Problem 1. Variable Elimination

1. $P(S=T|C=T) = \sum_{W,O,M} P(W)P(O)P(C=T)P(M)P(F|W,O)$
$P(G|C=T,M)P(B|F,G)P(S=T|F,B)$

↓ Eliminate W by $\sum_W P(W)P(F|W,O)$

$g_1(F,O) =$

O \ F	T	F
T	0.3×0.9 + 0.7×0.8 = 0.83	0.3×0.1 + 0.7×0.2 = 0.17
F	0.3×0.6 + 0.7×0.5 = 0.53	0.3×0.4 + 0.7×0.5 = 0.47

↓ Eliminate O by $\sum_O P(O)g_1(F,O)$

$g_2(F) =$

T	F
0.6×0.83 + 0.4×0.53 = 0.71	0.29

↓ Eliminate M by $\sum_M P(M)P(G|C=T,M)$

$g_3(G) =$

T	F
0.82	0.18

↓ Eliminate G by $\sum_G g_3(G)P(B|F,G)$

$g_4(B,F) =$

F \ B	T	F
T	0.864	0.136
F	0.651	0.349

↓ Eliminate F by $\sum_F g_4(B,F)\, g_2(F)\, P(S=T|F,B)$

$g_5(B) =$

T	F
0.714921	0.068057

↓ Eliminate B by $\sum_B g_5(B) = 0.714921 + 0.068057$

$= 0.782978$

Thus, $P(S=T|C=T) = 0.782978$

2、$P(F=T|G=T)$

There is no active trail between F and G when B is not observed, so, $P(F=T|G=T) = P(F=T) =$

$$\sum_{W,O} P(F=T|W,O)\, P(W)\, P(O) = 0.71$$

3、$P(M=T|G=T)$

In this case, we only need to consider C, M, G, since they are d-separated from other nodes.

$$\therefore P(M=T|G=T) = \frac{P(M=T, G=T)}{P(G=T)}$$

$$P(G=T) = \sum_{M,C} P(G=T|M,C) = \sum_M \sum_C P(G=T|M,C)$$

$$= \sum_M \left[0.4 \times P(G=T|M, C=T) + 0.6 \times P(G=T|M, C=F)\right]$$

$$P(M=T, G=T) = \underbrace{[0.4\times 0.1 + 0.6\times 0.01]}_{P(M=T)} = 0.046\times 0.1$$

$$P(M=F, G=T) = \underbrace{[0.4\times 0.9 + 0.6\times 0.]}_{P(M=F)} = 0.78\times 0.9$$

$$P(M=T \mid G=T) = \frac{0.046\times 0.1}{0.046\times 0.1 + 0.78\times 0.9} = \boxed{0.00651}$$

4. $P(M=T \mid G=T, S=T)$

since S is d-separated from the cluster G, C, M, the $P(M=T \mid G=T, S=T) = P(M=T \mid G=T) = \boxed{0.00651}$

5. $P(W=T \mid G=T, B=F, S=T)$

$= \sum P(W)P(O)P(F \mid W,O)P(S \mid F,B)P(B \mid F,G)P(G)$

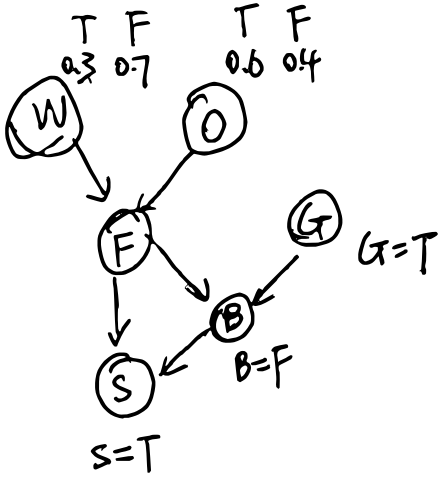
where $W=T, S=T, B=F, G=T$, so we only need sum over O, F

Also, C, M, H are ignored since summing over them gives an 1 in the final.

