Woo-Jong Jang
A97105059
Sarah Haroon
A10938989

CSE 150 Programming Assignment #4 Report

Question 1:

$P(J = 1| B = 0, E = 1) = (P(J = 1| B = 0, E = 1, A =1) * P(A = 1|B = 0, E = 1)) +$
$(P(J = 1| B = 0, E = 1, A =0) * P(A = 0|B = 0, E = 1))$
$= 0.9(0.29) + 0.05(0.71)$
$= .2965$

Question 2:

$P(B = 1| J = 1) = (P(J = 1| B = 1)* P(B))/P(B)$

$P(J = 1| B = 1) = (P(J = 1| B = 1, A = 1)*P(A =1| B =1))+$
$(P(J = 1|B = 1, A = 0)P(A = 0| B = 1))$
$= 0.9*0.94 + 0.5*0.05998 = 0.84899$

$P(A=1|B = 1) = (P(A=1|B = 1, E = 1)*P(E = 1)) +$
$(P(A=1|B = 1, E = 0)*P(E = 0))$
$= (0.95*0.002) + (0.94*0.998)$
$= 0.05998$

$P(J) = P(J|A=0)P(A=0) + P(J|A=1)P(A=1)$
$P(J|A =0) * [P(B = 0)P(E = 0)P(A = 0|B=0,E=0)$
$+ P(B = 0)P(E = 1)P(A = 0|B=0,E=1)$
$+ P(B = 0)P(E = 0)P(A = 0|B=1,E=0)$
$+ P(B = 1)P(E = 1)P(A = 0|B=1,E=1)]$
$+ P(J|A = 1) (0.00252)$ {From Lecture 13}
$= (0.05)(0.999*0.998*0.999 + 0.001*0.998*0.06 + 0.001*0.002*0.05)$
$+ 0.9*0.00252$
$= 0.05(0.9974) + 0.9(0.00252) = 0.052$

$P(B = 1| J = 1) = (P(J = 1| B = 1)* P(B))/P(B) = (0.848999*0.001)/0.0521 = 0.0163$

Question 3:

Description of Algorithms:

a. Rejection Sampling:
    i. For rejection sampling, we made two helper functions to achieve topological ordering and prior sampling named *topologicalOrder(self, outgoingNode, NodeEdges)* and *priorSample(self, givenVars)* respectively. The prior sampling function creates a Sample object and setAssignment the givenVars. Then it copy of root nodes list and passes that into the topological ordering function. The topological ordering function iterates through the network's edges and appends the destination node only when the source node of the edge is contained in the passed in list and if the node doesn't exist in the list. This modified topological node list is iterated through to get the probability of each node's variables and if it's not a given variable, runs a random() function to set the Sample object for that node to be True or False depending on the random's result compared to the probability. This priorSample() returns a dictionary of probabilities associated with the names of the nodes. Using this list, rejectionSampling() runs random() to check the list's queryValue's probability to see if the simulation would return True for the sample or not. With this, we increment the True count for probability and else we increment the False value of the count. The return probability would be the True count divided by the summation of True and False counts.
b. Likelihood Weighting:
    i. This method requires the assistance of two helper methods.
    ii. Helper method 1: Topological order - this method takes in a list of root nodes (nodes that are the source) that will be filled with the rest of the nodes in the map in topological order. That list will start with the root nodes. And This method also takes in all the edges in the map. It starts off with the root nodes and appends its children. This algorithm basically goes through the map in a BFS fashion while appending the nodes.
    iii. Helper method 2: weighted sample - This method returns a dictionary with the key of node strings and its value (true or false) and the weighted value. It takes in the given variable dictionary, the key being the variable name and the value being the probability of the given variable. First initialize the weight variable to 1.0. For each variable in topological order (via helper method 1), if the variable is an evidence variable then weight variable= weight variable*P(current variable|parents(current variable)) or else the current variable is randomly sampled - for that I randomly sampled a number and made it true if that number is less than or equal to the

P(current variable|parents(current variable)). This returns the dictionary of variable names and its values(true/false) and the weight variable.

iv. Likelihood Weighting method: This method takes in the variable we are trying to get the probability of, the given variables and the number of samples we want. We have weight-total variable and a weight-when-true variable. They are both initially zero. We get the weighted sample number-of-samples-we-want times. In that loop, when the variable we are trying to get the probability of is true, we increment weight-when-true with the returned weight of that sample and for all times we increment weight-total with the current sample's weight. And then we return the normalized weight or the weight-when-true divided by weight-total.

