Problem 1: [20 points] **

P(A) P(B) P(E)

a	$p(a)$
0	0.3
1	0.7

b	$p(b)$
0	0.6
1	0.4

e	$p(e)$
0	0.7
1	0.3

P(H|D)
P(H|E)
P(F|C)
P(C|A)

y	x	$p(x\|y)$
0	0	0.10
0	1	0.90
1	0	0.30
1	1	0.70

z	y	x	$p(x\|y,z)$
0	0	0	0.25
0	0	1	0.75
0	1	0	0.60
0	1	1	0.40
1	0	0	0.10
1	0	1	0.90
1	1	0	0.20
1	1	1	0.80

P(D|A,B)
P(G|C,D)
P(H|D,E)

Figure 3: Conditional probability tables

The question investigates the AND/OR search space of the network given in Figure 1 assuming that each variable is binary. The CPTs are given in Figure 3. The CPTs for G, H and D are identical to the 3-dimensional CPT in Figure 3 and the CPTs for H and F are identical to the 2-dimensional CPT in the same figure

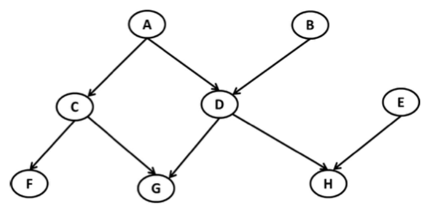
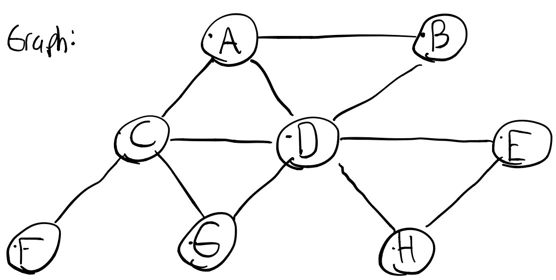


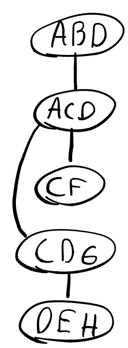
Figure 1:

- Find and present a pseudo tree of the network whose depth is minimal. Call it T_1. Do the best you can.

Moralize Graph:



