MCMC Part 1: Boolean

This is the first part of two labs to be done INDIVIDUALLY (no groups). As discussed in class, write a program to implement Gibs sampling. Your code will evetually need to use Metropolis Hastings (or just Metropolis) to handle continuous nodes but that is in the next part of the lab, take a look at that lab so you know where you are headed. For now you only need to handle Bernoulli nodes.

Please demonstrate your code on the networks given below AND on one you make up yourself. Share the network you make up by posting it (and a query) on the discussion page (called "Digital Dialog" in Learning Suite) BUT DO NOT POST THE ANSWER. Run your code on a couple of networks that have be posted by others.

Because of numerical issues, we strongly recommend doing calculations on the log scale.

Please submit a report on your code that is complete enough that I can see what you have done without having to run your code, although you need to submit the code too. Include your answers from the tests below, and from the discussion page. Include some mixing plots and, of course, prior and posterior plots for each example (the point isn't to see how many trees you can kill) but again, you must demonstrate that your code worked and got the right distributions.

Example Networks

Burglar Alarm

The Burglar Alarm example is described in Russell and Norvig's book "Artificial Inteligence a Modern Apporach". If you don't have the book, look at slides 5 and 6 on this pdf. This network involves only discrete nodes.

The model probabilities are:

```
P(Burglary=true) = 0.001
P(Eathquake=true) = 0.002
P(Alarm=true | Burglary=true, Earthquake=true) = 0.95
P(Alarm=true | Burglary=true, Earthquake=false) = 0.94
P(Alarm=true | Burglary=false, Earthquake=true) = 0.29
P(Alarm=true | Burglary=false, Earthquake=false) = 0.001
P(JohnCalls=true | Alarm=true) = 0.90
P(JohnCalls=true | Alarm=false) = 0.05
P(MaryCalls=true | Alarm=true) = 0.70
```

```
P(MaryCalls=true | Alarm=false) = 0.01
```

According to Russell and Norvig:

```
P(Burglary | JohnCalls=true, MaryCalls=true) = <0.284,
0.716>
```

Simulation should give a similar answer.

Our code reported the following results as well:

```
P(Alarm | JohnCalls=true, MaryCalls=true) = <0.75, 0.25>
P(Earthquake | JohnCalls=true, MaryCalls=true) = <0.17,
0.83>
P(Burglary | JohnCalls=false, MaryCalls=false) = <0.00,
1.00>
P(Burglary | JohnCalls=true, MaryCalls=false) = <0.01,
0.99>
P(Burglary | JohnCalls=true) = <0.02, 0.98>
P(Burglary | MaryCalls=true) = <0.05, 0.95>
```

Notes

- You should not need to use much from any statistics libraries. Really all you need to be able to do is to sample from a bernoulli.
- Try to keep your code somewhat general. You really should not have to have a separate Metropolis implementation for each different type of node.
- For Bernoulli/Binomial, sample directly instead of trying to use Metropolis.