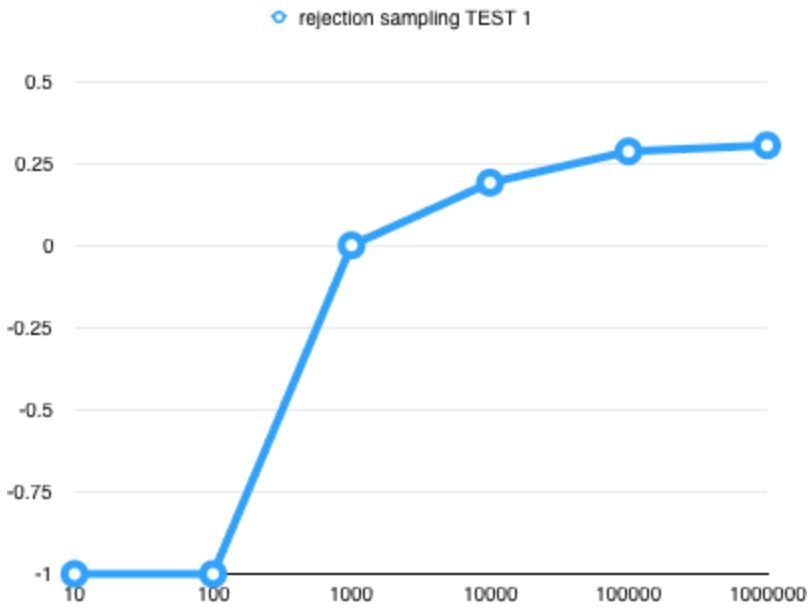
c. Gibbs Sampling: For Gibbs sampling, I wrote a helper function that returned the $P(x_i = True \mid mb(x_i))$. Using this value, I ran a random value generator to see if I should increment the True count or the False count. Then I returned True count divided by the number of times ran.

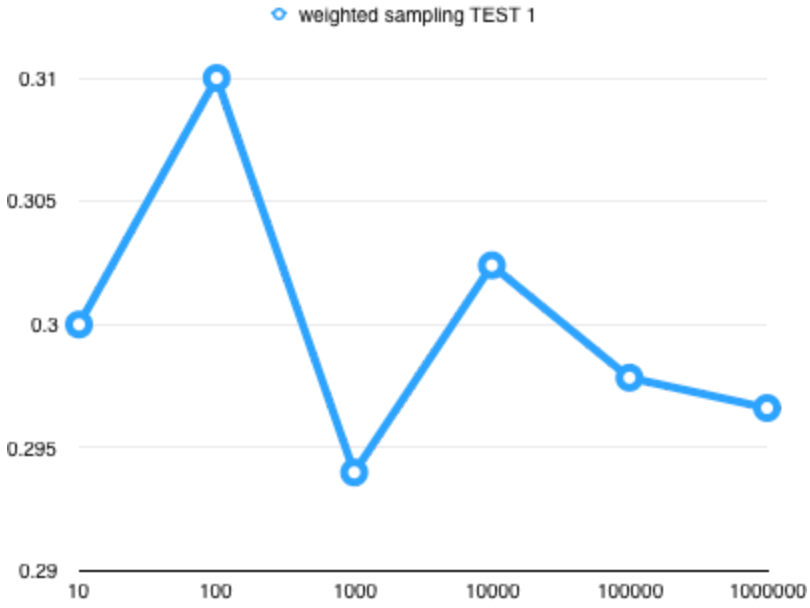
i. For the helper function that returned the $P(x_i = True \mid mb(x_i))$ value, I first ran $P(x_i = True \mid Parents(x_i))$ in a loop checking for parents and children of x_i and getting the probabilities of it. Then I multiplied the probability values of the *CapitalPI(*$P(y_i = True \mid Parents(y_i))$) by looping through the parents of children of x_i. The value of *probTrue* which is the multiplication of *CapitalPI(*$P(y_i = True \mid Parents(y_i))$ and $P(x_i = True \mid Parents(x_i))$. The value of *probFalse* is the same with $P(x_i = False \mid Parents(x_i))$. I return the value of (*probTrue* / (*probTrue* + *probFalse*)).