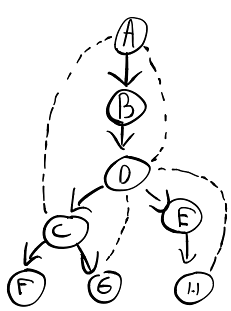
Tree decomposition



Treewidth = 2

min-pseudotree depth $\leq 2 \cdot \lg 8$
$\leq 2 \cdot 3$
≤ 6

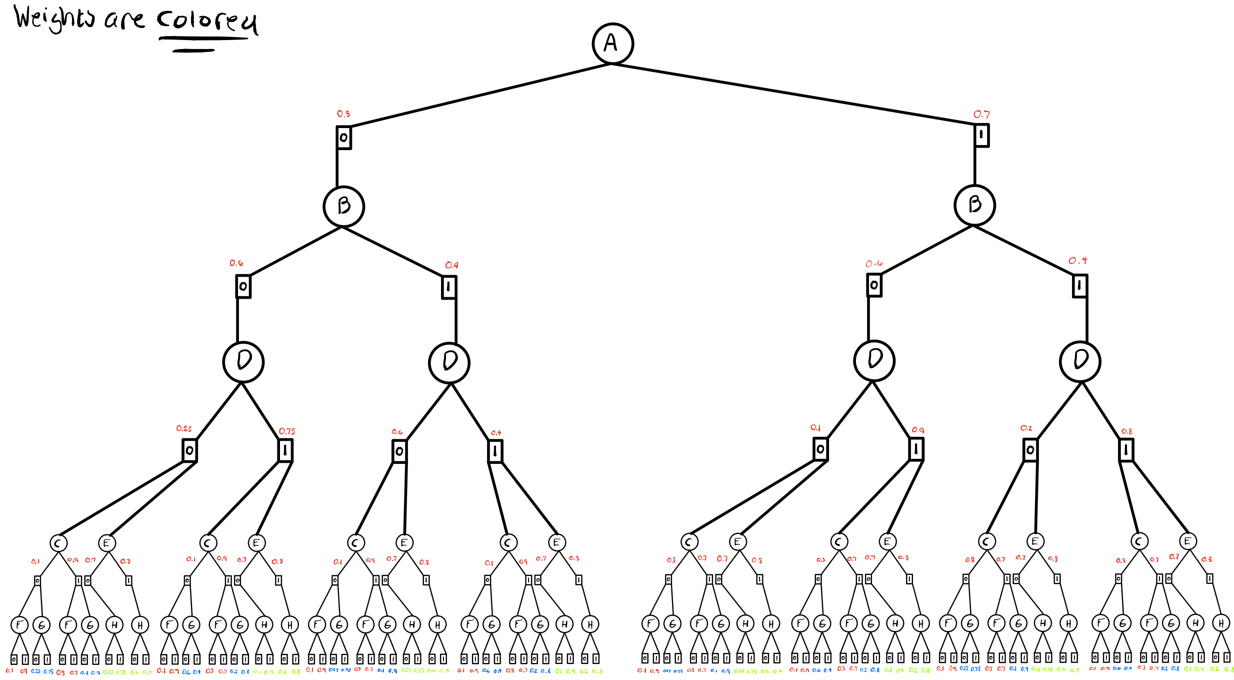
Pseudo-Tree:



$\boxed{\text{pseudo-depth} = 4}$

- Generate an AND/OR search tree driven by T_1 assuming that each variable has at most two values.
- Annotate the arcs with appropriate weights

