Artificial Intelligence Homework 5
Due date: 4/12 by 11:59pm

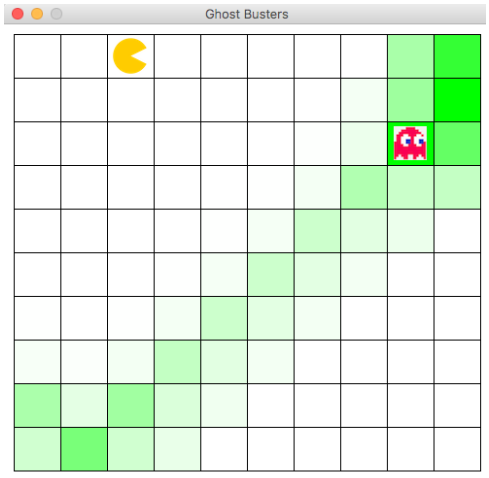
In the distant past, Pac-Men did not flee from ghosts. They hunted ghosts using only their ears. In this assignment, you will restore Pac-Man to his ghost-hunting origins using exact and approximate inference algorithms for tracking.

This is a programming assignment and you may work with a partner if you choose. Click here to download the starter code. You will notice that the code is arranged in packages:

- **graphics:** This package contains the graphical user interface and the sonar class. The GhostBustersPanel class contains different parameters that you can change:
    - There are 3 types of ghost to choose from: Ghost which just stands still, RandomGhost which chooses a random action, and GoEastGhost which simply goes east until it hits a wall.
    - There are 2 types of Pac-Man that you will implement: PacmanForwardAlgorithm and PacmanParticleFilter.

  You can change both the type of ghost and the type of Pac-Man in the constructor of the GhostBustersPanel class.
- **characters:** This package contains all of the ghost and pacman characters.
- **util:** This package contains a Coords class (to represent the cells on the grid) and a WeightedSet class. You do not need to implement any of these data structures however you should read through them because you will be using them extensively. The WeightedSet class in particular has a lot of extra functionality that I added to make this assignment easier for you.

Ghost Hunting



Pac-Man and the ghost live on a 10x10 grid world and can only move in cardinal directions: up, down, left, and right.

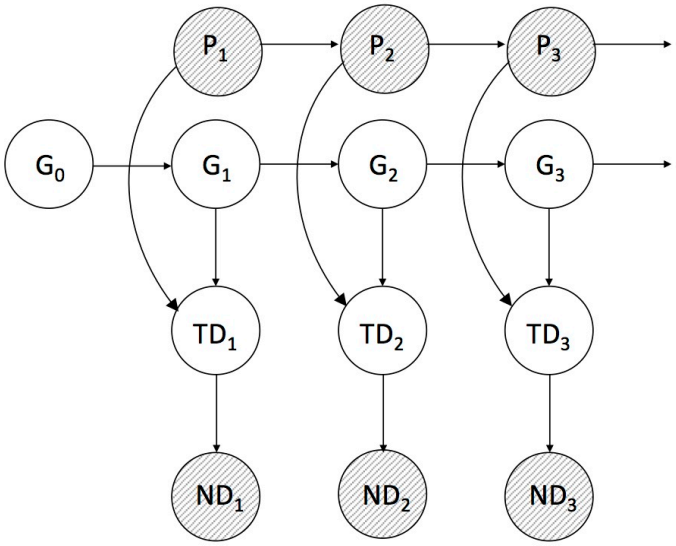
At each time step, a ghost chooses between staying where it is or moving to 1 of its cardinal neighbors. Depending upon the ghost's location, it may have 2, 3, or 4 neighbors.

Pac-Man uses his ears (sonar) to obtain a noisy estimate of the Manhattan distance between him and the ghost. Unfortunately, his ears are not perfect and Pac-Man does not know the true Manhattan distance. He has only his noisy estimate to use when tracking the ghost.

Pac-Man maintains a distribution over where he believes the ghost is located. The darker green cells indicate a higher level of belief and the lighter green cells indicate a lower level of belief.

Dynamic Bayesian Network

The image below shows the dynamic Bayesian Network that we will be using to model this domain.



- P_t is a random variable that tracks Pac-Man's location. The possible values of this random variable are the cells in the grid -- i.e., {(0,0), ..., (9,9)}. This random variable is shaded to indicate that Pac-Man knows (i.e. observes) his own position at each time step.