For each algorithm and each query, perform test with at least 5 different reasonably large values for the number of trials parameter, and graph how the estimated probabilities converge to the correct probability.

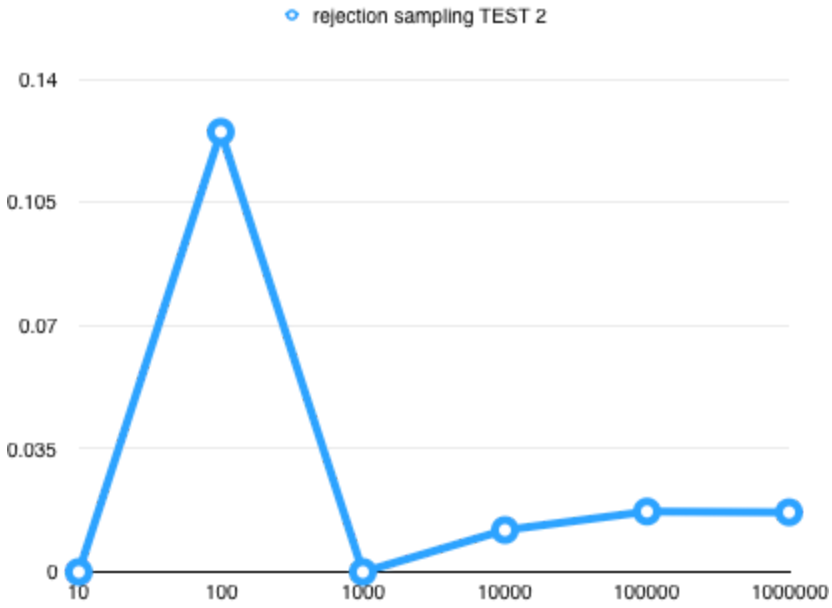
Runs	**10**	**100**	**1000**	**10000**	**100000**	**1000000**
rejection sampling TEST 1	-1	-1	0.0	0.190476190476	0.285714285714	0.304029304029
weighted sampling TEST 1	0.3	0.31	0.294	0.3024	0.297840000001	0.296611000004
Gibbs TEST 1	0.272072720727	0.270792707927	0.267742677427	0.272312723127	0.267252672527	0.2717527175277
rejection sampling TEST 2	0.0	0.125	0.0	0.011904761904	0.017173051519	0.0169501308296

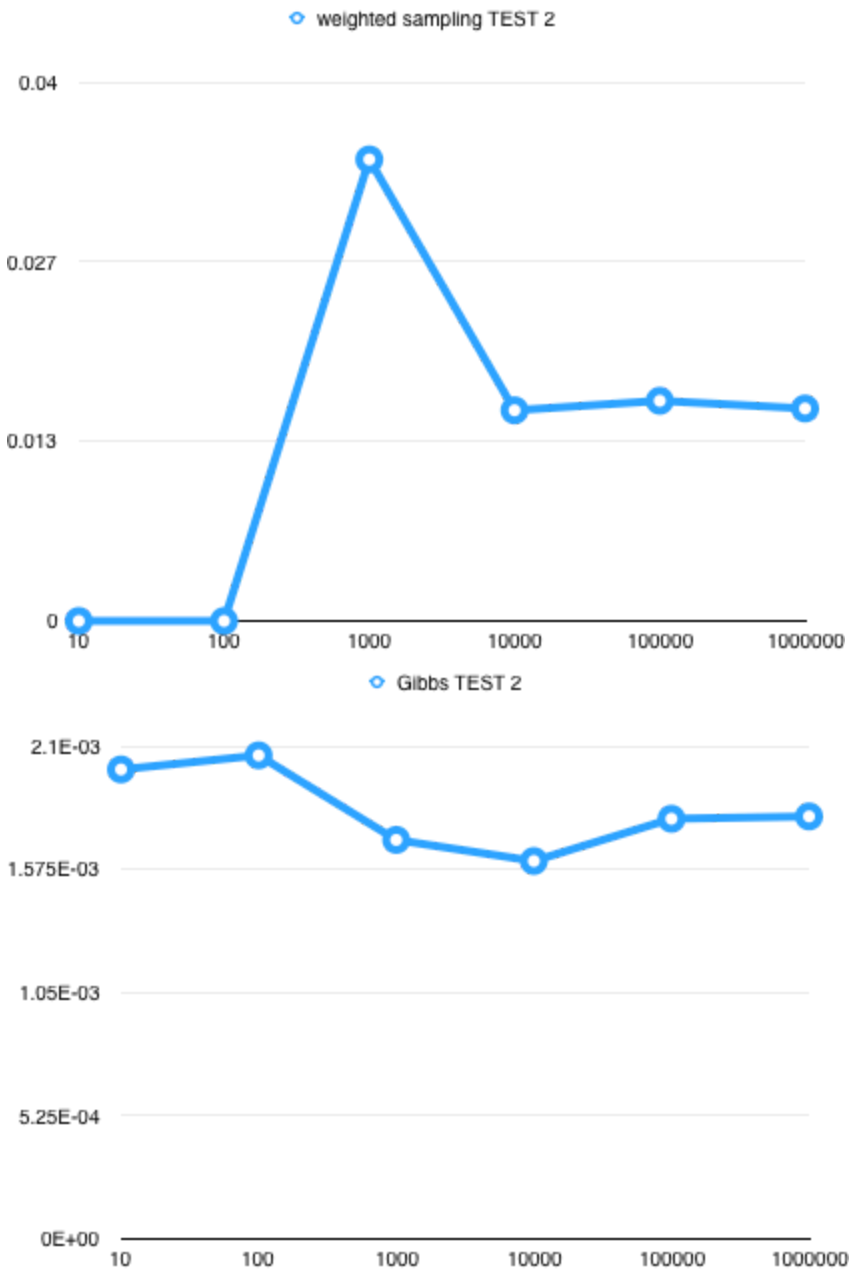
weighted sampling TEST 2	0.0	0.0	0.0342530922931	0.0156354916067	0.0163421285094	0.0157696861989
Gibbs TEST 2	0.0020000200002	0.0020600206 0021	0.0017000170 0017	0.0016100161 0016	0.0017900179 0018	0.0018000180 0018






