Lab 6: Pitfall

Due: Tuesday 03/19/2024 @ 11:59pm EST

The purpose of labs is to practice the concepts that we learn in class. To that end you will be writing java code that uses a game engine called Sepia to develop agents that solve specific problems. In this lab we will be playing the game we discussed in class: **Pitfall**. In order to make Pitfall work in Sepia, I had to change it to a two-player game. Fear not, the rules of Pitfall will remain the same: except now there is a (reactive) enemy agent controlling the board (to replace units when you attack them, etc).

1. Copy Files

Please, copy the files from the downloaded lab directory to your cs440 directory. You can just drag and drop them in your file explorer.

- Copy `Downloads/lab6/lib/pitfall.jar` to `cs440/lib/pitfall.jar`.
  This file is the custom jarfile that I created for you.
- Copy `Downloads/lab6/data/labs/pitfall` to `cs440/data/labs/pitfall`.
  This directory contains a game configuration and map files.
- Copy `Downloads/lab6/src/labs` to `cs444/src/labs`.
  This directory contains our source code `.java` files.
- Copy `Downloads/lab6/pitfall.srcs` to `cs440/pitfall.srcs`.
  This file contains the paths to the `.java` files we are working with in this lab. Just like last lab, files like these are used to speed up the compilation process by preventing you from listing all source files you want to compile manually.
- **There is no documentation this time**. There are no methods/functionality from `pitfall.jar` that you will need to use: everything is self contained within `src/labs/pitfall/BayesianAgent.java`

2. Test run

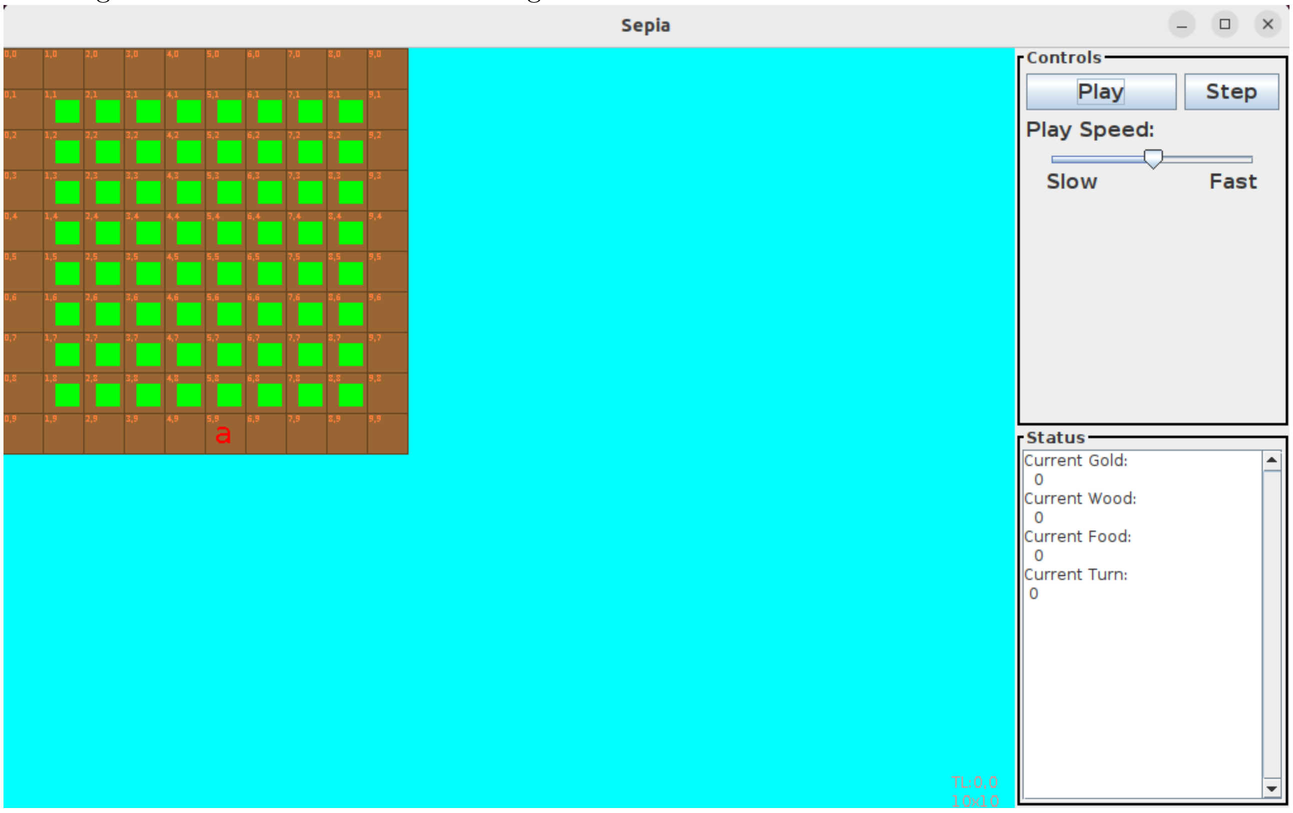
If your setup is correct, you should be able to compile and execute the given template code. You should see the Sepia window appear.

```
# Mac, Linux. Run from the cs440 directory.
javac -cp "./lib/*:." @pitfall.srcs
java -cp "./lib/*:." edu.cwru.sepia.Main2 data/labs/pitfall/easy/8x8.xml

# Windows. Run from the cs440 directory.
javac -cp "./lib/*;." @pitfall.srcs
java -cp "./lib/;." edu.cwru.sepia.Main2 data/labs/pitfall/easy/8x8.xml
```

