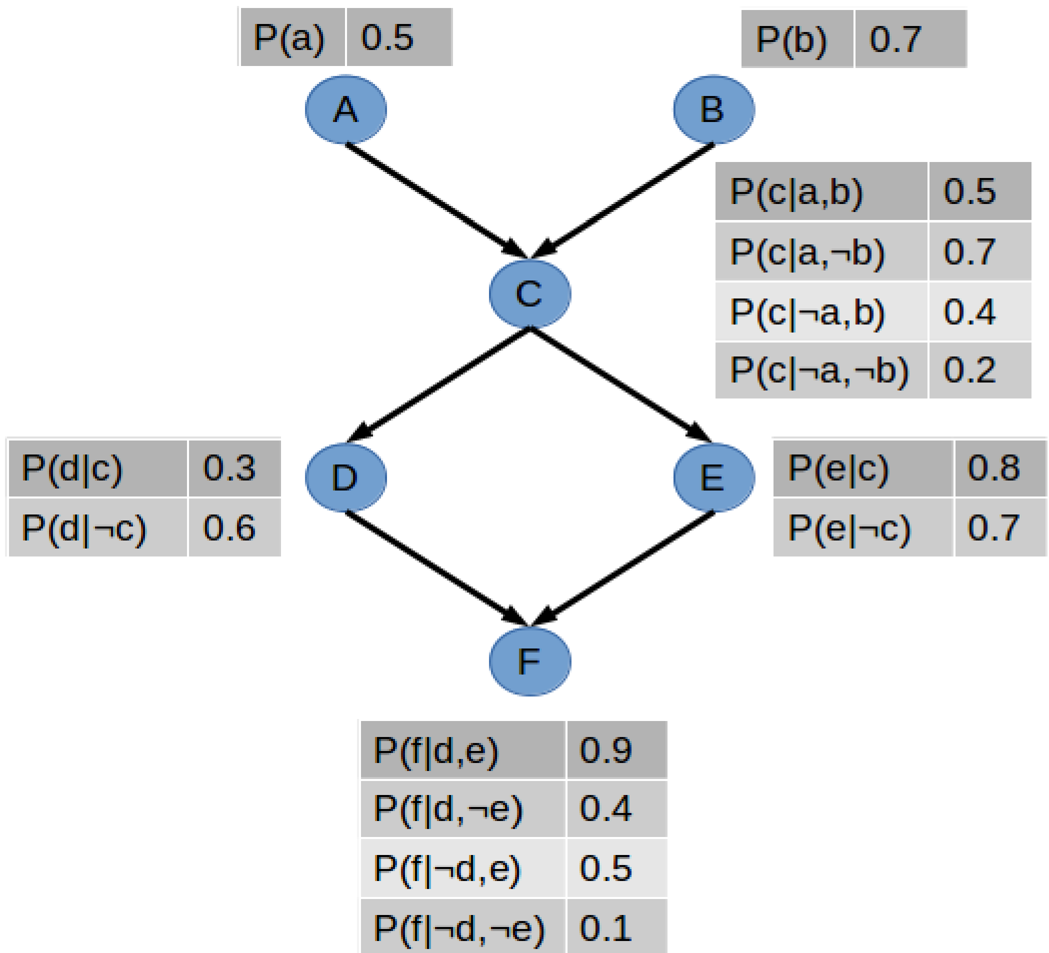
5512, Spring-2019
ASSIGNMENT 2:
Assigned: 02/19/19 Due: 03/03/19 at 11:55 PM (submit via Canvas, you may scan or take a picture of your paper answers) Submit only pdf or txt files (for non-code part), separate submission for code files
Show as much work as possible for all problems!