- G_t is a random variable that tracks the ghost's location. The possible values of this random variable are also the cells in the grid. The transition distribution $p(G_t \mid G_{t-1} = (x,y))$ tells us the probability of the ghost's location at time t given that the ghost was at location (x,y) at time t-1. This distribution should give equal probability to the ghost's current position and neighbors. Thus, from location (0,0) there is a 0.33 chance of staying at (0,0), a 0.33 chance of moving to (0, 1) and a 0.33 chance of moving to (1,0).

- TD_t is a random variable that represents the *true distance* between Pac-Man and the ghost. The probability distribution $p(TD_t \mid P_t, G_t)$ gives probability 1.0 to the true Manhattan distance and probability 0.0 to all other distances. (This is one way of encoding a deterministic variable as a random variable.) Thus, if $P_t = (0,0)$ and $G_t = (2,3)$ then $p(TD_t = k \mid P_t, G_t)$ is 1.0 for k = (3-0)+(2-0) = 5 and 0.0 for any other value of k.

- Finally, ND_t is a random variable that represents the *noisy distance* between Pac-Man and the ghost. The probability distribution $p(ND_t \mid TD_t = k)$ is provided by the Sonar class. You can call the method getEmissionTable(k) to get the probability distribution of the noisy distance given the true distance equals k.

Using Inference to Track The Ghost

Pacman.java is an abstract class that contains the abstract method:

```
public abstract void update(int noisyDistance);
```

There are 2 classes that extend Pacman: PacmanForwardAlgorithm and PacmanParticleFilter. Each of these classes must implement the abstract method update() which takes in the most recent observation of the ghost's position (i.e. the noisy distance) and updates Pacman's belief distribution over the location of the ghost.

Pac-Man's belief distribution is stored in an instance variable in Pacman.java. *Your update() method should store the updated belief distribution in this instance variable*. My move() method then takes this updated belief distribution and chooses a direction for Pacman to move.

In PacmanForwardAlgorithm, the update() method should use The Forward Algorithm to compute:

$$p(G_t \mid ND_{1:t})$$

The pseudocode for this algorithm is posted on the course webpage.

In PacmanParticleFilter, the update() method should use a particle filter to estimate the same distribution. Again, the pseudocode for this algorithm is posted on the course webpage.

Both of these algorithms require you to compute $p(ND_t \mid G_t)$. You will need to derive this quantity given the dynamic Bayesian network and the distributions defined above (using the inference by enumeration algorithm we discussed in class).

Note that your update() method performs only a single iteration of The ***Forward*** *Algorithm or the Particle Filtering algorithm. It is my code (in GhostBustersPanel) that will call update() at each time step to get the updated belief distribution. Thus, the constructor should perform the initialization step (using the prior distribution), and the update() method should then take that belief distribution and update it given an observation of the noisy distance of the ghost.*

Debugging Your Code

I highly recommend breaking your code into small methods so that you can test each method. I have done this for you inside the PacmanParticleFilter class.

I also highly recommend computing the Forward Algorithm values *by hand* on a sheet of paper to check your code against. If you do this, you should change the grid size to something more manageable like 3x3. (You can change the size of the grid inside of GhostBustersPanel.) Here is the emission distribution for a 3x3 grid:

$p(ND_t \mid TD_t)$

ND=0	**ND=1**	**ND=2**	**ND=3**	**ND=4**	**TD**
0.7	0.2	0.1	0.0	0.0	**TD=0**
0.3	0.4	0.2	0.1	0.0	**TD=1**
0.1	0.2	0.4	0.2	0.1	**TD=2**
0.0	0.1	0.2	0.4	0.3	**TD=3**
0.0	0.0	0.1	0.2	0.7	**TD=4**

Initially, the ghost has an equal probability of being in any of the cells -- i.e., $p(G_0)$ is a uniform distribution over the cells.

Running Your Code

The main() method is in the GhostBusters class. This class does not take any command line arguments.

Grading and Submission

Your assignment will be graded based upon correctness, efficiency (i.e. the time/space complexity of your implementation), as well as its adherence to the Java style guide.

To submit, rename your directory hw5_FirstName_LastName. Then zip your directory and upload it to Moodle. Please remember to rename your directory *before* you zip it.

Acknowledgements

This assignment was taken and adapted from the Pacman Projects.

Last modified: Tue Jul 15 13:34:17 PDT 2014