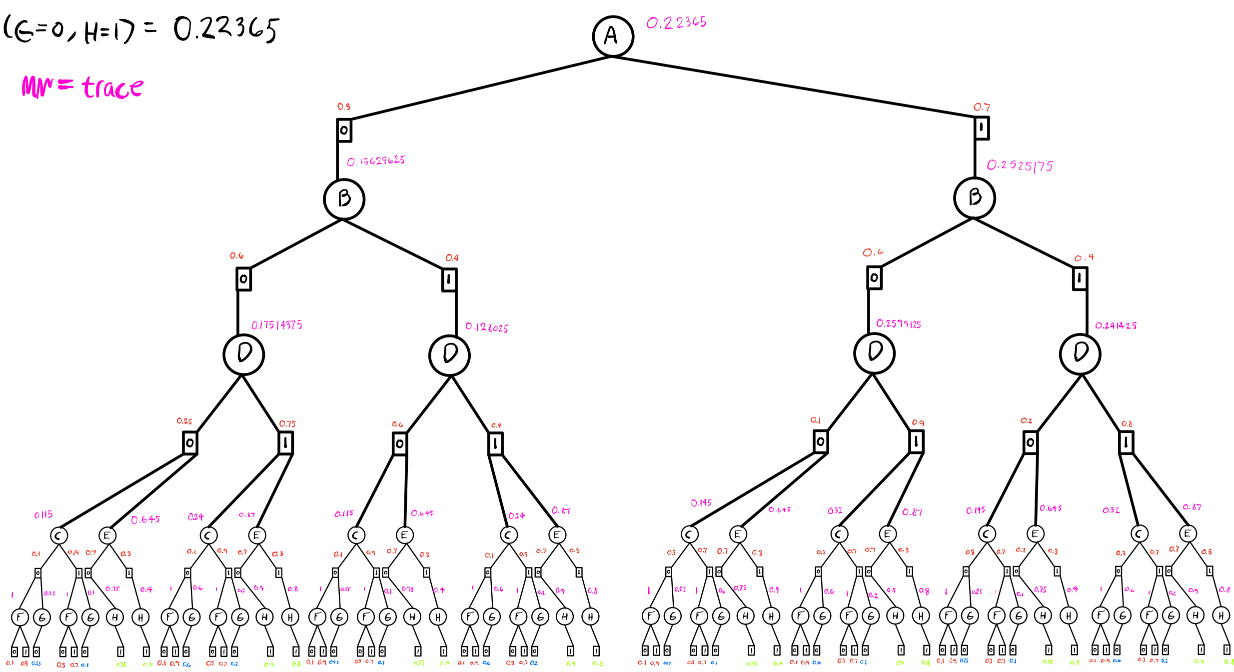
Weights are colored



- What is the computational cost of computing the probability of evidence $G = 0$ and $H = 1$ in such a network if you use depth-first search over the AND/OR search tree. Demonstrate your computation.

$P(G=0, H=1) = 0.22365$

MN = trace



# of primitive operations $= 48+32+16+8+4+4+2+2+1$
$= 117$, upper bounded by $O(8 \cdot 2^4)$

time complexity $T(n) = O(8 \cdot 2^4)$ where $8 = $ # of nodes
$2 = $ domain size
$4 = $ pseudo-tree depth

- Can the AND/OR search tree be reduced to a smaller AND/OR search graph.

Yes, some modifications can be done to optimize our graph

For example, we can merge our root identical nodes E when D=0 and D=1 to merge to reduce our search space.

- Assume that the CPT $P(x|y,z)$ is changed by making some entries deterministic as follows: the first two probabilities are changed to 1 and 0 respectively. Similarly, the last two probabilities are changed to 1 and 0 respectively. Show what would be the changes in the AND/OR search tree as a result.