Gibbs TEST 1
0.273
0.271
0.27
0.268
0.266
10
100
1000
10000
100000
1000000


rejection sampling TEST 2
0.14
0.105
0.07
0.035
0
10
100
1000
10000
100000
1000000

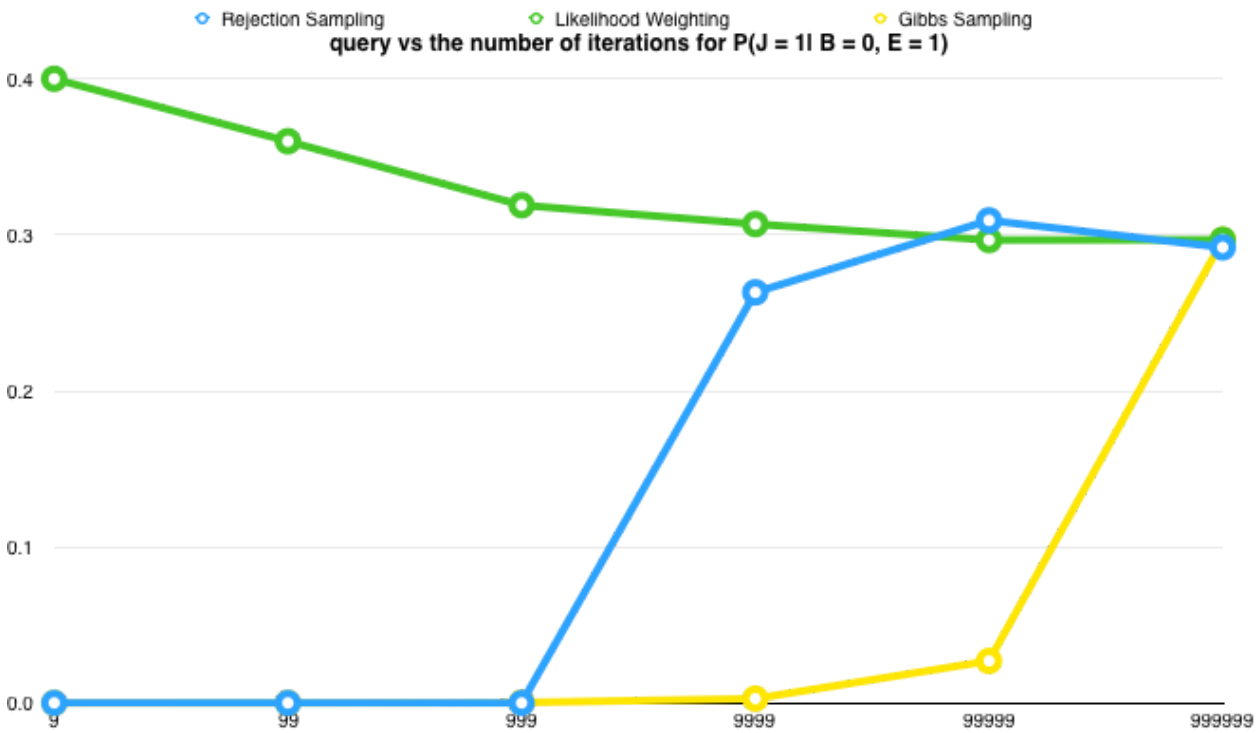


C. How many trials do you need for each method to give an estimate that was reasonably close to the actual probability? How do these methods compare to each other? Discuss your findings.
   a. Rejection Sampling: Around range 100 to 1000 trials, the jumps get smaller and close in around the probability.
   b. Likelihood Weighting: In around ranges 10 to 1500 trials, the jumps seems to vary the most and after 1500 seems to close in on the probability
   c. Gibbs Sampling: The estimation differences between each runs is very small. So the graph has large jumps since it zooms in on small changes.

From all three probabilities, Gibbs Sampling seems to be the most consistent in having the smallest amount of differences. Test 1 for all three algorithms seem to have more consistent values than Test 2. This might be because the actual probability value is smaller for Test 2.

D. Plot the probability of each query vs the number of iterations for all the three algorithms

	9	99	999	9999	99999	999999
Rejection Sampling	0.0	0.0	0.0	0.2631578947368421	0.30927835051546393	0.2919708029292
Likelihood Weighting	0.4	0.36000000000000004	0.31900000000000041	0.30689999999997064	0.29669000000075535	0.296859296863
Gibbs Sampling	0.000004000004000004	0.000038000038000038	0.000261000261000261	0.002742002742002742	0.026938026938026938	0.296611296615

