COS 402 – P3

1.

After about 40,000 iterations the probability of Burglary being true converged to 0.28 to two significant digits of the true probability. After about 1,000,000 iterations the probability converged to 0.284, which is three significant digits of the true probability.

```
 100000  0.28084719  0.71915281
 200000  0.28363858  0.71636142
 300000  0.28380905  0.71619095
 400000  0.28551179  0.71448821
 500000  0.28441343  0.71558657
 600000  0.28505452  0.71494548
 700000  0.28410531  0.71589469
 800000  0.28499714  0.71500286
 900000  0.28455524  0.71544476
1000000  0.28521171  0.71478829
```

2.

The probability Burglary is true converged to 0.0032 after about 1,000,000 iterations.

```
10000000  0.00328190  0.99671810
20000000  0.00330650  0.99669350
30000000  0.00328523  0.99671477
40000000  0.00327850  0.99672150
50000000  0.00326626  0.99673374
60000000  0.00326837  0.99673163
70000000  0.00327119  0.99672881
80000000  0.00327337  0.99672663
90000000  0.00326963  0.99673037
100000000  0.00327260  0.99672740
```

3.

The probability of PropertyCost converges to [0.46, 0.34, 0.17, 0.026] after about 20,000,000 iterations.

```
10000000  0.46665715  0.34516387  0.16294068  0.02523830
20000000  0.46212518  0.34003508  0.17106664  0.02677310
30000000  0.45502648  0.34205872  0.17564399  0.02727080
40000000  0.45606976  0.34224612  0.17459175  0.02709237
50000000  0.45487085  0.34173967  0.17617180  0.02721768
60000000  0.45519573  0.34169671  0.17555763  0.02754993
70000000  0.45621675  0.34262178  0.17405664  0.02710483
80000000  0.45734246  0.34223127  0.17349920  0.02692707
90000000  0.45948797  0.34241544  0.17157546  0.02652112
```

100000000 0.46006425 0.34208515 0.17126049 0.02659012

4.

The first query was to probability of an accident given that the driver's *Age* is an *Adolescent*, the *Antilock* is *False*, *GoodStudent* is False and the *DrivingHist* is *One*. Given this data we would except the probability of an accident being a little greater, mainly because the driver is young and one might infer that someone that is a bad student may be less likely to learn proper driving technique than a bad student. The results are shown below:

1000000	0.60348140	0.13986086	0.11278389	0.14387386
2000000	0.62809519	0.13390293	0.10224245	0.13575943
3000000	0.63824645	0.12712796	0.10030730	0.13431829
4000000	0.63761409	0.12779722	0.10064847	0.13394022
5000000	0.65134747	0.12647997	0.09531558	0.12685697
6000000	0.65890822	0.12246831	0.09418615	0.12443731
7000000	0.65905105	0.12092541	0.09460356	0.12541998
8000000	0.65754042	0.11898099	0.09615799	0.12732061
9000000	0.65969393	0.11617976	0.09638021	0.12774610
10000000	0.65622233	0.11571869	0.09806829	0.12999069

