y)

            stack.append((x-transfer,y+transfer))

        if y>0 and x<jug1:

            transfer=min(y, jug1 - x)

            stack.append((x+transfer,y-transfer))

* **Transfer between jugs**: If one jug has water and the other has space, the water can be transferred from one jug to the other. This is done in two possible directions:
  + From jug1 to jug2.
  + From jug2 to jug1.

These transitions generate new states based on the current amounts of water in the jugs.

**4. No Solution Case and Execution**

    print("No solution found")

    return False

initial\_state=(0, 0)

stack=[initial\_state]

visited=set()

dfs(stack, visited)