Hence, similarly, eliminating O, $g_1 = \sum_O P(O)P(F \mid W,O)$

and eliminating F, $g_2 = \sum_F g_1(F,W)P(S \mid F,B)P(B \mid F,G)$

Or, since $P(W=T \mid G=T, B=F, S=T) = \frac{P(W=T, G=T, B=F, S=T)}{P(G=T, B=F, S=T)}$



$P(S=T, B=F, G=T)$

$= \sum_{W,O,F,G,M} P(W)P(O)P(F|W,O)\underbrace{P(C)P(M)P(G|C,M)}_{A}P(B|F,G)P(S|F,B)$

$\sum_{C,M} P(C)P(M)P(G|C,M) = \text{constant} = A$

$P(S=T, B=F, G=T)$

$= \sum_{W,O,F} P(W)\underbrace{P(O)P(F|W,O)}P(B|F,G)P(S|F,B) \cdot A.$

↓ Eliminating O. $\quad \sum_{O} P(O)P(F|W,O) = g_1(F,W)$

W \ F	T	F
T	0.78	0.22
F	0.68	0.32

↓ Eliminating F. $\quad \sum_{F} g_1(F,W)P(B|F,G)P(S|F,B) = g_2(W)$

W	T	F
	0.0523	0.0488

$P(S=T, B=F, G=T)$

$$= \sum_{w} P(w)\, g_z(w)$$

Thus, answer $=$

$$\frac{0.3 \times 0.0523}{0.3 \times 0.0523 + 0.7 \times 0.0488}$$

$$= \frac{0.01569}{0.01569 + 0.03416}$$

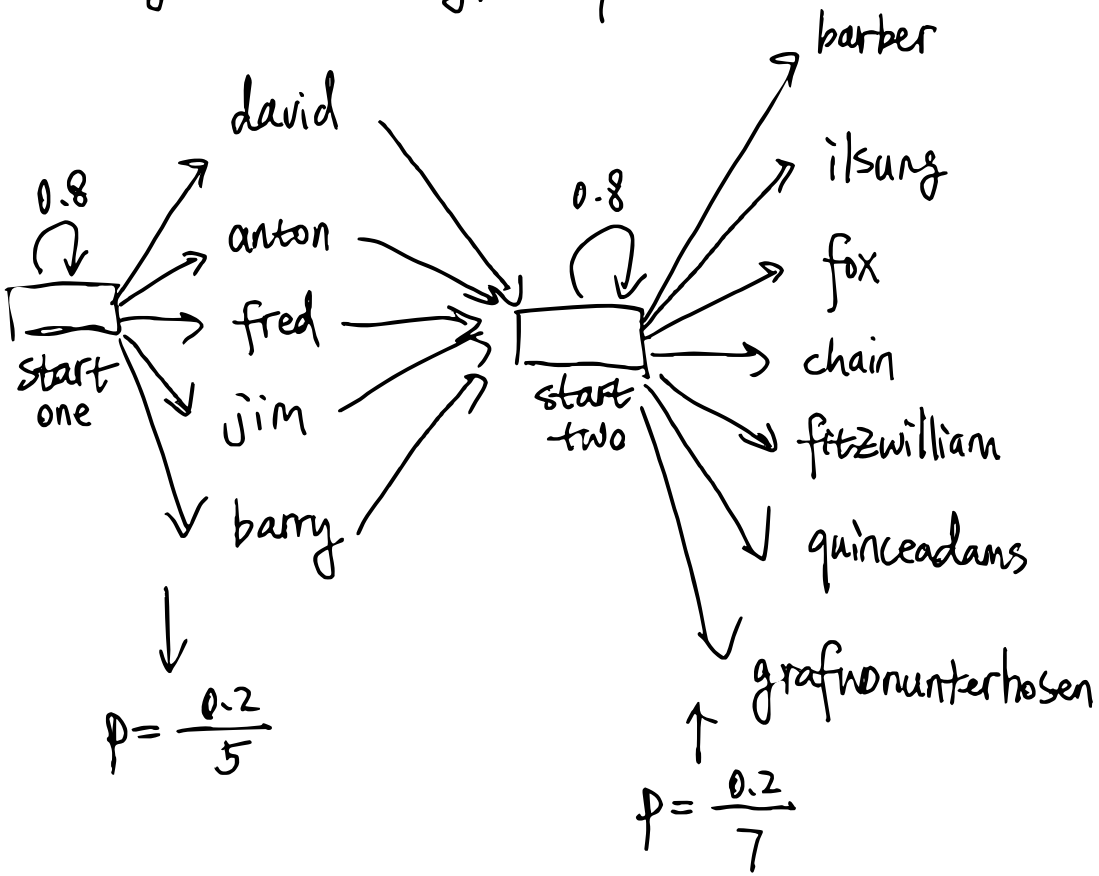
$$= 0.3147$$

Problem 2.

