EECS 114: Assignment 5

Nov 24, 2015

Due 7 Dec 2015 at 11:59pm

Scenario - In the motion picture *Die Hard 3*, Lt. John McClane (Bruce Willis) and Harlem electrician Zeus Carver (Samuel L. Jackson) are forced to scramble across the city and solve mind-bending riddles in ridiculously short amounts of time to prevent bombs, planted by a telephoning terrorist known only as *Simon* (Jeremy Irons), from detonating. One of the diabolical riddles is the following: Given a `3`-gallon jug and a `5`-gallon jug, fill the `5`-gallon jug with exactly `4` gallons. The solution they reached, which leaves `4` gallons in the `5` gallon jug is:

- Fill the `5`-gallon jug.
- Pour the `5`-gallon jug into the `3`-gallon jug.
- Empty the `3`-gallon jug.
- Pour the `5`-gallon jug into the `3`-gallon jug.
- Fill the `5`-gallon jug.
- Fill the `3`-gallon jug from the `5`-gallon jug.

1 Water Jugs Problem

For this assignment you will solve a more generalized version of this riddle. To ensure we only have a single solution, we will require that the smaller jug, in this case the `3` gallon jug, be empty once the required number of gallons are in the larger jug. Thus a correct solution to the *Die Hard 3* example would be `4` gallons in the 5 gallon jug and zero gallons in the `3` gallon jug.

You have two jugs, `A` and `B`, and an infinite supply of water. There are three types of actions:

- Fill jug `A` or fill jug `B`.
- Empty jug `A` or empty jug `B`.
- Pour water from one jug into the other.

Pouring water from one jug into the other stops when the first jug is empty or the second jug is full, whichever comes first. For example, if `A` has `5` gallons and `B` has `6` gallons and `B` has a capacity of `8`, then pouring from `A` to `B` leaves `B` full and `3` gallons remaining in `A`. Each action is now associated with a positive cost. A problem is given by the parameters `(Ca, Cb, N, fA, fB, eA, eB, pAB, pBA)`, where `Ca` and `Cb` are the capacities of the jugs `A` and `B`, respectively, and `N` is the goal. The cost to fill `A` is `fA`, `fB` is the cost to fill `B`, `eA` is the cost to empty `A`, and `eB` is the cost to empty `B`, `pAB` is the cost to pour `A` to `B` and `pBA` is the cost to pour `B` to `A`. A solution is a sequence of actions that leaves jug `A` empty, and exactly `N` gallons in jug `B`.

The possible actions are listed below where `fill` means to fill the jug from the infinite water supply, `empty` means to discard the water in the jug, `pour A B` means to pour the contents of jug `A` into jug `B` (this action stops when `B` becomes full or `A` becomes empty), and `success X` indicates that the goal has been accomplished, that is, jug `A` is empty, jug `B` contains `N` gallons, and the total cost of the operations is `X`.

```
fill A
fill B
empty A
empty B
pour A B
pour B A
success X
```

The output from your program will consist of a series of instructions from the list of the possible actions which will result in jug `A` being empty and jug `B` containing exactly `N` gallons of water. The last line of output for each puzzle should be the line `success X`, where `X` is the total cost of all the actions. Output lines start in column `1` and there should be no empty lines nor any trailing spaces.

2 Program Specification

Write a `Jug` class that takes 9 parameters in the constructor, namely `(Ca, Cb, N, fA, fB, eA, eB, pAB, pBA)`. Note that the order of the parameters is important. Use `assert` in your program to ensure the following: $0 < Ca \leq Cb$ and $N \leq Cb \leq 1000$. Add anything you need to the following class.

```
public class Jug
{
    // Solve() is used to check the input and find the solution if one exists
    // returns -1 if the input is invalid.
    // returns 0 if input is valid but a solution does not exist.
    // returns 1 if solution is found and also prints solution
    int Solve();
};
```

Sample Input

```
Jug die_hard(3, 5, 4, 1, 2, 3, 4, 5, 6);
die_hard.solve();
```

Sample Output

```
fill A
pour A B
fill A
pour A B
empty B
pour A B
fill A
pour A B
success 27
```

You will generate a graph to model this problem. The vertices of your graph will store information about the number of gallons of water in each of the two jugs, `A` and `B`. The edges in your graph will represent the actions possible for a given vertex. Your program must find the solution in the shortest number of actions. To achieve this you will implement the Breadth First Search(`BFS`) algorithm to find the shortest path in a graph where all edges have a weight of 1. To solve the *single-pair-shortest-path* problem for a graph with edge weights that vary, but are all positive, you will implement *Dijkstra's Algorithm*, and run it on your graph representation for the problem.

3 Assignment Breakdown

Here are the steps you should take to complete this assignment.

1. Decide what data structures, i.e., lists, arrays you will use to implement a graph generating function. You may use any of the data structures we have studied so far. Feel free to use `Java's Collections`. However using any pre-implemented graph or graph algorithm programs that you may find on-line is *prohibited*.

2. Write a test case (returns `true` if all conditions are met after a function is called and `false` otherwise) to test the correctness of your graph generating function developed from your design. Your test case should test that all vertices were correctly created and that they have the correct edge structure.

3. Implement and test your graph generating function.

4. Implement `BFS`. Use `BFS` on your graph to find the shortest path for the problem with the following parameters: `Jug die_hard( 3, 5, 4, 1, 1, 1, 1, 1, 1);`

5. Implement *Dijkstra's Algorithm* to find the shortest path when the graph costs has non-equal positive weights, e.g., `Jug die_hard( 3, 5, 4, 6, 5, 4, 3, 2, 1);` For full credit, use your min-heap PQ class from assignment 4.