Problem 1. (20 points)
Use variable elimination on the Bayesian network below to find: P(d|¬b)



Solution:
P(d|¬b) = α Sum_{a,c,e,f}P(a,¬b,c,d,e,f)
P(d|¬b) =α Sum_{a,c,e,f}P(¬b) P(d|c) P(c|a,¬b) P(a) P(e|c) P(f|e)
P(d|¬b) =α P(¬b) Sum_c P(d|c) Sum_a P(c|a,¬b) P(a) Sum_e P(e|c) Sum_f P(f|e)
P(d|¬b) =α P(¬b) Sum_c P(d|c) Sum_a P(c|a,¬b) P(a) Sum_e P(e|c) * 1
P(d|¬b) =α P(¬b) Sum_c P(d|c) Sum_a P(c|a,¬b) P(a) *1
P(d|¬b) =α f1 Sum_c f2(c,d) Sum_a f3(c,a) f4(a)

f4(a) = 0.5
f4(¬a) = 0.5
f3(a,c) = 0.7

f3(a,¬c) = 0.3
f3(¬a,c) = 0.2
f3(¬a,¬c) = 0.8

Then....
f34(a,c) = f3(a,c)*f4(a) = 0.7*0.5 = 0.35
f34(a,¬c) = f3(a,¬c)*f4(a) = 0.3*0.5 = 0.15
f34(¬a,c) = f3(¬a,c)*f4(¬a) = 0.2*0.5 = 0.1
f34(¬a,¬c) = f3(¬a,¬c)*f4(¬a) = 0.8*0.5 = 0.4

P(d|¬b) =α f1 Sum_c f2(c,d) sum_a f34(a,c)

f34a(c) = f34(a,c) + f34(¬a,c) = 0.35+0.1 = 0.45
f34a(¬c) = f34(a,¬c) + f34(¬a,¬c) = 0.15 + 0.4 = 0.55

P(d|¬b) =α f1 Sum_c f2(c,d) f34a(c)

f2(c,d) = 0.3
f2(c,¬d) = 0.7
f2(¬c,d) = 0.6
f2(¬c,¬d) = 0.4

f34a2(c,d) = f2(c,d) * f34a(c) = 0.3*0.45 = 0.135
f34a2(c,¬d) = f2(c,¬d) * f34a(c) = 0.7*0.45 = 0.315
f34a2(¬c,d) = f2(¬c,d) * f34a(¬c) = 0.6*0.55 = 0.33
f34a2(¬c,¬d) = f2(¬c,¬d) * f34a(¬c) = 0.4*0.55 = 0.22

P(d|¬b) =α f1 Sum_c f34a2(c,d)

f34a2c(d) = f34a2(c,d) + f34a2(¬c,d) = 0.135 + 0.33 = 0.465
f34a2c(¬d) = f34a2(c,¬d) + f34a2(¬c,¬d) = 0.315 + 0.22 = 0.535

P(d|¬b) =α f1 f34a2c(d)
The rest are just constant factors, just normalize (which it actually is...)

P(d|¬b) = 0.465

Problem 2. (10 points)
While I said the accuracy of likelihood weighting increases as 1/sqrt(N) if N samples are taken, (Dagum, Karp, Luby, Ross) show that a better bound is (μ is the actual probability):

$$P\Big((1-\epsilon)\mu \le LikelihoodEstimate \le (1+\epsilon)\mu\Big) > 1-\delta$$
$$N \ge \frac{4}{\mu\epsilon^2}\ln\frac{2}{\delta}$$

Using this information, determine how many samples you would need to have a 95% confidence that you are within 1% of the actual answer? (Hint: what assumptions can you make to make sure you are not underestimating?)

