# Artificial Intelligence C.A3


**COURSE CODE-INT-404**

Project – ***WATERJUG PROBLEM***

**CANDIDATES NAME**

| Sr. No. | Registration No | Name of Student | Roll No |
|---|---|---|---|
| 1 | 11608536 | Kiran Bala | K18UW-20 |
| 2 | 11801170 | Syed Mohd Rayyan | K18UW-60 |
| 3 | 11600763 | Mariya Eva Joseph | K18UW-17 |

**SUBMITTED BY- KIRAN BALA (11608536)(20)**

**SUBMITTED TO - ANKITA WADHWAN**

**GITHUB LINK-**
**https://github.com/Syedrayyan/Rey/blob/master/AI%20PROJECT**

# Artificial Intelligence C.A3

## TABLE OF CONTENTS

**CONTENTS**

# Artificial Intelligence C.A3


## Abstract:

In the world of finance, Problems solving is one of the most important activities. Professional traders have developed a variety of analysis methods such as fundamental analysis, technical analysis, quantitative analysis, and so on. Such analytically methods make use of different sources ranging from news to price data, but they all aim at predicting the company'sfuture stock prices so they can make educated decisions on their trading.And calculations while solving problems.

In recent years, the increasing prominence of machine learning in various industries have enlightened ;many fields to apply machine learning techniques to the field, and some of them have produced quite promising results for forecast and solving problems . In this paper, we will focus on short-term price prediction and solving problems with the help of algorithms such as water jug problem algorithm in AI.

# Artificial Intelligence C.A3

## INTRODUCTION

A Water Jug Problem: You are given three jugs, user gives the goal state and capacity, a pump which has unlimited water which you can use to fill the jug, and the ground on which water may be poured. Neither jug has any measuring markings on it.

The System uses built in artificial intelligence to answer the query.

## PURPOSE

The purpose of this project is to make clearify the water jug methodt, which is smart enough to give answer on the basis of user input. This Water Jug problem uses series of algorithm which predict the answer on the basis of current circumstances.

.Water pouring puzzles (also called water jug problems, decanting problems[1] or measuring puzzles) are a class of puzzle involving a finite collection of water jugs of known integer capacities (in terms of a liquid measure such as liters or gallons). Initially each jug contains a known integer volume of liquid, not necessarily equal to its capacity. Puzzles of this type ask how many steps of pouring water from one jug to another (until either one jug becomes empty or the other becomes full) are needed to reach a goal state, specified in terms of the volume of liquid that must be present in some jug or jugs.