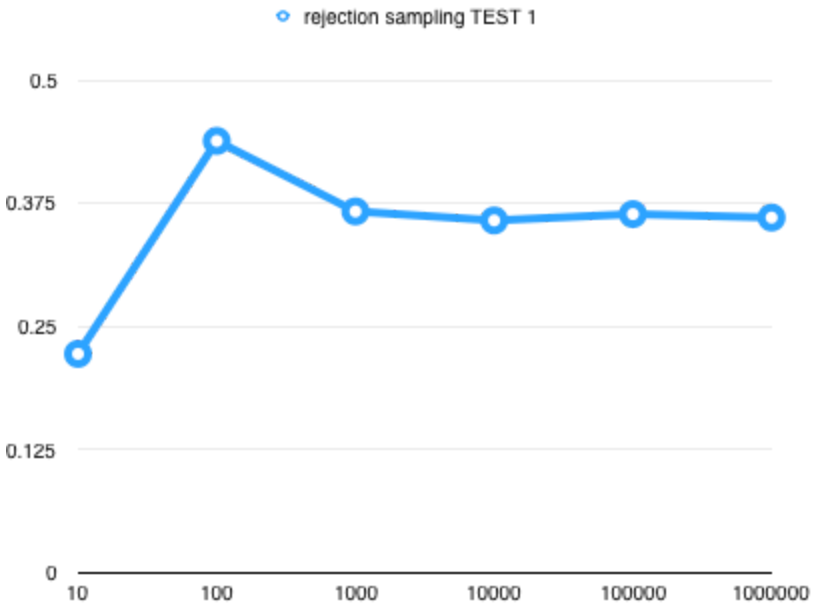


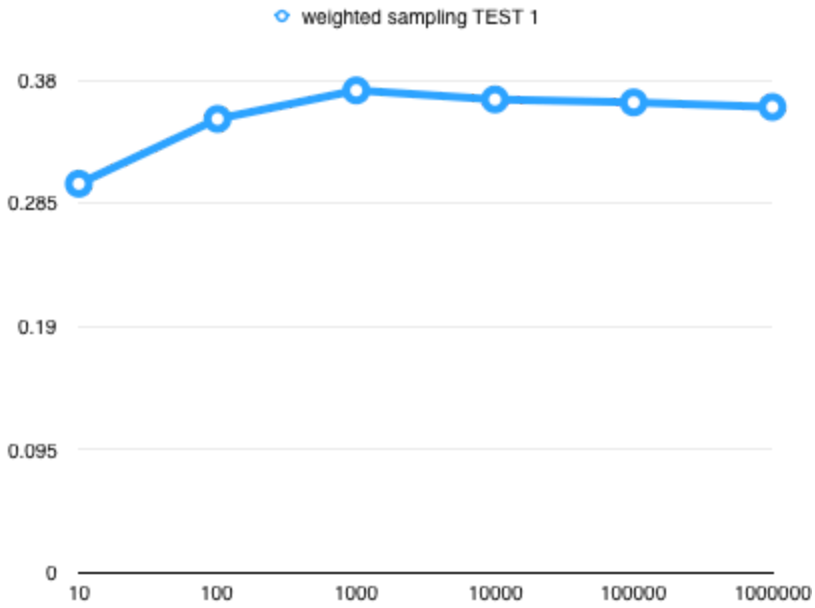
Question 4:

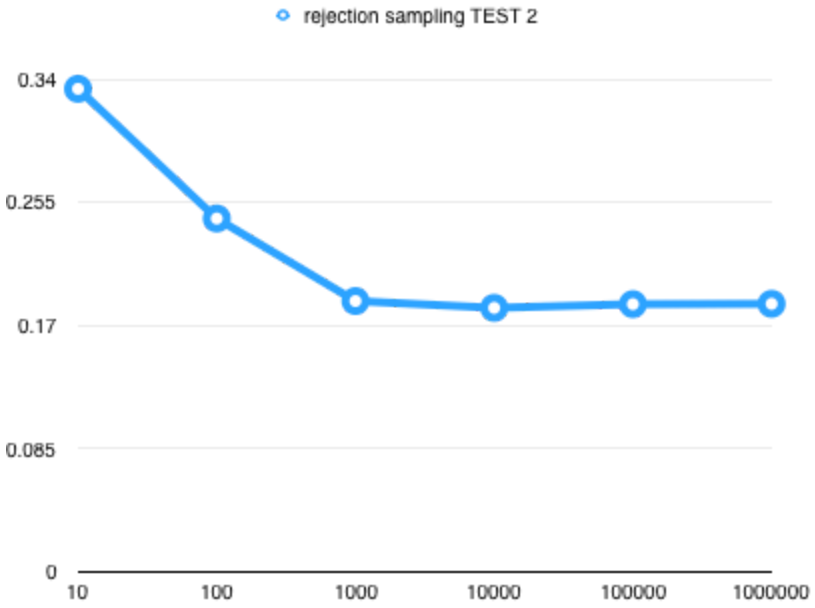
For the guilty network, compute the query of P(J = 1|B = 1, M = 0) and P(M = 1|J = 1), using the three sampling methods. Repeat 3(b) and 3(c).

