Solving Water Jug problem with Artificial Intelligence

Created by Christopher Look and Deven Thiel

What is the Water Jug Problem?

- We are given two jugs with no measure markings on them. One is a 7 gallon another is a 3 gallon Jug. The goal is to get exactly 5 gallons in the jug by pouring one jug to another jug.

- The same problem can be done with two jugs with any value x and y trying to get a result z.

- Our program takes in two **variables** and creates a knowledge base. It also (if given the goal state) will apply a search algorithm to find a solution.

Water Jug Rules

- a Fill the 7 gallon jug
- b Fill the 3 gallon jug
- c Empty the 7 gallon jug
- d Empty the 3 gallon jug
- e If the water of 7 gallon jug + the water of 3 gallon jug ≥ 3 gallon, then pour water from 7 gallon jug into 3 gallon jug until 3 gallon jug is full.
- f If the water of 7 gallon jug + the water of 3 gallon jug < 3 gallon, then empty water from 7 gallon jug into 3 gallon jug.
- g If the water of 7 gallon jug + the water of 3 gallon jug ≥ 7 gallon, then pour water from 3 gallon jug into 7 gallon jug until 7 gallon jug is full.
- h If the water of 7 gallon jug + the water of 3 gallon jug < 7 gallon, then empty water from 3 gallon into 7 gallon jug.

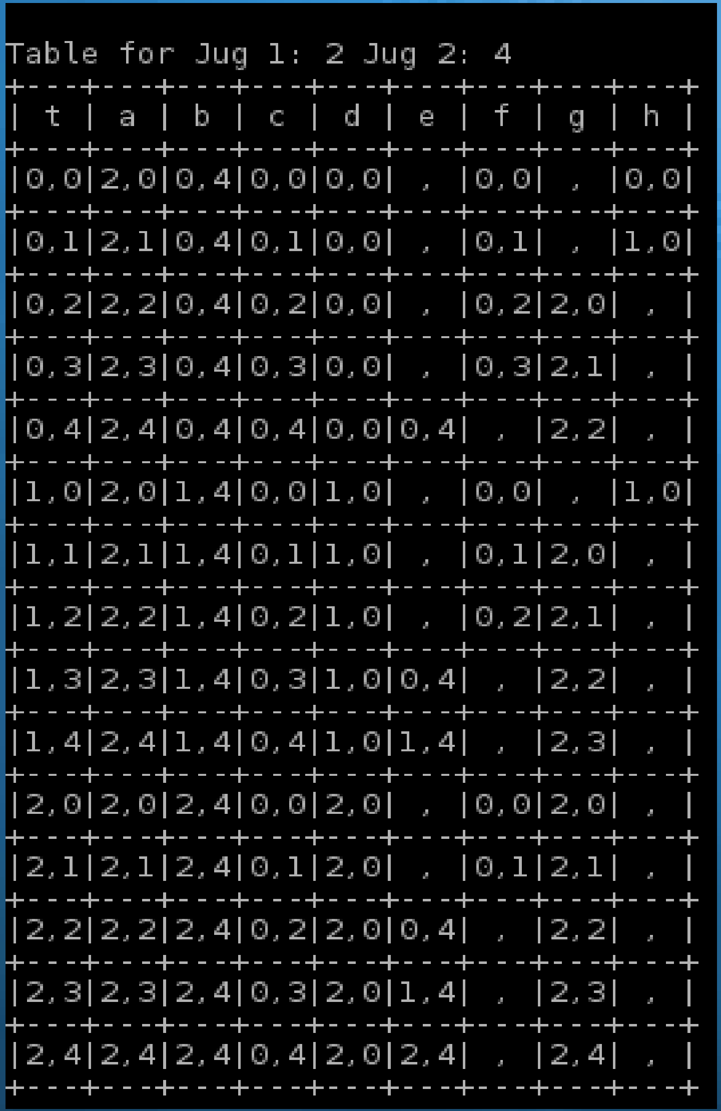
Knowledge Base

- Our program creates a table of possibilities based on what values the user has inputted. The transition table will always be (x+1) (y+1) number of rows.