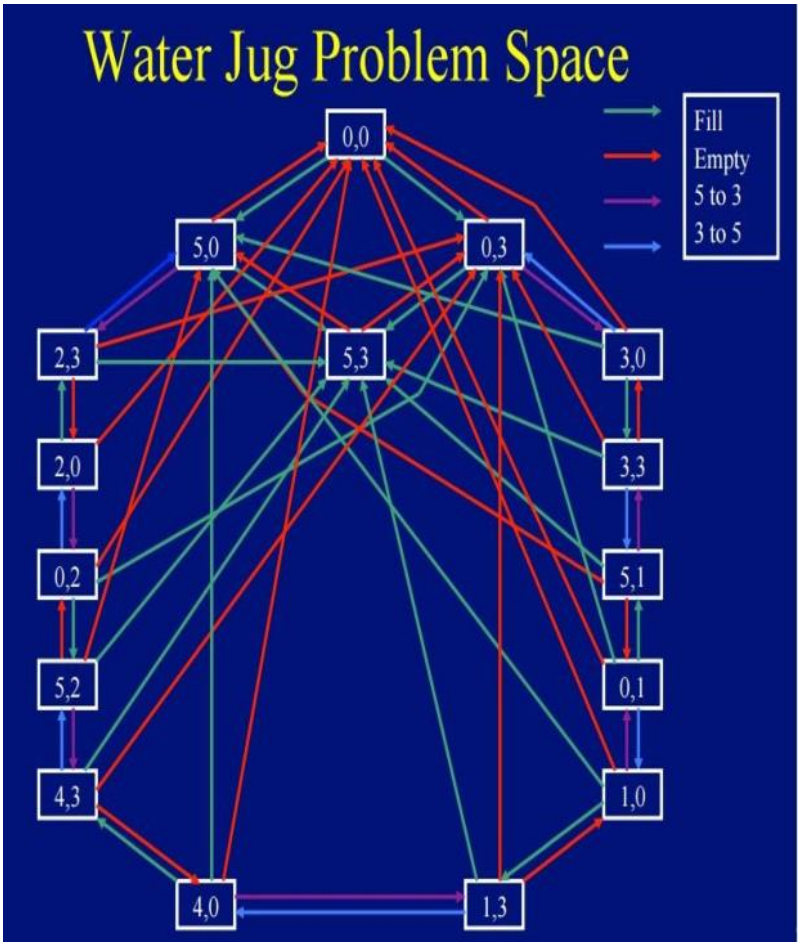
# Artificial Intelligence C.A3

## ADVANTAGES

- ✓ This application saves time for the user.
- ✓ Better engagement.
- ✓ Personalized Experience.
- ✓ Gather and analyze customer data.
- ✓ Make solving problems easier
- ✓ Lead nurturing.
- ✓ Quick response time.

## PROBLEM SPACE OF WATER JUG

# Artificial Intelligence C.A3




NOTE: THIS IS FOR WATER JUG USING TWO JUGS.

# Artificial Intelligence C.A3

## Literature review


- Cowley, Elizabeth B. (1926). "Note on a Linear Diophantine Equation". Questions and Discussions. American Mathematical Monthly. 33 (7): 379–381. doi:10.2307/2298647. MR 1520987.
- Tweedie, M. C. K. (1939). "A graphical method of solving Tartaglian measuring puzzles". Math. Gaz. 23 (255). pp. 278–282. JSTOR 3606420.
- Saksena, J. P. (1968). "Stochastic optimal routing". Unternehmensforschung. 12 (1). pp. 173–177. doi:10.1007/BF01918326.
- Atwood, Michael E.; Polson, Peter G. (1976). "A process model for water jug problems". Cogn. Psychol. 8. pp. 191–216. doi:10.1016/0010-028\\\\\\\\\\\\\\\\
- 5(76)90023-2.

# Artificial Intelligence C.A3

## Different methods

## Water Jug problem using BFS

You are given a a litre jug,b litre jug and a c litre jug . The jugs are initially empty. The jugs don't have markings to allow measuring smaller quantities. You have to use the jugs to measure d litres of water where d is less than n.

(X, Y,Z) corresponds to a state where X refers to amount of water in Jug1 , Y refers to amount of water in Jug2 and Z refers to amount of water in Jug3.

Determine the path from initial state (xi, yi,zi) to final state (xf, yf,zf), where (xi, yi,zi) is (0, 0,0) which indicates all Jugs are initially empty and (xf, yf,zf) indicates a state which could be (0, d) or (d, 0).

The operations you can perform are:

Empty a Jug, (X, Y)->(0, Y) Empty Jug 1

Fill a Jug, (0, 0)->(X, 0) Fill Jug 1

Pour water from one jug to the other until one of the jugs is either empty or full, (X, Y) -> (X-d, Y+d)

Examples:

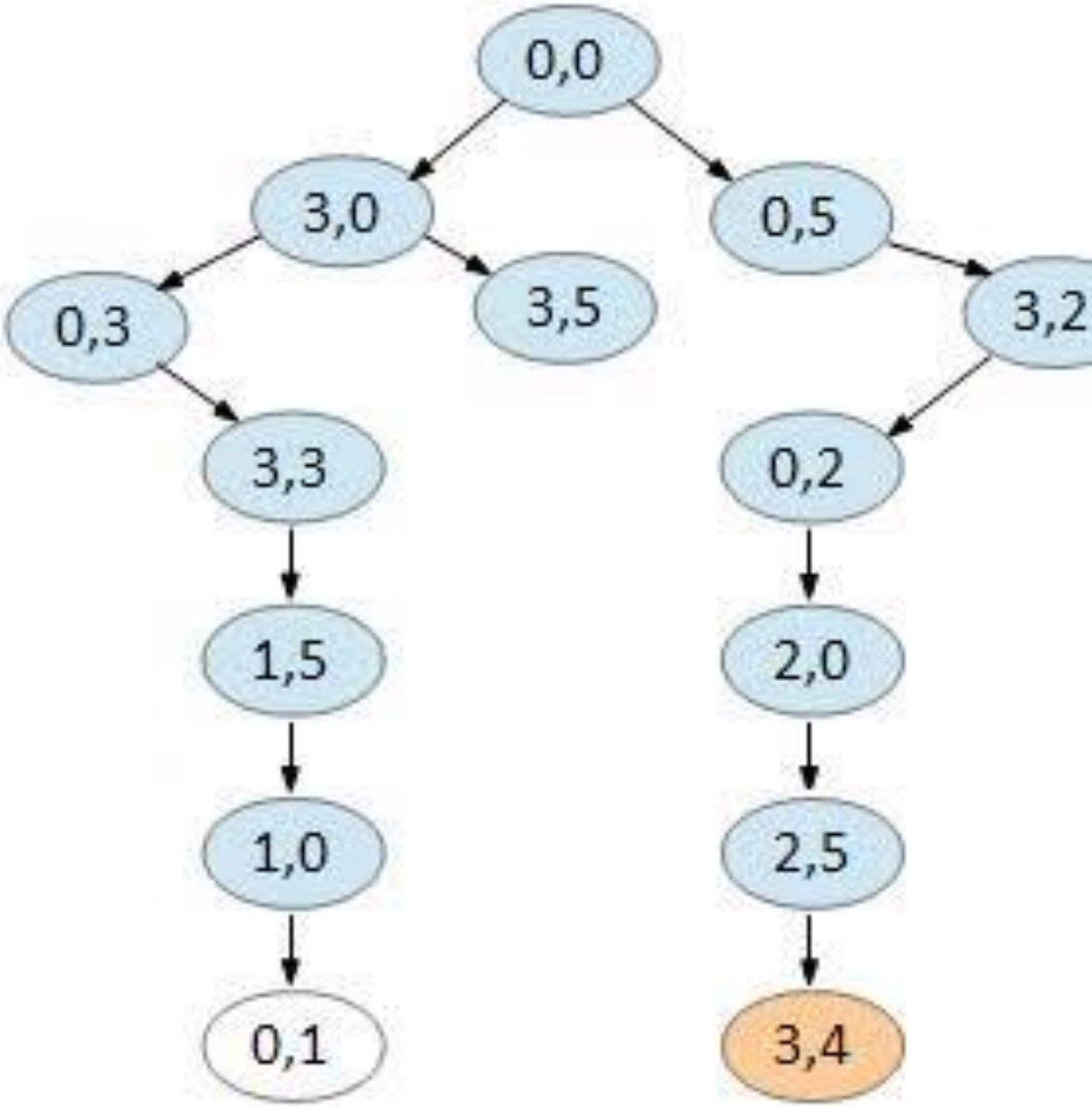
Input : 4 3 2

Output : {(0, 0), (0, 3), (4, 0), (4, 3),

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

(0, 2)}

# Artificial Intelligence C.A3




**NOTE:THIS DFD IS FOR WATER JUG USING TWO JUGS**

# Artificial Intelligence C.A3

## Coding /proposed methodology

```
from math import factorial

global list_previous_jug_states

list_previous_jug_states = []

global list_running_count

list_running_count = []

global list_running_index

list_running_index  = []

global list_volumes

list_volumes = []

global list_jug_max

list_jug_max = []


class CreateJugs:

def __init__ (self,jug_name,jug_max):

self.name = jug_name

self.max = jug_max
```

```
list_jug_max.append(self)


@property
def jug_max (self):
return self.max


def set_fill_states (number_jugs, jug_max):
global list_binary_states
list_binary_states = []
for i in range (1<<number_jugs):
binary_state =bin(i)[2:]
binary_state ='0'*(number_jugs-
len(binary_state))+binary_state
list_binary_states.append(binary_state)
list_binary_states =
list_binary_states[0:len(list_binary_states)-1]
```

```
new_list = []
for x in range (number_jugs):
for item in list_binary_states:
if item.count('1') ==  x:
new_list.append(item)
list_running_count.append(x)
list_binary_states = new_list[:]
for n in range (len(list_binary_states)):
jug_binary_state = list_binary_states[int(n)]
jug_state = []
for x in range (number_jugs):
if int(jug_binary_state[x]) == 1:
jug_state.append(list_jug_max[x].max)
else:
jug_state.append (0)
list_previous_jug_states.append(jug_state)
list_running_index.append([n])
```