Logic:

P(generate correct) = 0.3

P(generate wrong) = 0.7



Thus, we have a transition matrix of 83×83 and an emission matrix of 83×26.

Problem 2

I did not get the results though I write a lot of code to solve this problem. I believe I get the right emission matrix and transition matrix, they are stored in the intersection results.

- Intersection results are stored in prob2inter.mat
- Codes are in HMMprob2.m and problem2.py, none of them works, but the problem both happens to the HMM algorithm
- In HMMprob2.m, I used the HMM algorithm from BRMLtoolbox, they principally should work but I don't know why it doesn't. I discuss and compare them with my friends but still cannot find my way debugging this so I give up.

Problem 3.

$P(a,b,c) = P(a|b)P(b|c)P(c)$

we know that

$P(a|b) =$

B \ A	T	F
T	0.3	(0.7)
F	0.2	(0.8)

$P(b|c) =$

C \ B	T	F
T	(0.75)	0.25
F	0.1	(0.9)

$P(c) =$

T	F
0.4	(0.6)

$$\max_{a,b,c} P(a,b,c) = \max_{a,b,c} P(a|b)P(b|c)P(c)$$

$$= \max_{c}\left[P(c)\cdot \max_{b}\left(P(b|c)\cdot \max_{a} P(a|b)\right)\right]$$

$$= \max_{c}\{0.4\cdot \max_{c=T}\{\underbrace{0.75\times 0.7}_{b=T}, 0.25\times 0.8\},$$

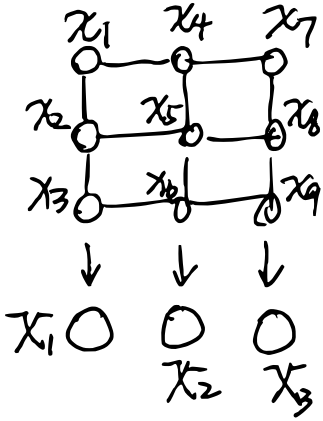
$$0.6\cdot \max\{0.1\times 0.7, 0.9\times 0.8\}\}.$$

$$= 0.6\times 0.9\times 0.8$$

$\therefore \operatorname{argmax}_{a,b,c} P(a,b,c) =$ **a=b=c=False**

Problem 4.

Complexity:

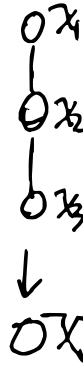


Example

when $n=3$.

For each x_k, $k \in [1, n^2]$ and $k \in \mathbb{N}$

when considering $\Rightarrow$



$$X_1(x_1, x_2, x_3) = \phi(x_1, x_2)\,\phi(x_2, x_3) = \Phi^{n-1}$$

Each $\phi(x_i, x_{i-1}) = \begin{bmatrix} e & 1 \\ 1 & e \end{bmatrix} \Rightarrow$ 4 choices

Thus Each X_j has $O(4^n)$, There are n X_j in total

Therefore, ~~total~~ complexity is $\boxed{O(n4^n)}$

Problem 4

Answer: 189

Code reference:

https://github.com/taheris/BRML.jl/blob/d8897d0e3d0d2f72a055c67e2e55e875d0bd6868/matlab/DemosExercises/demoSumprod.m

https://github.com/taheris/BRML.jl/blob/d8897d0e3d0d2f72a055c67e2e55e875d0bd6868/matlab/DemosExercises/demoSARinference.m

Problem4.py stores code that do not work (there is a brute force function actually works but take long)

Problem4.m stores code that work, but utilize BRML toolbox and referenced the above two demo programs from the BRML github repo. I am not sure if that is ok.

Problem 5

1. See solution page (next page)

2. results are the same!

Explain:

We can make the algrithom more efficient by using the assumption "Each symptom connects to 3 parent diseases".

$$p(sy) = \sum_{\vec{v}} p(sy \mid parent\vec{v})\, p(parent\vec{v})$$

By cleverly selecting the order according to structure, junction tree formalism can be surpassed.

Similar as Problem 7.

3. next page.