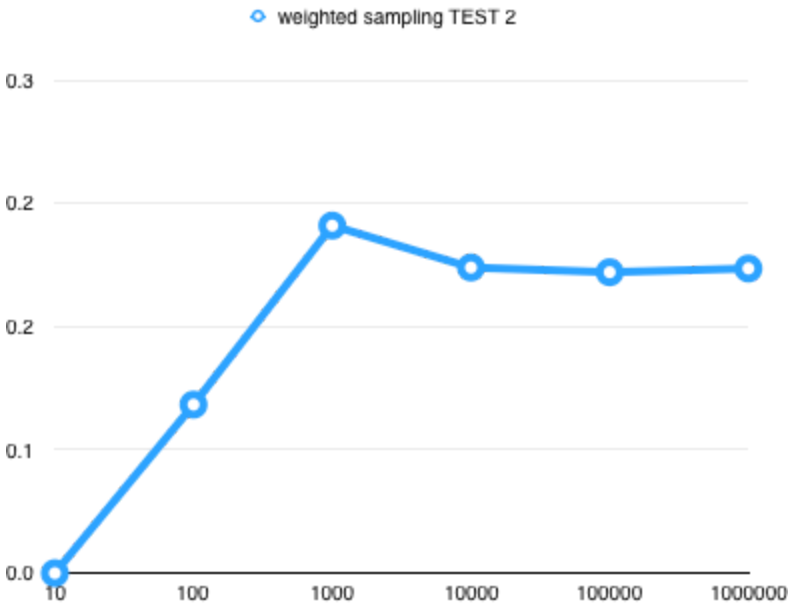
B. For each algorithm and each query, perform test with at least 5 different reasonably large values for the number of trials parameter, and graph how the estimated probabilities converge to the correct probability.