# Artificial Intelligence C.A3

```
def make_moves (jug_state,
        running_total, running_index,
        target_volume, number_jugs):
for from_jug in range (number_jugs):
from_jug_max = list_jug_max[from_jug].jug_max
from_jug_state = jug_state[from_jug]

for to_jug in range (number_jugs):
    if to_jug == from_jug: continue

to_jug_max = list_jug_max[to_jug].jug_max
to_jug_state = jug_state[to_jug]

new_jug_state = jug_state [:]
new_jug_state[from_jug] = 0
if new_jug_state not in list_previous_jug_states:
```

# Artificial Intelligence C.A3

```
list_previous_jug_states.append(new_jug_state)
        list_running_count.append (running_total+1)
        new_index = [len(list_previous_jug_states)-1]
        list_running_index.append (running_index +
        new_index)
        new_jug_state = jug_state [:]
        new_jug_state[from_jug] = from_jug_max
    if new_jug_state not in list_previous_jug_states:

list_previous_jug_states.append(new_jug_state)
        list_running_count.append (running_total+1)
        new_index = [len(list_previous_jug_states)-1]
        list_running_index.append (running_index +
        new_index)

        if to_jug_state == to_jug_max: continue
        if from_jug_state == 0: continue
```

```
if from_jug_state < (to_jug_max-to_jug_state):
    new_jug_state = jug_state [:]
    new_jug_state[from_jug] = 0
    new_jug_state[to_jug] = to_jug_state + from_jug_state
else:
    amount_transfer = to_jug_max - to_jug_state
    new_jug_state = jug_state [:]
    new_jug_state[from_jug] = from_jug_state - amount_transfer
    new_jug_state[to_jug] = to_jug_state + amount_transfer

if new_jug_state not in list_previous_jug_states:

list_previous_jug_states.append(new_jug_state)
    list_running_count.append (running_total+1)
    new_index = [len(list_previous_jug_states)-1]
```

```
list_running_index.append (running_index +
new_index)

if any (jug == target_volume for jug in
new_jug_state):
print ("Target reached in ", running_total + 1,
"steps")
for item in running_index:
print (list_previous_jug_states[item])
print (new_jug_state)
return True

return False, 0

if __name__ == "__main__":
number_jugs = int(input("Please enter the number of
jugs you have: "))
```

```
for i in range (number_jugs):
jug_volume = int(input(f"Please enter the volume of jug {i+1}: "))
list_volumes.append(jug_volume)
target_volume = int(input("Please enter the target volume: "))
list_volumes.sort(reverse=True)
for i in range (number_jugs):
jug_name = "Jug" + str(i+1)
CreateJugs (jug_name, list_volumes[i])
set_fill_states(number_jugs,list_volumes)
for item in list_previous_jug_states:
jug_state = item

index = list_previous_jug_states.index(item)
running_total = list_running_count [index]
running_index = list_running_index [index]
is_destination = make_moves (jug_state,
```

```
                                        running_total,
                                        running_index,
                                        target_volume,
                                        number_jugs)

                                    if is_destination == True:
                                    print ("=====")
                                    break
```

The standard puzzle of this kind works with three jugs of capacity 8, 5 and 3 liters. These are initially filled with 8, 0 and 0 liters. In the goal state they should filled with 4, 4 and 0 liters. The puzzle may be solved in seven steps, passing through the following sequence of states (denoted as a bracketed triple of the three volumes of water in the three jugs):

$$[8,0,0] \rightarrow [3,5,0] \rightarrow [3,2,3] \rightarrow [6,2,0] \rightarrow [6,0,2] \rightarrow [1,5,2] \rightarrow [1,4,3] \rightarrow [4,4,0].$$

# Artificial Intelligence C.A3

Cowley (1926) writes that this particular puzzle "dates back to mediaeval times" and notes its occurrence in Bachet's 17th-century mathematics textbook.

## Result and discussion

After solving the water jug problems we find out the goal state and discussed its algorithm and different techniques such as Using BFS etc.

## References

^ Weisstein, Eric W. "Three Jug Problem". mathworld.wolfram.com. Retrieved 2020-01-21.

^ Hint to Riddle #22: The 3 & 5 Litre Die Hard Water Puzzle. Puzzles.nigelcoldwell.co.uk. Retrieved on 2017-07-09.

^ Weisstein, Eric W. "Three Jug Problem". mathworld.wolfram.com. Retrieved 27 August 2019.