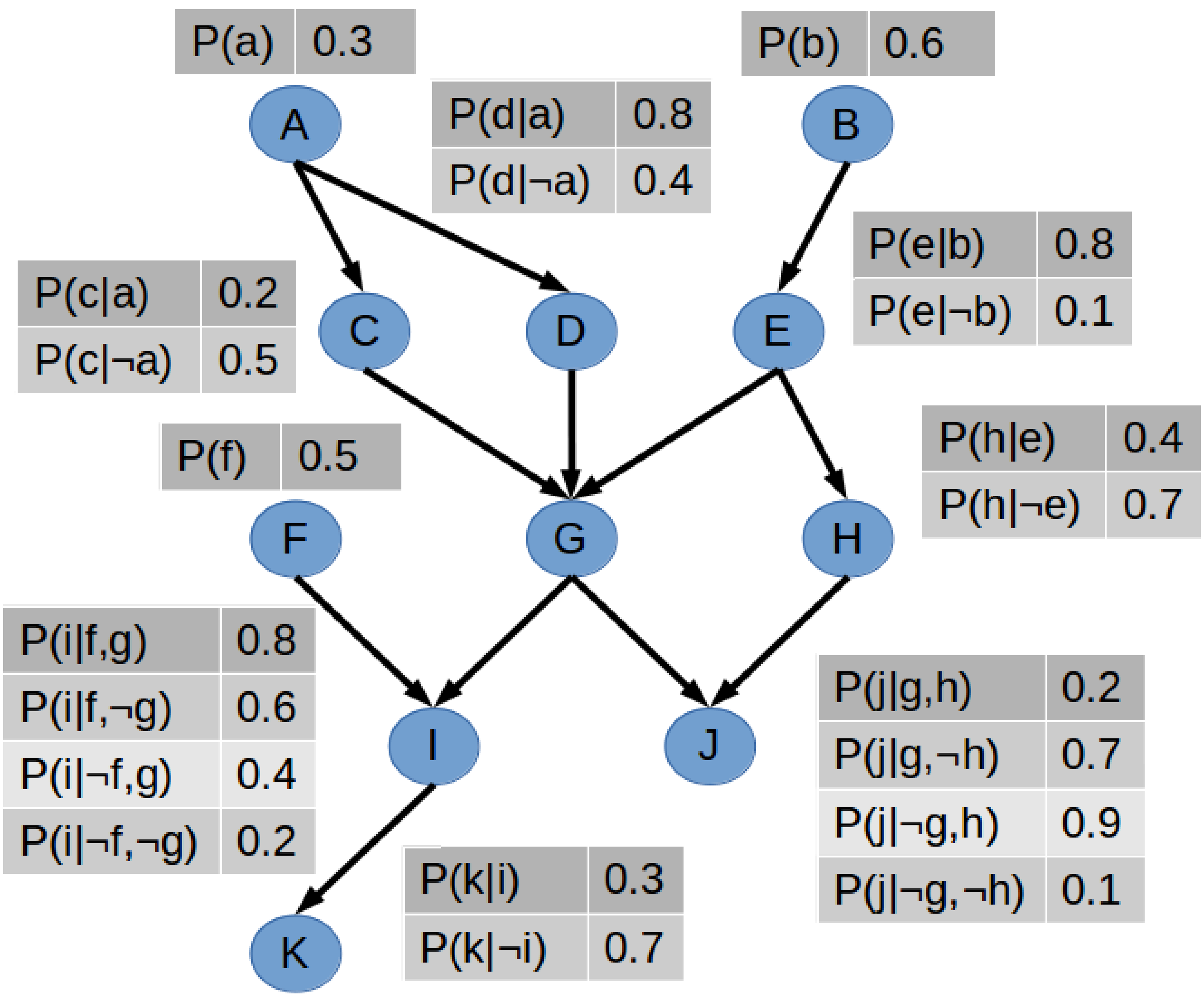
Solution:
Here we have: ε=0.01, δ=0.05
So plugging into the second equation should give:
N ≥ 147555.178164557 / mu

This is problematic for small mu's as this number tends to get very large (towards infinity as mu=0). This is because the first equation is defined with relative error (so the relative error when mu=0 is asking for perfect accuracy, which is not possible with an approximate method).

If we were using likelihood weighting to find the probability of a single variable, this can be avoided by finding:
P(a | stuff) with about 300,000 samples. If we find that P(a | stuff) < 0.5, we then go find P(¬a | stuff) as this should be over 0.5. And then use compute: P(a | stuff) = 1 - P(¬a | stuff)

We could also evaluate mu as we sample (i.e. use a while loop with the condition of N and mu instead of a for-loop)

Problems 3, 4 and 5 will use the following Bayesian network:



P(g|c,d,e)	0.1
P(g|c,d,¬e)	0.2
P(g|c,¬d,e)	0.3
P(g|c,¬d,¬e)	0.4
P(g|¬c,d,e)	0.5
P(g|¬c,d,¬e)	0.6
P(g|¬c,¬d,e)	0.7
P(g|¬c,¬d,¬e)	0.8

Problem 3. (25 points)
Use likelihood weighting to estimate P(g|k,¬b,c). Use an appropriate amount of samples, which will require you to write code. Submit your your code as a supplement. You have the options of Python (preferred), Matlab or Java.