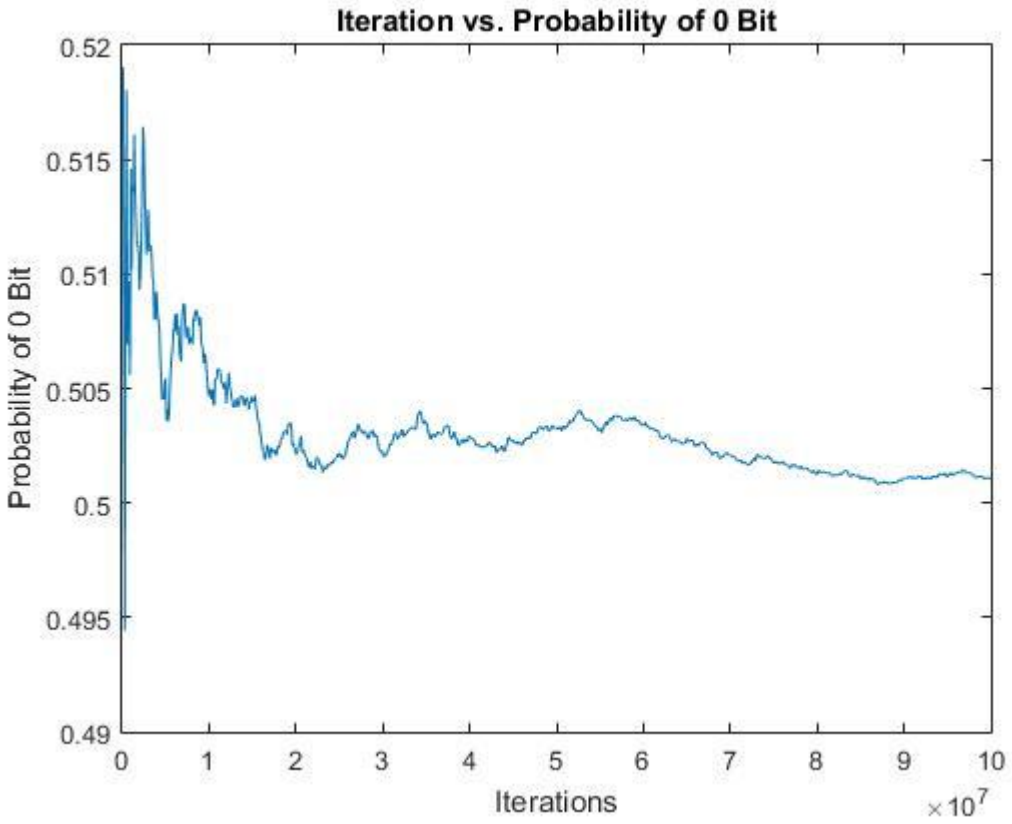
The results make some sense in that there is weight given to the latter three, however less so than expected. This might be because of the combination of variables, being an adolescent might increase the probability for an accident (right-most variables), but this might be offset that the driving history is relatively good (one accident for an adolescent).

The second query is the probability of *LiabilityCost* given that the driver's *Age* is an *Adolescent*, the *DrivQuality* is *Poor*, and the *VehicleYear* is *Current*. We would infer that the liability cost would be higher because the driver is young, poor driver with a new car. Intuitively we might think that the driver is more likely to be in an accident and because the car is new, the liability is much higher. The result are shown below:

1000000	0.92985507	0.03551496	0.02030898	0.01432099
2000000	0.93370153	0.03305548	0.01960949	0.01363349
3000000	0.93154069	0.03453799	0.02008466	0.01383666
4000000	0.93379877	0.03339099	0.01949325	0.01331700
5000000	0.93671061	0.03179199	0.01881820	0.01267920
6000000	0.93514601	0.03260266	0.01919250	0.01305883
7000000	0.93769844	0.03128985	0.01864928	0.01236243
8000000	0.93732126	0.03137200	0.01883212	0.01247462
9000000	0.93682801	0.03154722	0.01898466	0.01264011
10000000	0.93714761	0.03129140	0.01884820	0.01271280

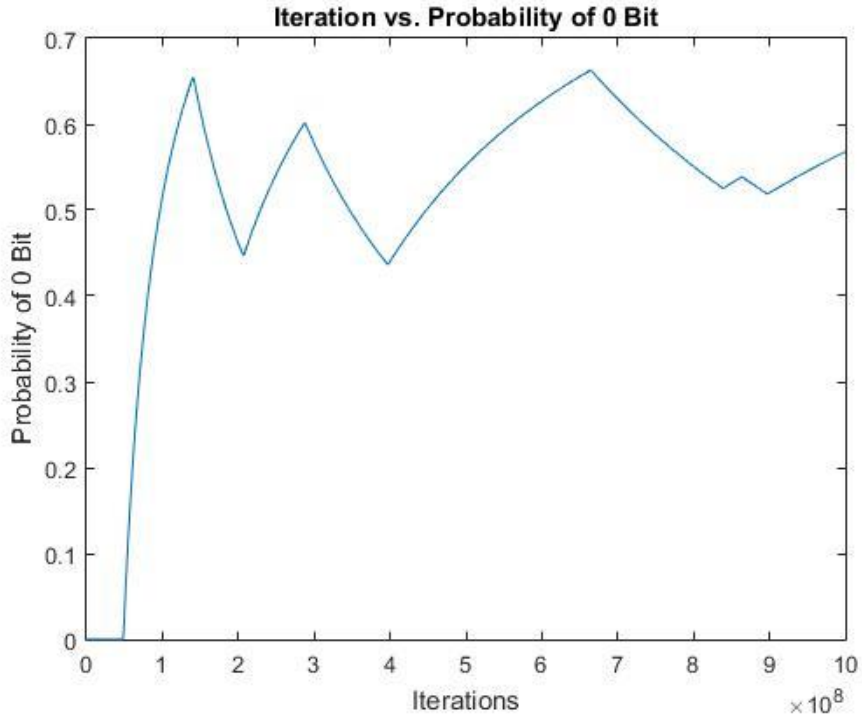
These results are very unexpected, from the explanation above, I expected more weight to be put on the variables on the right. This might occur because of the other variables built into the network (the nodes that hold safety ratings and cost might offset this).

