Connor Norton

CSCI-6511

**Project 1: Water Jug Problem**

For this project, I developed a Java program to solve a variation of the water pitcher problem. To solve this “search” problem, we were required to develop and implement an A\* algorithm to find the ideal, shortest path. This requires that dynamically created graphs, when there are no nodes or edges in sight.

There are 4 primary actions that can be taken within the scope of the problem.

1. A pitcher can be filled to its capacity.
2. A pitcher can be drained until it is empty.
3. A pitcher can be poured into another pitcher.
4. A pitcher can be poured into the goal pitcher.

The A\* Algorithm

GOAL STATE

* When the Pitcher in the index 0 position is equal to the goal value. Does not consider the states of all other pitchers.

COST

* This is equal to the number of steps required to reach the goal state.

HEURISTIC

* Minimum Difference calculation
  + Looks for the pitcher that is closest to the goal volume.
  + Saves the minimum difference to a variable.
* Checks if pouring between 2 pitchers will create a volume even closer to the goal volume.
  + If yes, the minimum difference is set to this value.
* Minimum difference value returned.

HEURISTIC LOWER BOUND

* The lower bound of a heuristic is the smallest possible estimate of the remaining distance to the goal. In the case of the water jug problem, the lower bound of the heuristic would be the minimum number of steps required to reach the goal state from the current state, assuming that pouring water from one jug to another can change the relative capacities of the two jugs.
* The lower bound is the number of steps required to reach the goal state.
* When goal = 0, Heuristic(g) = 0

UNIT TESTING

* JUnit Testing is implemented in Eclipse for several test cases. The purpose of the tests is to determine if suboptimal solutions exist. For example, is Pitchers={2,5,6,72} and goal={143}, the algorithm returns 8, when the optimal solution is 7. This shows that my algorithm is suboptimal in some cases.