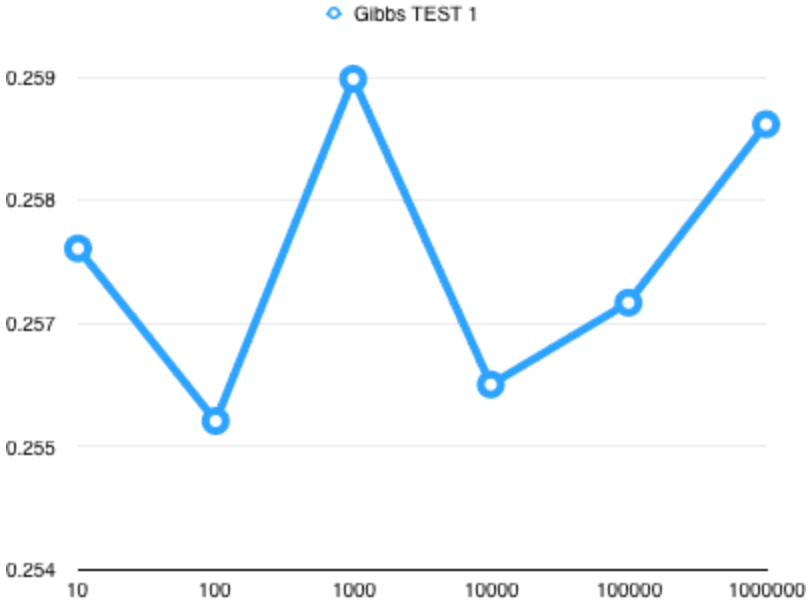
Runs	10	100	1000	10000	100000	1000000
rejection sampling TEST 1	0.222222222222	0.438202247191	0.366708385482	0.357568238213	0.363991446547	0.360469948711
weighted sampling TEST 1	0.3	0.35	0.372	0.365	0.362750000001	0.359159999991
Gibbs TEST 1	0.257262572626	0.255512555126	0.258982589826	0.255882558826	0.256712567126	0.258522585226
rejection sampling TEST 2	0.333333333333	0.243902439024	0.186991869919	0.182282793867	0.184829894032	0.185069230705
weighted sampling TEST 2	0.0	0.102564102564	0.211488250653	0.185993975904	0.183138656731	0.185395317544
Gibbs TEST 2	0.284972849728	0.283222832228	0.285832858329	0.286572865729	0.285242852429	0.283412834128

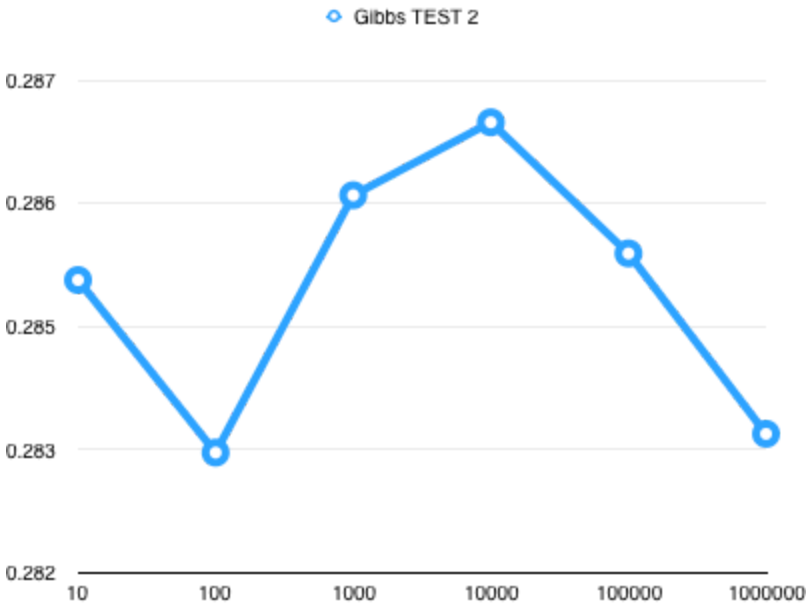



weighted sampling TEST 1
0.38
0.285
0.19
0.095
0
10
100
1000
10000
100000
1000000


rejection sampling TEST 2
0.34
0.255
0.17
0.085
0
10
100
1000
10000
100000
1000000


weighted sampling TEST 2
0.3
0.2
0.2
0.1
0.0
10
100
1000
10000
100000
1000000


Gibbs TEST 1
0.259
0.258
0.257
0.255
0.254
10
100
1000
10000
100000
1000000



How many trials do you need for each method to give an estimate that was reasonably close to the actual probability? How do these methods compare to each other? Discuss your findings.