5. The query is the probability of the *Leader*, since query has no evidence variables, the answer is the conditional probability of *Leader*. Since this has no parent, it is simply [0.5, 0.5].

6. 

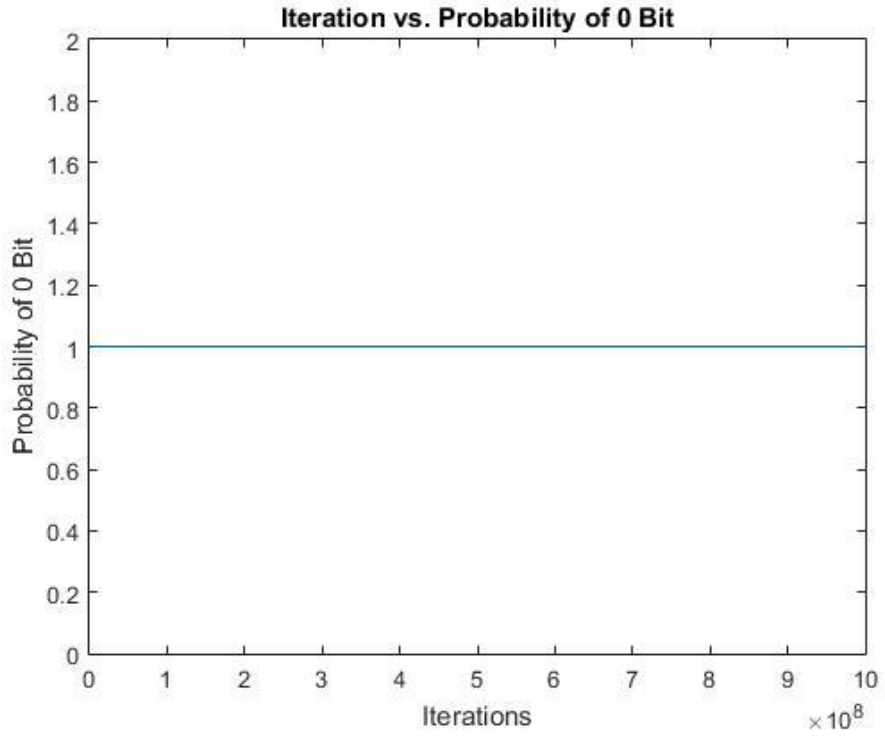
The MCMC appears to converge to about 0.501 after about $6x10^7$ iterations. At first, the probability fluctuates and stabilizes after about $6x10^7$ iterations, but still fluctuates a little.

7. For k =25:



The probability reaches between 0.4 and 0.6 after the first $1x10^8$ iteration, but continues to fluctuate largely between those two values.

For k = 50:



The results are very unexpected. We might be inclined to think that the probability of the leader choosing 0 is 1.0. However, we know that makes no sense (since it is 0.5). Therefore, It must mean that MCMC has not had enough iterations to converge to the true value. Even with a billion iterations the Bayes Net with 50 followers fails to even fluctuate between certain values. Based on the design of the net this occurred because the initial random value of Leader was 0. If it so happened that it was 1, then the probability would be 0.

8.

From the three examples above, it appears that MCMC has trouble approximating the probability of this Bayes Net as the number of followers increases. Even with a billion iterations, the last net failed to get anywhere close to the true probability.

The reason this occurs is because of the distribution of the followers and the number of followers in the Bayes Net. For the distribution of the query to be accurately approximated, MCMC must eventually sample from P(Leader | MB(Leader)) and change its value. Let's say the leader randomly start out at 0. In this case the P(Leader = 0| MB(Leader)) is proportional to P(Leader=0) * P(Follower1 | Leader =0) * P(Follower2 | Leader = 0)*… What ends up happening is eventually the MCMC picks a random follower node and samples from it; this has a large probability of switching the follower to what the leader's current value is. Eventually all the followers will have the same value as the leader. This completely dominates the distribution of Leader, since the opposite value of what the leader started out as is proportional to 0.1^k. Making it virtually guaranteed that the leader stays the same in MCMC. This means that once the leader switches values it stays at that value for a long period of time.