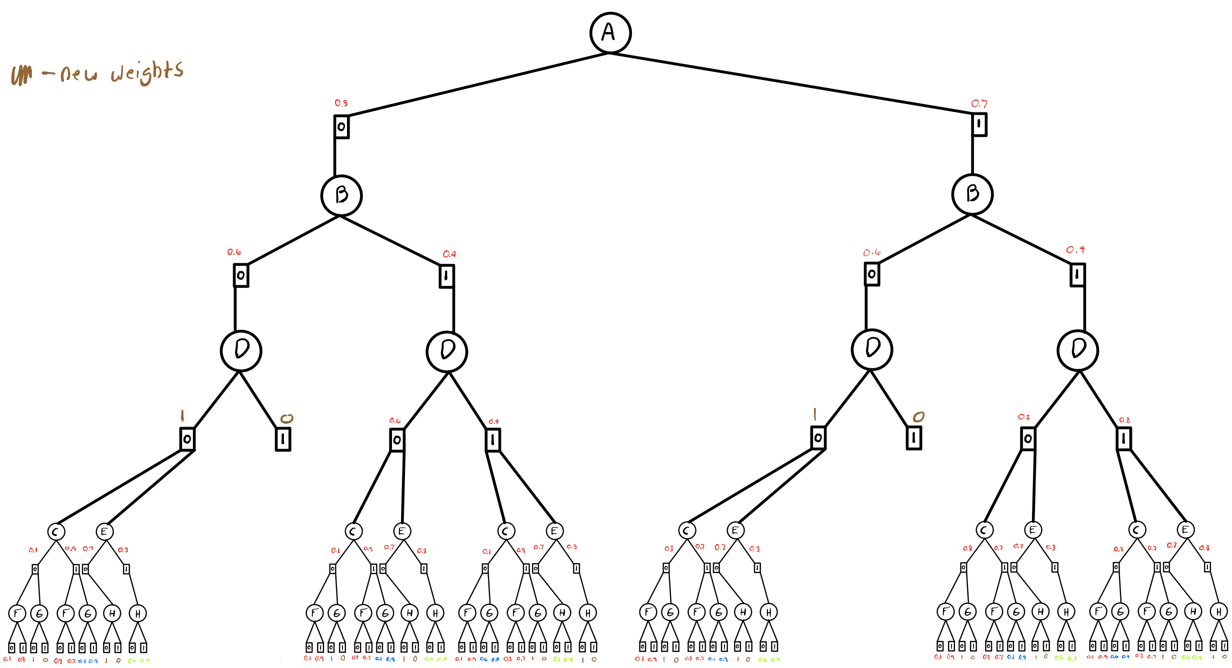
$P(x|y,z) =$

z	y	x	$p(x\|y,z)$
0	0	0	1
0	0	1	0
0	1	0	0.60
0	1	1	0.40
1	0	0	0.10
1	0	1	0.90
1	1	0	1
1	1	1	0

$P(D|A,B)$

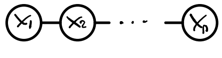
$P(G|C,D)$

$P(H|D,E)$



As seen above, our most useful optimization ended up being $P(D|A,B)$ because it had many subnodes beneath it that could be removed.

Part 2: Inference [10 points] Consider a chain Markov network $X_1 - X_2 - X_3 - \ldots - X_n$. Provide an optimal algorithm which calculates $\Pr(X_i, X_j)$ for **all pairs** $i \neq j$. Prove its optimality.



$$P(X_1 = x_1, X_2 = x_2) = \frac{1}{Z} \sum \phi(X_1, X_2, X_3) \ldots \phi(n-2, n-1, n)$$

Brute Force: Var elim on each pair $\quad w^* = 2$

$d^2 n^2$ pairs, var elim is $n d^{w^*-1}$ $\quad d =$ domain size (maximum)

Naive Var Elim is $O(n^3 d^3)$ in complexity

Idea: Iterate over bucket elimination as usual.

Starting at first variable in ordering, compute partition function for $(X_1=0, X_2=0), \dots (X_1=1, X_n=1)$

Save information of $\sum_{X_1} \phi$ to use for X_2 (no evidence)

Compute with evidence $(X_2=0, X_3=0), \dots, (X_2=1, X_n=1)$

Save information of $\sum_{X_2} \varphi \sum_{X_1} \phi$ to use for X_3 (no evidence)

Repeat until $i = X_n$

return all probabilities

```
VarElim(F-factors, prev_info):
    ordering : S_start, ..., S_n
    for o in ordering:
        phi ← factors in F with variable o
        F ← factors in F without variable o
        R ← product of everything in phi
        if first iteration:
            R ← R · prev_info
        F.append(R.sumVariable(o))
    return product of F
```

```
AllPairsVarElim(F-factors):
        ordering ← S_1, S_2, ..., S_n
        prev-info ← empty factor (equivalently 1)
        L ← empty list
n {     for o in ordering:
d {         for d in F_o.cardinality:
n-1 {           for each variable v after o:
d                   for d_2 in F_v.cardinality:
                        F_n ← F instantiated with o=d, v=d_2
                        L.append(VarElim(F_n , prev-info))
            prev_info ← VarElim(F, prev_info)
          remove S_o from F
        return L
```

$T(n) = (n-1)(n-2)d^3 + (n-2)(n-3)d^3 + \dots + 2d^3 = O(n^2 d^3)$

And Brute force is $O(n^3 d^3)$

Therefore, this algorithm is optimal.

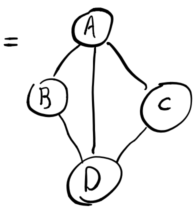
Exercise 12.9

In this question, we consider the application of importance sampling to Markov networks.

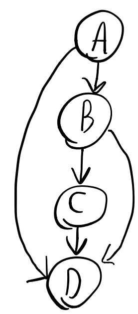
a. Explain intuitively why we cannot simply apply likelihood weighting to Markov networks.
b. Show how likelihood weighting can be applied to chordal Markov networks. Is this approach interesting? Explain.
c. Provide a technique by which the more general framework of importance sampling can be applied to Markov networks. Be sure to define both a reasonable proposal distribution and an algorithmic technique for computing the weights.

a) The issue with markov networks is that in Bayesian networks we can start at the root variables, then use those sample from the children and so on. In Markov Networks, this is not as simple because when we take variables we do not have conditional probabilities and distributions between 0 and 1. This makes it not intuitive to sample from.

b) If we have a chordal markov network $G =$



Bayesian N: ordering: A, B, C, D



Step 1: convert Markov network to Bayes net.

