# Algorithmics SAT 2

## Pseudocode:

Set variables for max quantities and desired quantity

X = 9

Y = 13

Z = 10

Set jugs to 0,0 as a tuple

Define function called pourx\_y(values):

Convert the tuple input to a list so the values can be changed

While True:

If the first element in value equals 0 or the second value equals y:

Break

Decrement the first value by one

Increment the second value by 1

Convert the list to a tuple

Return the value

Define function called poury\_x(values):

Convert value to a list so the values can be changed

While True:

If the second element in value equals 0 or the first value equals y:

Break

Decrement the second value by one

Increment the first value by 1

Convert the list to a tuple

Return the value

Define function called waterJug(value)

Set an empty queue for all possible values that stem from the current state

Set an empty dictionary for all seen values

Create a node based with id equal to value

Add the value to the seen values and assign it as the node

Enqueue value to the queue

While the queue is not empty:

Assign the first element of the queue to be the current value

Dequeue the first element from the queue

create an empty list called actions

append the values if the first element in jugs was filled

append the values if the second element in jugs was filled

append the values if the first element in jugs was emptied

append the values if the second element in jugs was emptied

append the values pourx\_y(current)

append the values poury\_x(current)

for each action in actions:

If the action has been seen:

if there’s less than 3 edges between the action and the current node and the action isn’t the current node:

Create an edge between the current node and the existing node with the identified value

Else:

enqueue the action to the queue

create a node with the id value of that action

add an edge between that node and the current node

Add the action to the seen values and assign it to be that node

If the node contains the desired value z:

highlight the node red

## Reflective Analysis:

When this task was just given out, I was very intrigued by the problem and wanted to jump right into coding. I was interested in learning about classes and coded up a watering state class by defining all the possible actions that could be undertook. However, this was short lived as the code didn’t use a queue and every time the jug combination would change, the previous state would be forgotten. I went back to the drawing board and decided to write pseudocode first before coding. At this point the DFS and BFS algorithms were just covered in class, so I opted to incorporate a form of the BSF into the design. Instead of reading every node that was connected to the current one, every possible value that could stem from the current node would be generated and added to the graph. Since this was a form of BFS, these possible values would be added to a queue rather than a stack. I opted for the BFS over DFS as the task required all possible options to be found. If the task was to find the desired value node or the path to it DFS would be a better option.