CSE 3002: Artificial Intelligence

Lab Assignment 3

1. Given two water jugs with capacities X and Y litres. Initially, both the jugs are empty. Alsogiven that there is an infinite amount of water available. The jugs do not have markings to measure. The task is to determine whether it is possible to measure Z litres of water using both the jugs. And if true, print any of the possible ways.

i. Solve for X = 4, Y = 3, Z = 2.

Input:

```
#21BCE8421 Harshita Pasupuleti
from collections import defaultdict
# x and y are capacities of two water jugs.
# z is the amount of water to be measured using both jugs.
x, y, z = 4, 3, 2
visited = defaultdict(lambda: False)
#L1 is the amount of water present in 1st jug at a certain time.
#L2 is the amount of water present in 2nd jug at a certain time.
def wtrJSolv(L1, L2):
    if (L1 == z and L2 == 0) or (L2 == z and L1 == 0):
        print(L1, L2)
        return True
    if visited[(L1, L2)] == False:
        print(L1, L2)
        visited[(L1, L2)] = True
        return (wtrJSolv(0, L2) or
                wtrJSolv(L1, 0) or
                wtrJSolv(x, L2) or
                wtrJSolv(L1, y) or
                wtrJSolv(L1 + min(L2, (x-L1)), L2 - min(L2, (x-L1))) or
                wtrJSolv(L1 - min(L1, (y-L2)), L2 + min(L1, (y-L2))))
    else:
        return False

print("Process: ")
wtrJSolv(0,0)
```

Output:

```
Process:
0 0
4 0
4 3
0 3
3 0
3 3
4 2
0 2
```

ii.Solve for X = 3, Y = 5, Z = 4

Input:

```
#21BCE8421 Harshita Pasupuleti
from collections import defaultdict
# x and y are capacities of two water jugs.
# z is the amount of water to be measured using both jugs.
x, y, z =3, 5, 4
visited = defaultdict(lambda: False)
#L1 is the amount of water present in 1st jug at a certain time.
#L2 is the amount of water present in 2nd jug at a certain time.
def wtrJSolv(L1, L2):
    if (L1 == z and L2 == 0) or (L2 == z and L1 == 0):
        print(L1, L2)
        return True
    if visited[(L1, L2)] == False:
        print(L1, L2)
        visited[(L1, L2)] = True
        return (wtrJSolv(0, L2) or
                wtrJSolv(L1, 0) or
                wtrJSolv(x, L2) or
                wtrJSolv(L1, y) or
                wtrJSolv(L1 + min(L2, (x-L1)), L2 - min(L2, (x-L1))) or
                wtrJSolv(L1 - min(L1, (y-L2)), L2 + min(L1, (y-L2))))
    else:
        return False
print("Steps: ")
wtrJSolv(0,0)
```

Output:

```
Steps:
0 0
3 0
3 5
0 5
3 2
0 2
2 0
2 5
3 4
0 4
```

iii. Solve for X = 5, Y = 11, Z = 4.

Input:

```
#21BCE8421 Harshita Pasupuleti
from collections import defaultdict
# x and y are capacities of two water jugs.
# z is the amount of water to be measured using both jugs.
x, y, z = 5, 11, 4
visited = defaultdict(lambda: False)
#L1 is the amount of water present in 1st jug at a certain time.
#L2 is the amount of water present in 2nd jug at a certain time.
def wtrJSolv(L1, L2):
    if (L1 == z and L2 == 0) or (L2 == z and L1 == 0):
        print(L1, L2)
        return True
    if visited[(L1, L2)] == False:
        print(L1, L2)
        visited[(L1, L2)] = True
        return (wtrJSolv(0, L2) or
                wtrJSolv(L1, 0) or
                wtrJSolv(x, L2) or
                wtrJSolv(L1, y) or
                wtrJSolv(L1 + min(L2, (x-L1)), L2 - min(L2, (x-L1))) or
                wtrJSolv(L1 - min(L1, (y-L2)), L2 + min(L1, (y-L2))))
    else:
        return False
print("Steps: ")
wtrJSolv(0,0)
```

Output:

```
Steps:
0 0
5 0
5 11
0 11
5 6
0 6
5 1
0 1
1 0
1 11
5 7
0 7
5 2
0 2
2 0
2 11
5 8
0 8
5 3
0 3
3 0
3 11
5 9
0 9
5 4
0 4
```

2. You have an 8 litre jug full of water and two smaller jugs, one that contains 5 litres and the other 3 litres. None of the jugs have markings on them, nor do you have any additional measuring device. You have to divide the 8 litres of water equally between your two best friends, so that each gets 4 litres of water. How can you do this?

Input:

```
#21BCE8421 Harshita Pasupuleti
capacity = (8,5,3)
x = capacity[0]
y = capacity[1]
z = capacity[2]
memory = {}
ans = []
def get_all_states(state):
    # Let the 3 jugs be called a,b,c
    a = state[0]
    b = state[1]
    c = state[2]
    if(a==4 and b==4):
        ans.append(state)
        return True
        if((a,b,c) in memory):
            return False
            memory[(a,b,c)] = 1
            #empty jug a
            if(a>0):
                #empty a into b
                if(a+b<=y):
                    if(get_all_states((0,a+b,c)) ):
                        ans.append(state)
                        return True
                    else:
                        if( get_all_states((a-(y-b), y, c)) ):
                            ans.append(state)
                            return True
                            #empty a into c
                            if(a+c<=z):
                                if( get_all_states((0,b,a+c)) ):
                                    ans.append(state)
                                    return True
                                else:
                                    if(get_all_states((a-(z-c), b, z)) ):
                                        ans.append(state)
                                        return True
                                        #empty jug b
                                        if(b>0):
                                            #empty b into a
                                            if(a+b<=x):
                                                if( get_all_states((a+b, 0, c)) ):
                                                    ans.append(state)
                                                    return True
                                                else:
                                                    if( get_all_states((x, b-(x-a), c)) ):
                                                        ans.append(state)
                                                        return True
                                                        #empty b into c
                                                        if(b+c<=z):
                                                            if( get_all_states((a, 0, b+c)) ):
                                                                ans.append(state)
                                                                return True
                                                            else:
                                                                if( get_all_states((a, b-(z-c), z)) ):
                                                                    ans.append(state)
                                                                    return True
                                                                    #empty jug c
                                                                    if(c>0):
                                                                        #empty c into a
                                                                        if(a+c<=x):
                                                                            if( get_all_states((a+c, b, 0)) ):
                                                                                ans.append(state)
                                                                                return True
                                                                            else:
                                                                                if( get_all_states((x, b, c-(x-a))) ):
                                                                                    ans.append(state)
                                                                                    return True
                                                                                    #empty c into b
                                                                                    if(b+c<=y):
                                                                                        if( get_all_states((a, b+c, 0)) ):
                                                                                            ans.append(state)
                                                                                            return True
                                                                                        else:
                                                                                            if( get_all_states((a, y, c-(y-b))) ):
                                                                                                ans.append(state)
                                                                                                return True
                                                                                                return False
                                                                                                initial_state = (8,0,0)
                                                                                                print("Steps : \n")
                                                                                                get_all_states(initial_state)
                                                                                                ans.reverse()
                                                                                                print("Starting work")
                                                                                                for i in ans:
                                                                                                    print(i)
```

Output:

```
Starting work
(8,0,0)
(3,5,0)
(0,5,3)
(5,0,3)
(5,3,0)
(2,3,3)
(2,5,1)
(7,0,1)
(7,1,0)
(4,1,3)
(4,4,0)
```

Name: Harshita Pasupuleti

Registration Number: 21BCE8421