d. Rejection Sampling: After about 1000 trials, it seems to converge on the probability for both Test 1 and Test 2.
e. Likelihood Weighting: Similar to Rejection Sampling it seems to converge on after about 1000 trials. However, compared to rejection sampling, the initial 10 to 1000 trial jumps between results is much larger.
f. Gibbs Sampling: The convergence for Gibbs seem to be better and less distanced between each runs like the first Bayesian Network. And similar to the first network, the jumps seem big because the differences are much smaller.

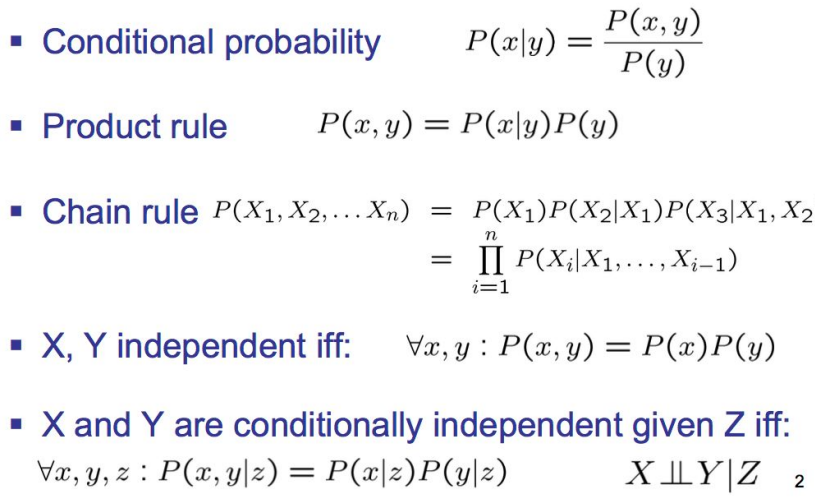
Question 5:

For the Gibbs sampling algorithm, you must compute P(Zi|mb(Zi)), where mb(Zi) denotes the values of the variables in Zi's Markov blanket. Perform the following:

A. From the textbook, we learned that in a Bayesian network, a variable is independent of all other variables in the network, given its Markov blanket (The variable is independent of all its non-descendents given its parents). So that means that you calculate the probabilities of the parents of the node's children, with the parents being given calculate the node probability. This is equal to
    a. $P(xi|x1,x2,...xi-1,xi+1,..,xn) = P(x1,x2,..,xn) / P(x1,x2,...xi-1,xi+1,..,xn)$.

    as being the Probability of *xi* given every other variable probabilities. You do this with calculating all the possible hidden variable probabilities and creating summations on

them. As an example, similar to how we did Question 2 in the first page using the



following rules:

B. For the P(x_i = True | mb(x_i)) value:
   1. I first ran P(x_i = True | Parents(x_i)) in a loop checking for parents and children of x_i and getting the probabilities of it.
   2. Then I multiplied the probability values of the *CapitalPI(*P(y_i = True | Parents(y_i))) by looping through the parents of children of x_i.
   3. The value of *probTrue* which is the multiplication of *CapitalPI(*P(y_i = True | Parents(y_i))) and P(x_i = True | Parents(x_i)).
   4. The value of *probFalse* is the same with P(x_i = False | Parents(x_i)).
   5. I return the value of (*probTrue* / (*probTrue* + *probFalse*)).

Question 6:

A paragraph from each author stating what their contribution was and what they learned.

- Sarah Haroon: I worked on most of the write up, getting probability of the first two questions, the tables, the graphs, the likelihood weighting. We helped each other with everything and were mainly pair programming and double checking.
- Woo-Jong Jang: Worked on Rejection Sampling, Prior Sampling, Topological Ordering, as well as Gibbs Sampling along with its helper methods.