Bonus Question (S8)

CSI4106: Introduction to Artificial Intelligence

Submitted by:

Arthur Leung 6766207

Christine Kandalaft 7216942

Water Jug Problem (Subject 8)

Subject S8: You are given two jugs, a 4-gallon one and a 3-gallon one, a pump which has unlimited water which you can use to fill the jugs, and the ground on which water may be poured. Neither jug has any measuring markings on it. The objective is to get exactly 2 gallons of water in the 4-gallon jug. Use blind search and informed search.

Note: util.py and astar_search.py were based off of both our Assignment 1 code and Filip Drobnjakovic/Meesam Haider's Bonus Subject 1 solution.

Assumption: Each move has a cost of 1 (i.e. arc length).

Implementation:

For this blind search, we have used BFS, which does not give the optimal path (as our move costs are the same as the arc length), but explores in 'breadth' (so it visits extra states). We also used DFS, which does not give the optimal path either.

For informed search, we have used A star with heuristic h(s) where s is a state (x,y):

h(s) = |x-G| + |y-G| ; such that x,y are jugOne (4G) and jugTwo (3G) respectively and G is the goal value (2 gallons)

The heuristic is relaxing the condition of needing exactly 2 gallons in the 4 gallon jug as the goal state. Instead, we will try to reach 2 gallons of total water in *either* of the two jugs (4 gallon and 3 gallon jugs). This can be seen in h(s) as we minimize h(s) as the distance between both x (the 4 gallon jug) and y (the 3 gallon jug) with the goal state (2 gallon) respectively.

This is admissible and therefore optimal for A star as the real cost c(s) to the goal is always the sum of arc lengths, while the estimated cost is the distance is the calculated distance h(s).

For the final move, the actual arc length cost to goal is 1, while the estimated cost h(s) is 0.

Therefore, h(s) <= c(s).

Output:

```
Initial Config

DFS ----------------------------------

The Solution is   [(4, 0), (4, 3), (0, 3), (3, 0), (3, 3), (4, 2), (0, 2), (2, 0)]

The Solution is at depth  8

The path cost is  8

Number of visited nodes: 8
```

```
The execution time is  0.000657630274265323 seconds.

Done!

BFS-----------------------------

The Solution is   [(0, 3), (3, 0), (3, 3), (4, 2), (0, 2), (2, 0)]

The Solution is at depth  6

The path cost is  6

Number of visited nodes: 13

The execution time is  0.000868142485384035 seconds.

Done!

A* -----------------------------------

The Solution is   [(0, 3), (3, 0), (3, 3), (4, 2), (0, 2), (2, 0)]

The Solution is at depth  6

The path cost is  6

Number of visited nodes: 11

The execution time is  0.0010617290915886793 seconds.

Done!
```

Conclusion:

As we can see in the results, DFS finds a solution at depth 8 (and path cost of 8), visiting 8 nodes in 0.000657630274265323 seconds. BFS finds a solution at depth 6 (and path cost of 6), visiting 13 nodes in 0.000868142485384035 seconds. A* finds a solution at depth 6 (and path cost of 6), visiting 11 nodes. While DFS is the fastest in computational time (and visits the least nodes), it does not return an optimal solution. On the other hand, BFS and A* return the optimal solution, with BFS being faster in computational time. That said, A* visits less nodes than BFS, as BFS searches by ‘breadth’.

Although DFS is the most efficient in this case, it should not be used if the problem had a chance to follow an infinite path or an optimal solution was required. If a optimal path was required, BFS or A* (with our heuristic) would be used.