Problem 2 Bayes Nets (31 points)

This problem contains independent parts, so **do not give up on the problem as a whole just because you give up on part of the problem**.

You are given the task of making admissions decisions to MIT. To this end you interview the admissions staff and you conclude that the process can be modeled with four binary-valued random variables:

- Q – indicates whether the applicant is qualified for admission, that is, would inherently be able to do well at MIT.
- G – indicates whether the applicant has a high grade point average in high school.
- R – indicates whether the applicant has excellent recommendations from high school teachers.
- A – indicates whether the applicant is admitted.

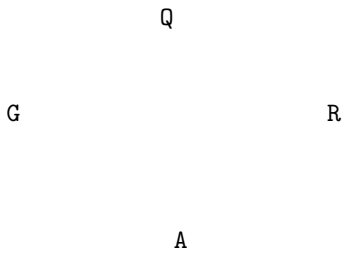
You also get the following information from your interviews.

- One third of all applicants are qualified.
- 95% of qualified applicants have a high grade point average. On the other hand, 50% of unqualified applicants also have a high grade point average.
- 90% of qualified applicants have excellent recommendations. On the other hand, 75% of unqualified applicants also have excellent recommendations.
- Applicants that have a high grade point average and excellent recommendations have a 75% chance of being admitted. If an applicant only has a high grade point average or excellent recommendations but not both, then there is a 50% chance of getting admitted. If the applicant has neither a high grade point average nor excellent recommendations, there is no chance they'll get admitted.

Part A (9 points)

We will now construct a Bayes net that captures the data above.

First, in the space below draw *only* the necessary links between the nodes. Do not fill in the probabilities on this picture; we will deal with them later.



Now, specify the conditional probability tables for the nodes. In the top row of each table, identify the variables on which the conditional probability depends. Then, use the other rows to show the conditional probability values for each combination of variable values.

We've given you room to write the biggest possible table, you need not use every entry in these tables.

Var Name	Var Name	Var Name	Cond Prob
			P(Q | __,__,__)

Var Name	Var Name	Var Name	Cond Prob
			P(G | __,__,__)

Var Name	Var Name	Var Name	Cond Prob
			P(R | __,__,__)

Var Name	Var Name	Var Name	Cond Prob
			P(A | __,__,__)

Part B (8 points)

Could the conditional probability table for node A (as specified above) have been written, equivalently, using the noisy-or model? If yes, clearly indicate the causal probabilities and leak probability. If no, explain in (at most) two sentences why it cannot be done.

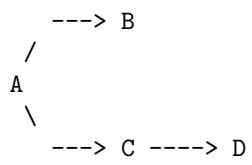
Check YES or NO: ☐ YES ☐ NO

For "yes," give causal probabilities and leak; for "no," give explanation:

Part C (6 points)

The following problems are independent of the ones above.

Given the following Bayes Net:



Which of the following is the simplest correct expression for the probability of a fully specified state? Circle your choice.

1. P(A=a ,B=b,C=c,D=d) = P(D=d| A=a,B=b,C=c) P(B=b| C=c,A=a) P(C=c| A=a) P(A=a)
2. P(A=a,B=b,C=c,D=d) = P(D=d| A=a,B=b,C=c) P(B=b| A=a) P(C=c| A=a) P(A=a)
3. P(A=a,B=b,C=c,D=d) = P(D=d| B=b,C=c) P(C=c| B=b,A=a) P(B=b| A=a) P(A=a)
4. P(A=a,B=b,C=c,D=d) = P(D=d| A=a,B=b,C=c) P(C=c| A=a) P(B=b| A=a) P(A=a)
5. P(A=a,B=b,C=c,D=d) = P(D=d| B=b,C=c) P(B=b| C=c,A=a) P(C=c| A=a) P(A=a)
6. P(A=a,B=b,C=c,D=d) = P(D=d| B=b,C=c) P(B=b| A=a) P(C=c| A=a) P(A=a)
7. P(A=a,B=b,C=c,D=d) = P(D=d| C=c) P(B=b| C=c,A=a) P(C=c| A=a) P(A=a)
8. P(A=a,B=b,C=c,D=d) = P(D=d| C=c) P(B=b| A=a) P(C=c| A=a) P(A=a)
9. P(A=a,B=b,C=c,D=d) = P(D=d| C=c) P(B=b) P(C=c| A=a) P(A=a)
10. P(A=a,B=b,C=c,D=d) = P(D=d| C=c) P(B=b) P(C=c) P(A=a)
11. P(A=a,B=b,C=c,D=d) = P(D=d) P(B=b) P(C=c) P(A=a)

Part D (8 points)

Assume all the variables in the net above are binary valued. Give a symbolic expression for computing P(B=T,D=F | A=T,C=F). Write the expression completely in terms of terms of the form P(A=a,B=b,C=c,D=d), where a,b,c,d are either T or F. **USE THE NUMBER FOR EACH OF THESE TERMS DRAWN FROM THE TABLE BELOW**:

1	P(A=T,B=T,C=T,D=T)	9	P(A=F,B=T,C=T,D=T)
2	P(A=T,B=T,C=T,D=F)	10	P(A=F,B=T,C=T,D=F)
3	P(A=T,B=T,C=F,D=T)	11	P(A=F,B=T,C=F,D=T)
4	P(A=T,B=T,C=F,D=F)	12	P(A=F,B=T,C=F,D=F)
5	P(A=T,B=F,C=T,D=T)	13	P(A=F,B=F,C=T,D=T)
6	P(A=T,B=F,C=T,D=F)	14	P(A=F,B=F,C=T,D=F)
7	P(A=T,B=F,C=F,D=T)	15	P(A=F,B=F,C=F,D=T)
8	P(A=T,B=F,C=F,D=F)	16	P(A=F,B=F,C=F,D=F)

Your answer should involve `ONLY` arithmetic symbols and integers between 1 and 16, as in $\frac{4+12}{7\times 2}$, for example.

Please write neatly!

Problem 4 How they work (23 points)

The purpose of this problem is to test your understanding of various mechanisms at the level required to predict performance in the face of various perturbations.

This problem contains independent parts, so **do not give up on the problem as a whole just because you give up on part of the problem**.

Part A Neural Nets

A.1 (5 points)

You have purchased neural-net training software from MicroMonopoly. Not surprisingly, it is screwed up.

Instead of the *standard* threshold function,

$$l = \frac{1}{1 + e^{-m}}$$

they have provided an *altered* function,

$$l = \frac{1}{1 + e^{+m}}$$

which, when plotted, gives you the mirror image of the *standard* threshold function.

You are told that a network with the *standard* threshold function can be trained to perform a certain recognition task successfully with a given training set.

After a bit of thought, you conclude that there is a set of weights that will enable an identically arranged network to perform the recognition task with the *altered* threshold function.

Next, you learn that the MicroMonopoly training software uses exactly the training formulas given in the book:

$$\begin{aligned} \Delta w_{ij} &= r o_i o_j (1 - o_j) \beta_j, \\ \beta_j &= \textstyle\sum_k w_{jk} o_k (1 - o_k) \beta_k \text{ for nodes in hidden layers,} \\ \beta_z &= d_z - o_z \text{ for nodes in the output layer.} \end{aligned}$$

Check the box next to the best expectation; place an X in the other boxes:

☐ The training software won't work; in principle, networks that use the *altered* threshold function can't be trained using the given training formulas.

☐ The training software may work, but, on the other hand, using initial weights and a rate constant that would be appropriate for the correct threshold function, it may take much longer to train the network, and the training may not converge on a suitable set of weights at all.

☐ Using the initial weights and a rate constant that would be appropriate for the correct threshold function, the software will work just about as it would if there were no bug.

A.2 (5 points)

You call customer "support" and the person who you talk to recommends that you save all files, reformat your disk, and then reload the operating system and all your applications.

Fortunately, before you descend into that hell, "Network 98" is released. Now, the threshold function they provide is:

$$l = \frac{1}{1 + e^{-1{,}000{,}000 \times m}}$$

Check the box next to the best expectation; place an X in the other boxes:

☐ The training software won't work; in principle, networks that use the *altered* threshold function can't be trained using the given training formulas.

☐ The training software may work, but, on the other hand, using initial weights and a rate constant that would be appropriate for the correct threshold function, it may take much longer to train the network, and the training may not converge on a suitable set of weights at all.

☐ Using the initial weights and a rate constant that would be appropriate for the correct threshold function, the software will work just about as it would if there were no bug.

A.3 (5 points)

Finally, you do a work around that enables you to train the network with the following, normal, threshold function:

$$l = \frac{1}{1 + e^{-m}}$$

Your trained network is to be used in a handheld computer game, so you hand the training results over to a circuit designer who burns the network into a VLSI chip.

Unfortunately, the designer works for a MicroMonopoly spinoff. The threshold functions burned into the chip reflect a previous release of MicroMonopoly Network 98:

$$l = \frac{1}{1 + e^{-1000m}}$$

Even more unfortunately, you learn it would cost a fortune to alter most of the characteristics of the chip, but a few things can be done inexpensively.

Check the box next to the right thing to do; place an X in the other boxes:

☐ Multiply all inputs to first-layer nodes by 10^3.

☐ Multiply all weights by 10^3.

☐ Multiply all final-layer outputs by 10^3.

☐ Multiply all inputs to first-layer nodes by 10^{-3}.

☐ Multiply all weights by 10^{-3}.

☐ Multiply all final-layer outputs by 10^{-3}.

☐ None of the above.