Problem 5

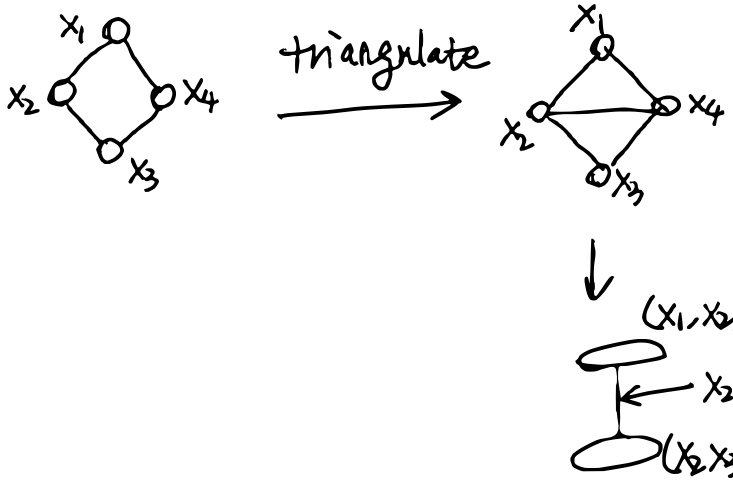
Code reference:
https://github.com/taheris/BRML.jl/blob/master/matlab/DemosExercises/demoJTree.m
https://github.com/taheris/BRML.jl/blob/master/matlab/DemosExercises/demoJTreeSample.m

s[1]=1 || 0.441834	s[1]=1 || 0.441834
s[2]=1 || 0.456675	s[2]=1 || 0.456675
s[3]=1 || 0.441405	s[3]=1 || 0.441405
s[4]=1 || 0.491275	s[4]=1 || 0.491275
s[5]=1 || 0.493892	s[5]=1 || 0.493892
s[6]=1 || 0.657483	s[6]=1 || 0.657483
s[7]=1 || 0.504563	s[7]=1 || 0.504563
s[8]=1 || 0.268693	s[8]=1 || 0.268693
s[9]=1 || 0.649077	s[9]=1 || 0.649077
s[10]=1 || 0.49074	s[10]=1 || 0.49074
s[11]=1 || 0.422552	s[11]=1 || 0.422552
s[12]=1 || 0.429096	s[12]=1 || 0.429096
s[13]=1 || 0.545021	s[13]=1 || 0.545021
s[14]=1 || 0.632959	s[14]=1 || 0.632959
s[15]=1 || 0.42954	s[15]=1 || 0.42954
s[16]=1 || 0.458794	s[16]=1 || 0.458794
s[17]=1 || 0.427559	s[17]=1 || 0.427559
s[18]=1 || 0.404255	s[18]=1 || 0.404255
s[19]=1 || 0.582093	s[19]=1 || 0.582093
s[20]=1 || 0.589591	s[20]=1 || 0.589591
s[21]=1 || 0.76127	s[21]=1 || 0.76127
s[22]=1 || 0.695588	s[22]=1 || 0.695588
s[23]=1 || 0.508702	s[23]=1 || 0.508702
s[24]=1 || 0.419962	s[24]=1 || 0.419962
s[25]=1 || 0.351942	s[25]=1 || 0.351942
s[26]=1 || 0.389611	s[26]=1 || 0.389611
s[27]=1 || 0.325973	s[27]=1 || 0.325973
s[28]=1 || 0.469624	s[28]=1 || 0.469624
s[29]=1 || 0.522868	s[29]=1 || 0.522868
s[30]=1 || 0.717312	s[30]=1 || 0.717312
s[31]=1 || 0.524199	s[31]=1 || 0.524199
s[32]=1 || 0.353704	s[32]=1 || 0.353704
s[33]=1 || 0.512679	s[33]=1 || 0.512679
s[34]=1 || 0.529404	s[34]=1 || 0.529404
s[35]=1 || 0.385751	s[35]=1 || 0.385751
s[36]=1 || 0.489095	s[36]=1 || 0.489095
s[37]=1 || 0.633595	s[37]=1 || 0.633595
s[38]=1 || 0.589604	s[38]=1 || 0.589604
s[39]=1 || 0.423164	s[39]=1 || 0.423164
s[40]=1 || 0.528234	s[40]=1 || 0.528234

d[1]=1 under condition =0.0297756
d[2]=1 under condition =0.38176
d[3]=1 under condition =0.954235
d[4]=1 under condition =0.396644
d[5]=1 under condition =0.496467
d[6]=1 under condition =0.435154
d[7]=1 under condition =0.187487
d[8]=1 under condition =0.701183
d[9]=1 under condition =0.0431266
d[10]=1 under condition =0.610313
d[11]=1 under condition =0.287322
d[12]=1 under condition =0.489833
d[13]=1 under condition =0.8996
d[14]=1 under condition =0.619565
d[15]=1 under condition =0.920476
d[16]=1 under condition =0.706096
d[17]=1 under condition =0.201247
d[18]=1 under condition =0.908494
d[19]=1 under condition =0.864967
d[20]=1 under condition =0.883929

