CSC 242 Project 3
Maryfrances Umeora
mumeora
I did not collaborate with anyone for this project.

Program Files Used

- BN Base Package
    - All teacher provided files, plus some edits by me
- BN Core Package
    - All teacher provided files, plus some edits by me
- BN Examples Package
    - All teacher provided files, plus some edits by me
- BN Parser Package
    - All teacher provided files
- BN Util Package
    - XandW.java
- BN Inference Package
    - EnumerationInferencer.java
    - RejectionSampler.java
    - LikelihoodWeighter.java
    - GibbsSampler.java
- Default Package
    - Main.java

How To Run My Project

I wrote our project in Java, so it should be able to run on any Java compiler and/or terminal. To check my inferencers, you can either run the main method of the AIMA alarm example, where I have neatly shown the results with all four inferencers.
Or, you can run Main.java using the command rules specified in the project description (pg 10-11).
Remember, when mentioning file, use the format:

```
src/bn/examples/aima-alarm.xml
```

You have to specify the location cos cmd isn't smart :)

Part I

Implement Bayesian Network - Teacher provided

Part II

This involved implementing the algorithm for Exact Inference.

The formula for Exact Inference is described below:

$$\mathbf{P}(X \mid \mathbf{e}) = \alpha \mathbf{P}(X, \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y})$$

Basically, for every possible value of the random variable X (if it is boolean, these values would be *true* and *false*), then return the sum of the probabilities of all variables *y* in the network given their parents, considering *y* as evidence, sort of multiplied by the probabilities of the other *y*'s given their parents. Ya, it was fun.

For the most part, I followed the enumerate-ask algorithm from the book. However, the book's algorithm is hard to understand, so I used this article (http://courses.csail.mit.edu/6.034s/handouts/spring12/bayesnets-pseudocode.pdf) from MIT to help guide me.

Part III

This part of the project had two sub-parts (technically 3, but I will discuss Gibbs Sampling separately as Bonus).

Recommended Number of Samples: 100000 or above

↓

First discussed was Rejection Sampling.

It's pretty straight-forward. First I generated prior samples with my `priorSample()` method. It returns an assignment or *possible world*. We reject all the samples that are not consistent with the query (see my `isConsistent()` method in bn.base ⇒ Assignment.java) by simply only considering those that *are*. For every generated sample that is consistent, we add one to the current probability that we're returning.

↓

Next was Likelihood Weighting.

It's hard to explain in just a paragraph, but the basic idea is to only generate worlds where we don't have to reject samples. In order to make sure the values we get in the end are still indicative of the real world, we keep track of a decimal value *weight*, and we update this weight according to evidence variables. In the end, instead of adding 1 to the probability we're returning, we add the weight of the world.

In my code, I follow the book algorithm which involves storing a vector of weighted counts for each value of X, then I call my `weightedSample()` method. It generates an event that is consistent with the evidence and returns this event along with its associated weight. Finally, I add to that vector the new value of the probability of each random variable in it.

↓

Note that when you run my Part III with command line, I show the results of all three types of approximate inference.

Part IV

This was basically just making the run configurations. Depending on what the input is, I run *x* type of inferencer on *y* file.
See Main.java in the default class.
My code *can* run xml files, but I'm not sure about bif files :(

BONUS

I did Gibbs Sampling for 20% extra credit.
You can find the code in bn.Inferencer → GibbsSampler.java.
Now, this one was fun. But honestly, I did it in the middle of the night, staring at the very tiny words of the textbook and somehow it ended up working, but my code is well-commented, so look at that for explanations!

♥thank you for reading my readme!♥