- The transition table is created with columns A - H each one having a different rule for that column.

- The table is useful for users who want to find any goals within one set of values.

- Many times the problems have many solutions to the problem thus the table is useful to show all possible paths.



```
Table for Jug 1: 2 Jug 2: 4
+---+---+---+---+---+---+---+---+---+
| t | a | b | c | d | e | f | g | h |
+---+---+---+---+---+---+---+---+---+
|0,0|2,0|0,4|0,0|0,0| , |0,0| , |0,0|
+---+---+---+---+---+---+---+---+---+
|0,1|2,1|0,4|0,1|0,0| , |0,1| , |1,0|
+---+---+---+---+---+---+---+---+---+
|0,2|2,2|0,4|0,2|0,0| , |0,2|2,0| , |
+---+---+---+---+---+---+---+---+---+
|0,3|2,3|0,4|0,3|0,0| , |0,3|2,1| , |
+---+---+---+---+---+---+---+---+---+
|0,4|2,4|0,4|0,4|0,0|0,4| , |2,2| , |
+---+---+---+---+---+---+---+---+---+
|1,0|2,0|1,4|0,0|1,0| , |0,0| , |1,0|
+---+---+---+---+---+---+---+---+---+
|1,1|2,1|1,4|0,1|1,0| , |0,1|2,0| , |
+---+---+---+---+---+---+---+---+---+
|1,2|2,2|1,4|0,2|1,0| , |0,2|2,1| , |
+---+---+---+---+---+---+---+---+---+
|1,3|2,3|1,4|0,3|1,0|0,4| , |2,2| , |
+---+---+---+---+---+---+---+---+---+
|1,4|2,4|1,4|0,4|1,0|1,4| , |2,3| , |
+---+---+---+---+---+---+---+---+---+
|2,0|2,0|2,4|0,0|2,0| , |0,0|2,0| , |
+---+---+---+---+---+---+---+---+---+
|2,1|2,1|2,4|0,1|2,0| , |0,1|2,1| , |
+---+---+---+---+---+---+---+---+---+
|2,2|2,2|2,4|0,2|2,0|0,4| , |2,2| , |
+---+---+---+---+---+---+---+---+---+
|2,3|2,3|2,4|0,3|2,0|1,4| , |2,3| , |
+---+---+---+---+---+---+---+---+---+
|2,4|2,4|2,4|0,4|2,0|2,4| , |2,4| , |
+---+---+---+---+---+---+---+---+---+
```

Breath First Search

- Breath First Search is a searching algorithm that searches from the start state to each adjacent node before advancing further. Frequently used to find the end state

- Using the transition table that was created create after the user has built the data base. Once this is done it can traverse the data on the file to find the specific state needed.

Depth First Search

- Depth First Search is a searching algorithm that instead of searching all the adjacent nodes once I moves down it follows a path until that node branch ends. The problem with this is if the node branch is really long it has to backtrack until the start of that branch and goes to the next one over.

- Depth First Search strength is solving problems where you need to find out what is at the end of the path, therefore problems like mazes because it will follow one straight path to the end and backtrack to each starting path until it finds the goal.

- Both types of searches do different things however for finding the shortest path in this case Depth First Search will most of the time take longer unless the first path it choses is the goal state. This is because it will have to commit to one path after it choses it where as in Breath First Search it will explore all possible options on that level of the tree before advancement.

Conclusion

- In essence the Water Jug problem is a math problem that is solved by first creating a database, then using a search method to find a solution.

- However mapping out all the states is really the only way to know for sure that the goal state is not one of the overlooked states. Either a table or a tree will do that efficiently.

- Our program does both displays a table and outputs it to a file and simulates a tree to find the shortest path from the starting location. Each time the program is ran the location and goal can be different depending on what the user has in mind. If there is no goal inputted or is it is invalid it would only display the table and output it to the file