Solution:
Should be about:
0.27

Problem 4. (25 points)
Use Gibbs sampling to re-estimate P(g|k,¬b,c). Again you have to use sufficient samples to be close enough to the correct answer (you will lose points if you are too far away). Submit your code as a supplement. You have the options of Python (preferred), Matlab or Java.

Solution:
Should be about:
0.27

Problem 5. (25 points)
For problem 5, you can use whatever method you want to find probabilities (though do say how you get them briefly).
(5.1) What is the Markov blanket of G?
(5.2) What is P(g| MarkovBlanket(G))? Assume all parts of the Markov Blanket are true (i.e. positive x, not ¬x).
(5.3) Find P(g|c,d,e,f). Then find P(g|c,d,e). Explain the relationship between these probabilities.
(5.4) Find P(g). Find P(g|f). Explain the relationship between these probabilities.
(5.5) What is the minimal set of given information (evidence) to make G conditionally independent from A?
(5.6) What is the minimal set of information (evidence) to make G conditionally independent from J?

Solution:
(5.1) C, D, E, F, G, I, J
(5.2)
P(g | c, d, e, f, g, i , j) = α P(g | c, d, e) * P(i | f, g) * P(j | g, h) = α 0.1 * 0.8 * 0.2 = 0.016
P(¬g | c, d, e, f, g, i , j) = α P(¬g | c, d, e) * P(i | f, ¬g) * P(j | ¬g, h) = α 0.9 * 0.6 * 0.9 = 0.486

Normalize:
P(g | c, d, e, f, g, i , j) = 0.016 / (0.016 + 0.486) = 0.03187251

(5.3) Using likelihood weighting:
P(g|c,d,e,f) = 0.10
P(g|c,d,e) = 0.10

This would indicate that G and F are conditionally independent given C, D, and E. This is, in fact, one of our rules. F is a non-descendant of G, so given G's parents (C, D, and E), then G and F are conditionally independent.

(5.4) Using likelihood weighting:
P(g) = 0.479
P(g|f) = 0.479

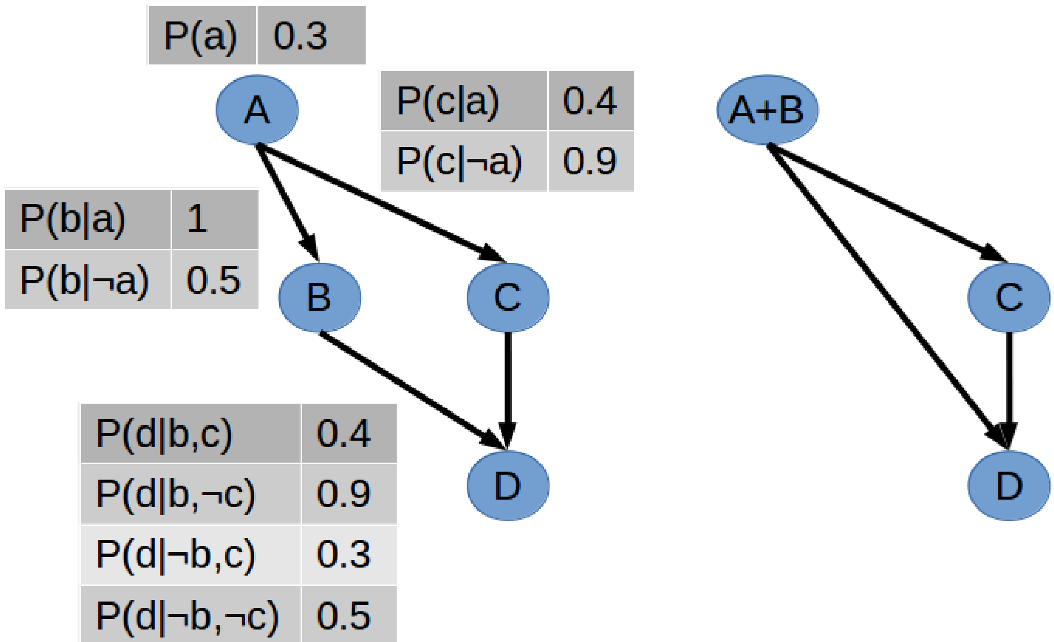
This would seem to indicate that G and F are (unconditional/normally) independent. The easiest way to see this from the Bayesian network is to think about how the network was constructed (like we did on homework 1). You are not too sure about the order of nodes, but you know node I was pick after both nodes G and F (since it has them both as parents). So this followed one of two order of picking nodes:

[some nodes], F, [more nodes] G, [more nodes] I
... or ...
[some nodes], G, [more nodes] F, [more nodes] I

In either case, when the second node between G or F was chosen, no edge was drawn between the two. In fact, F has no edges at all drawn into it, so when F was added it was independent of all nodes currently on the graph (so if in case 2, done) and remained independent for all later nodes (including G if in case 1) until node I was picked.

Thus regardless of case, F and G and independent.

Problem 6. (10 points)
Consider the Bayesian network below on the left. Suppose we want to merge/cluster variables/nodes A and B into the graph on the right. Provide probability tables for the three nodes on the right that will keep the probabilities of C and D the same as in the original Bayesian network. Then describe how you can deduce the values of A and B in the original Bayesian network from the value of A+B in the new Bayesian network.



Solution:
You could follow the book's way, where the node keeps track of four values and end up with:

P("A+B" = {T,T}) = 0.3
P("A+B" = {T,F}) = 0
P("A+B" = {F,T}) = 0.7*0.5
P("A+B" = {F,F}) = 0.7*0.5

Then you would get C as:
P(C | "A+B" = {T,T}) = 0.4
P(C | "A+B" = {T,F}) = 0.4
P(C | "A+B" = {F,T}) = 0.9
P(C | "A+B" = {F,F}) = 0.9

And D's table as:
P(D | "A+B" = {T,T}, c) = 0.4
P(D | "A+B" = {T,T}, ¬c) = 0.9
P(D | "A+B" = {T,F}, c) = 0.3
P(D | "A+B" = {T,F}, ¬c) = 0.5
P(D | "A+B" = {F,T}, c) = 0.4
P(D | "A+B" = {F,T}, ¬c) = 0.9
P(D | "A+B" = {F,F}, c) = 0.3
P(D | "A+B" = {F,F}, ¬c) = 0.5

While the above is technically correct, it is actually quite a bit more inefficient than the original graph (which is rather purpose defeating). The book/slides do this to help convert the problem into a polytree (for efficient computation), but here we still have a non-polytree. So the added computation is not beneficial. The book/slides are also a case where the variables are conditionally independent (given their shared parent), so you can't really just store one probability, as the probability of the other variable is not effected.

Instead, it would be best to just use the dependent relationship between the nodes to minimize the representation.

The easiest is to just have "A+B" be represented just by "A" in the original Bayesian network. So in this case, you would have:

Node "A+B": (same as A)
P("A+B" = 0.3)

Node C: (basically the same)
P(c | "A+B") = 0.4
P(c | ¬"A+B") = 0.9

Node D:
P(d | "A+B", c) = 0.4 // as "A+B" means A is true, which in turn means B is true
P(d | "A+B", ¬c) = 0.9
P(d | ¬"A+B", c) = 0.35 // as ¬"A+B" means A is false, so we only have 50/50 whether or not B is true, so we compute as: P(d | ¬"A+B", c) = sum_b P(b, d | ¬"A+B", c) = sum_b P(b | ¬"A+B", c) * P(d | b, ¬"A+B", c) = sum_b P(b | ¬"A+B") * P(d | b, c) = P(b | ¬"A+B") * P(d | b, c) + P(¬b | ¬"A+B") * P(d | ¬b, c) = 0.5*0.4 + 0.5*0.3 = 0.35
P(d | ¬"A+B", ¬c) = 0.7