⇓ transform factors into CPTs

Step 2: sample same as Bayesian network now

This is useful because we can avoid too much complexity and simply use importance sampling on Bayesian Networks

c) For this situation, we can use an adaptive proposal distribution That is, simply generate your variables uniformly, and we can adjust our distribution through a function which weights our samples based on its duplicates. So if we generate two samples and they are the same, then they will be weighted to adjust so we don't use two duplicate samples.

Exercise 12.13

Show directly from equation (12.21) (without using the detailed balance equation) that the posterior distribution $P(\mathcal{X} \mid \boldsymbol{e})$ is a stationary distribution of the Gibbs chain (equation (12.22)).

X_{-k} = All variables except X_k $\qquad$ $\mathcal{X}$ = all Variables

Gibbs Chain = $P(\mathcal{X}' \mid e) = T((X_{-k}, X_k) \to (X_{-k}, X_k'))$

$$= \frac{1}{|\mathcal{X}|} \sum_{k \in \mathcal{X}} \sum_{x_k \in Val(X_k)} P(x_k, X_{-k} \mid e) P(x_k' \mid X_{-k}, e) \cdot \frac{P(X_{-k} \mid e)}{P(X_{-k} \mid e)}$$

$X_{-k}, e = X_{-k}$

$$= \frac{1}{|\mathcal{X}|} \sum_{k \in \mathcal{X}} \sum_{x_k \in Val(X_k)} P(x_k \mid X_{-k}) P(x_k' X_{-k} \mid e)$$

$$= P(X' \mid e) \underbrace{\frac{1}{|\mathcal{X}|} \sum_{k \in \mathcal{X}} \sum_{x_k \in Val(X_k)} P(x_k \mid X_{-k})}_{\text{sums to } 1}$$

$$= P(X' \mid e)$$

Which satisfies the inequality

Thus, $P(X \mid e)$ is a stationary distribution

Part 5: [**10 points**] In class, we saw that the Metropolis-Hastings algorithm converges to a stationary distribution π if it satisfies the following detailed balance condition:

$$\pi(\mathbf{x})\mathcal{T}(\mathbf{x} \to \mathbf{x}')A(\mathbf{x} \to \mathbf{x}') = \pi(\mathbf{x}')\mathcal{T}(\mathbf{x}' \to \mathbf{x})A(\mathbf{x}' \to \mathbf{x})$$

Show that if we use

$$A(\mathbf{x} \to \mathbf{x}') = \frac{\pi(\mathbf{x}')T(\mathbf{x}' \to \mathbf{x})}{\pi(\mathbf{x})\mathcal{T}(\mathbf{x} \to \mathbf{x}') + \pi(\mathbf{x}')\mathcal{T}(\mathbf{x}' \to \mathbf{x})}$$

Thus, $A(x' \to x) = \dfrac{\pi(x)T(x \to x')}{\pi(x')T(x' \to x) + \pi(x)T(x \to x')}$

then the detailed balance condition is satisfied.

$$\frac{\pi(x)T(x \to x')\ \pi(x')T(x' \to x)}{\pi(x)T(x \to x') + \pi(x')T(x' \to x)}$$

$$= \pi(x')T(x' \to x)\left(\frac{\pi(x)T(x \to x')}{\pi(x)T(x \to x') + \pi(x')T(x' \to x)}\right)$$

$$= \pi(x')T(x' \to x)\ A(x' \to x)$$

Which satisfies the equality. Thus, metropolis-hastings algorithm converges to a stationary distribution under these assumptions.