These experiments show that it is difficult to show how many iterations are necessary and when the probability converges. From the last example it would appear that it did converge; one billion iterations is qualitatively a lot, however, in this example it was nowhere near enough. Even though the last graph looked like it converged, it didn't.

9.

The network I constructed simulates a class at Princeton a student might take. At the top level it has the class difficulty and professor quality. The child of those are the class rating and the amount of class attended. The child of class attended are the midterm grades and the number of psets completes. Finally, the child of the last two is the final grade. Intuitively we might think a class's difficulty is the top since it doesn't depend on anything else; COS 402 might be harder than COS 126. Included with that is how good a professor is. We are inclined to believe that a classes rating at the end of a semester is based on how difficult the class was and how well the professor taught it, same with how often the student attended class. Additionally the amount of class attended affects a student's midterm grade and the number of psets that student completes. Finally, the last two contribute the final grade and is a predictor of how the student might do on the final. In general I believe this network is moderately realistic, since there might actually be more edges (a student might attend less classes after the midterm if he/she did well). Additionally, there would be more values that each variable could take on. Overall though, I believe the network is intuitive.

The first query is the probability of *FinalGrade* given that *Midterm* is *above* average and the number of *PsetsCompleted* is *none*. This is interesting because it will reveal how much weight the network gives to psets of the midterm (because some classes weigh the midterm higher and vice-versa).

The second query is the probability of *ClassRating* given that the *FinalGrade* was *below* average, *ClassAttended* was *above* average, and the *ProfQuality* was *good*. It is unclear what the answer might be since, on one hand, if a student did poorly they might be resentful and give the class a low rating, however, if the professor taught well and the student attended most of the classes, it suggests that the class was interesting and the student liked it.

10.

For the first query P(FinalGrade | Midterm=above, PsetsComplete=none):

100000	0.49721503	0.29527705	0.20750792
200000	0.49442253	0.30146349	0.20411398
300000	0.49729501	0.30007567	0.20262932
400000	0.49652626	0.30098425	0.20248949
500000	0.49681901	0.30229140	0.20088960
600000	0.49687917	0.30211116	0.20100966
700000	0.49814215	0.30178100	0.20007686
800000	0.49911688	0.30011962	0.20076350
900000	0.49830500	0.30113189	0.20056311
1000000	0.49907650	0.30047570	0.20044780

The results are somewhat reasonable. From the probabilities given above it seems the network places a larger weight on psets; if you don't do the psets, it severely hurts your grade. However, looking at the probabilities of an average and above average grade, we see that they are similar which shows that an average midterm score can result in a good final grade (given you do amazing on the final). This is mainly reasonable because half of the time you will do below average since you did average on the midterm and you haven't done any of the psets, meaning your current grade is below average and therefore it is likely your final grade will be as well.

For the second query P(ClassRating | FinalGrade=below, ClassAtended=above, ProfQuality=g):

100000	0.47162528	0.34565654	0.18271817
200000	0.47005765	0.34838326	0.18155909
300000	0.47302509	0.34570551	0.18126940
400000	0.47041382	0.34926663	0.18031955
500000	0.47160106	0.34824330	0.18015564
600000	0.46949255	0.35010608	0.18040137
700000	0.47012219	0.34967664	0.18020117
800000	0.46978191	0.34964831	0.18056977
900000	0.46944948	0.34996294	0.18058758
1000000	0.47003253	0.34918765	0.18077982

The results are a little surprising. The probabilities show that the class rating is heavily influenced by the final grade; students that did poorly will likely not give the class a good review, despite the quality of the professor and the amount of class attended. This result is somewhat consistent with my intuition, since ultimately if you did poorly throughout the class and you tried (shown by attendance) you might naturally blame the class since you simply did not understand the material and therefore it must be the class's fault (a student like this probably shouldn't major in this class's department). Although, it is a little strange that there is such a large weight given to a below average rating despite the quality of the professor, this might be because the network isn't as consistent as it should be (since it is difficult with the number of conditional probabilities).

In general these queries reveal how a query on a network can reveal a lot about how the network works. In this case the queries revealed a lot about where the network placed weight in determining query variables.