It should be easily seen that P(C) is the same in the new/old network.
P(D) in old network:
sum_{a,b,c} P(a)P(b|a)P(c|a)P(d|b,c) = 0.3*1*0.4*0.4 + 0.3*1*0.6*0.9 + 0.3*0*0.4*0.3 + 0.3*0*0.6*0.5 + 0.7*0.5*0.9*0.4 + 0.7*0.5*0.1*0.9 + 0.7*0.5*0.9*0.3 + 0.7*0.5*0.1*0.5 = 0.4795

P(D) in new network:
sum_{"A+B", C} P("A+B") P(c | "A+B") P(d | "A+B", c) = 0.3*0.4*0.4 + 0.3*0.6*0.9 + 0.7*0.9*0.35 + 0.7*0.1*0.7 = 0.4795

From here you can "retrieve" the probability of node B by: (the table that was removed)
P(b | "A+B") = 1
P(b | ¬"A+B") = 0.5

And since node A is basically node "A+B":
P(a | "A+B") = 1
P(a | ¬"A+B") = 0

... A third way is that you could have "A+B" represent node B. First you would have to find P(b). P(b) = sum_a P(a,b) = sum_a P(a)*P(b|a) = 0.3*1 + 0.7*0.5 = 0.65.

Thus P("A+B") = 0.65
Since "A+B" is basically node B, the table for D would remain unchanged:
P(d | "A+B", c) = 0.4
P(d | "A+B", ¬c) = 0.9
P(d | ¬"A+B", c) = 0.3

$P(d \mid \neg$“A+B”$, \neg c) = 0.5$

However, the table for C, you would need to reverse-engineer the probability of A
$P(c \mid b) = \text{sum\_a } P(a,c \mid b) = \text{sum\_a } P(a|b) * P(c|a,b) = \alpha \text{ \textbackslash sum\_a } P(a)*P(b|a)*P(c|a) = \alpha\ (0.3*1*0.4 + 0.7*0.5*0.9) = \alpha\ \ 0.435$
$P(\neg c|b) = \ \alpha\ (0.3*1*0.6 + 0.7*0.5*0.1) = \alpha\ 0.215$
So after normalizing: $P(c \mid b) = 0.669230769 = P(c \mid$ “A+B”$)$

Then repeat for $\neg b$:
$P(c|\neg b) = \alpha\ (0.3*0*0.4 + 0.7*0.5*0.9) = \alpha\ 0.315$
$P(\neg c|\neg b) = \alpha\ (0.3*0*0.6 + 0.7*0.5*0.1) = \alpha\ 0.035$
So, $P(c|\neg b) = 0.9 = P(c \mid \neg$“A+B”$)$ //same as $P(c|\neg a)$, which shouldn’t be a surprise

Thus in this third network we could compute $P(d)$ as:
sum_{“A+B”,c} P(“A+B”)*P(c | “A+B”)*P(d | “A+B”, c) = 0.65*0.66923*0.4 + 0.65*(1-0.66923)*0.9 + 0.35*0.9*0.3 + 0.35*0.1*0.5 = 0.4795

To get node A from node “A+B” is:
$P(a \mid$ “A+B”$) = \ \alpha\ P(a)*P(b|a) = \alpha\ 0.3*1 = \alpha\ 0.3$
$P(\neg a \mid$ “A+B”$) = \alpha\ P(\neg a)*P(b|\neg a) = \alpha\ 0.7*0.5 = \alpha\ 0.35$
Normalize: $P(a \mid$ “A+B”$) = 0.461538462$

$P(a \mid \neg$“A+B”$) = \alpha\ P(a)*P(\neg b|a) = \alpha\ 0.3*0 = \alpha\ 0 = 0$ (after normalization of whatever)