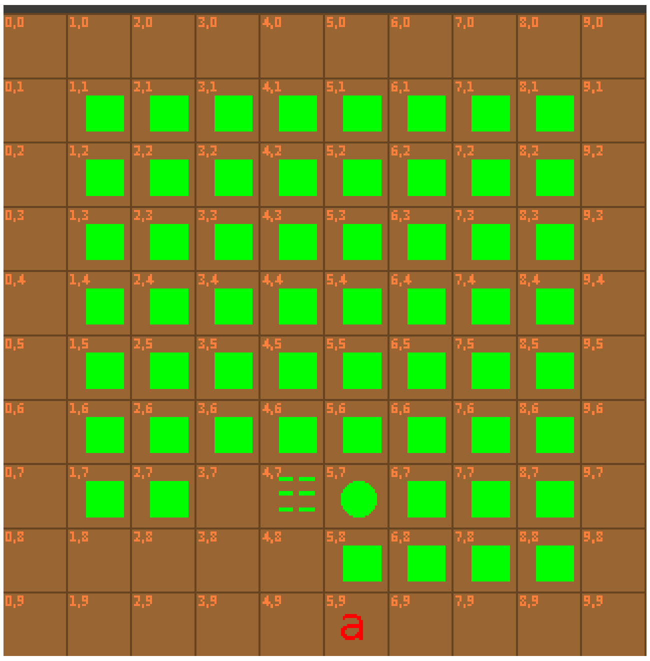
3. Pitfall Rules & Information

The game of Pitfall is very similar (in fact a simplified) version of another game called *Minesweeper*. You are given a grid of squares, the contents of each square are initially hidden from you. The player interacts with the game by clicking on a (currently hidden) square, at which point the contents of that square are revealed. A square can contain two kinds of things: a pit, or nothing. If you click on a square that reveals a pit, you fall into the pit and you lose. If you click on a sqaure that does not have a pit, you are instead told whether or not adjacent squares contain pits (cardinal directions are adjacent) through the form of a *breeze*. If you perceive a *breeze*, you know that some combination of unknown adjacent squares contain pits.
Our version of Pitfall is similar. Let's say that you run `data/labs/pitfall/easy/8x8.xml`. You should get a window that looks something like this:



Current Gold:
0
Current Wood:
0
Current Food:
0
Current Turn:
0

Any square that has a green box within it are hidden (i.e. unclicked on) squares. Your agent controls the red "a" at the bottom of the grid. The border of unmarked squares are not part of the game: they are only there so that the "a" has somewhere to exist on the map.
Your agent will play Pitfall by attacking squares in the map. Your agent is only allowed to attack hidden squares (i.e. squares with green boxes in them): attacking any other square will result in that square firing back at you (and killing you). Your unit as well as all hidden squares die in one hit. Once a hidden square has been killed, the enemy agent will replace that unit with a new unit whose ascii-art is either empty (a safe square) or three dotten lines (a breeze). For instance, consider the partial state of the following 8 × 8 game:



Here we can see that square (4,7) contains a breeze (so some adjacent squares of (4,7) contains pits), and we unfortunately explored (5,7) which has a pit. The next turn of this game, we will lose.

4. Inference Rules

Your agent needs to use the breeze information (called "clues") to figure out the probability that each of the "frontier" squares (the set of unexplored squares adjacent to explored squares) contain a pit given the breeze information that we have seen. Essentially, I am asking you to use the equation we derived in class to implement a specific method that will calculate this probability.

5: Bayesian Network Inference (50 points)

In this task, I want you to implement the `src.labs.pitfall.agents.BayesianAgent.PitfallBayesianNetwork.getNextCoordinateToExplore` method. This method should iterate over every coordinate in the frontier (calculated for you) and use the breeze evidence so far (calculate for you) to compute $Pr[P_{i,j}|breezes]$. We have already derived an equation for this probability, and your method should implement this equation.

I have already provided code in this method. Currently, your agent will randomly guess a frontier square (so that you can see the game play if you run the game). This behavior however is highly likely to run into pits and lose the game, so replace this code! If you want to find this method quickly, I left you a "TODO" comment with a some tips.

Notes

- Whenever you want to test your implementation, I would recommend playing any of the three games in the `data/labs/pitfall/easy` directory. Like the name of the directory states, there are easy games, medium (difficulty) games, and hard games. I would recommend testing your code initially on the easy games and then moving on to trying the medium and hard games. Here are the games that you can run for each difficulty:
  - `easy`: There are easy games with three different sized boards: 8x8, 9x9, and 10x10. They are contained in the `8x8.xml`, `9x9.xml`, and `10x10.xml` files. Each easy game has a pit probability of 0.1.

  - `medium`: There are two medium games with different sized boards: 13x15 and 16x16. They are contained in the `13x15.xml` and `16x16.xml` files. Each medium game has a pit probability of 0.25.
  - `hard`: There two hard games: a 30x16 contained in the `30x16.xml` file, and a 30x30 contained in the (30x30.xml) file. These games have a pit probability of 0.4.

- You may create whatever helper methods you want in order to accomplish this goal, however I am not sure that you will need to create any in `BayesianAgent.java`.

Task 6: Submitting your lab

Please submit `BayesianAgent.java` on gradescope (just drag and drop in the file). Be warned, the autograder can take a while!