Problem 6

1、 $p(x_1,x_2,x_3,x_4)=\phi(x_1,x_2)\phi(x_2,x_3)\phi(x_3,x_4)\phi(x_4,x_1)$



The junction tree is:

2、 Absorption procedure:

From graph: $p(x_1,x_2,x_3,x_4)=\frac{\phi(x_1,x_2,x_4)\phi(x_2,x_3,x_4)}{Z}$

$$Zp(x_1,x_2,x_4)=\phi(x_1,x_2,x_4)\sum_{x_3}\phi(x_2,x_3,x_4)$$

$$Zp(x_2,x_3,x_4)=\phi(x_2,x_3,x_4)\sum_{x_1}\phi(x_1,x_2,x_4)$$

Multiply, we have

$$p(x_1,x_2,x_3,x_4)=\frac{\phi(x_1,x_2,x_4)\phi(x_2,x_3,x_4)}{p(x_2,x_4)}$$

Absorption:

$$\phi^*(x_2,x_4)=\sum_{x_1}\phi(x_1,x_2,x_4)$$

$$=\sum_{x_1}\phi(x_1,x_2)\phi(x_1,x_4)$$

$$\phi^*(x_2,x_3,x_4)=\phi(x_2,x_3,x_4)\cdot\frac{\phi^*(x_2,x_4)}{\phi(x_2,x_4)}=\frac{\phi(x_2,x_3)\phi(x_3,x_4)}{\phi(x_2,x_4)}\cdot\sum_{x_1}\phi(x_1,x_2)\phi(x_1,x_4)$$

$$\Downarrow$$

$$\phi^*(x_2,x_3,x_4)=\frac{\sum_{x_1}\phi(x_1,x_2)\phi(x_2,x_3)\phi(x_3,x_4)\phi(x_4,x_1)}{\phi(x_2,x_4)}$$

$$=\frac{\sum_{x_1}p(x_1,x_2,x_3,x_4)}{\phi(x_2,x_4)}=\frac{p(x_2,x_3,x_4)}{\phi(x_2,x_4)}$$

$$\frac{\phi(x_1,x_2,x_4)\phi^*(x_2,x_3,x_4)}{\phi^*(x_2,x_4)}=\frac{\phi(x_1,x_2,x_4)\cdot p(x_2,x_3,x_4)}{\sum_{x_1}\phi(x_1,x_2)\phi(x_1,x_4)\phi(x_2,x_4)}$$

$$=\frac{\phi(x_1,x_2,x_4)\cdot p(x_2,x_3,x_4)}{\sum_{x_1}\phi(x_1,x_2,x_4)}=p(\chi)$$

★ backward:

$$\phi^{**}(x_2,x_4)=\sum_{x_3}\phi^*(x_2,x_3,x_4)=\frac{p(x_2,x_4)}{\phi(x_2,x_4)}$$

$$\phi^*(x_1,x_2,x_4)=\frac{\phi(x_1,x_2,x_4)\phi^{**}(x_2,x_4)}{\phi^*(x_2,x_4)}=\frac{\phi(x_1,x_2,x_4)\cdot p(x_2,x_4)}{\sum_{x_1}\phi(x_1,x_2,x_4)}=p(\chi)$$

From the last equation we know that

$$p(x_1) = \sum_{x_2,x_4} \phi^*(x_1,x_2,x_4) = \sum_{x_2,x_4} \frac{\phi(x_1,x_2,x_4)\, p(x_2,x_4)}{\sum_{x_1} \phi(x_1,x_2,x_4)}$$

$$= \sum_{x_2,x_4} \frac{\phi(x_1,x_2,x_4) \sum_{x_1,x_3} \phi(x_1,x_2)\,\phi(x_2,x_3)\,\phi(x_3,x_4)\,\phi(x_1,x_4)}{\sum_{x_1} \phi(x_1,x_2)\,\phi(x_1,x_4)}$$

$$= \sum_{x_2,x_4} \frac{\phi(x_1,x_2)\,\phi(x_1,x_4) \sum_{x_3} \phi(x_2,x_3)\,\phi(x_3,x_4) \cdot \sum_{x_1} \phi(x_1,x_2)\,\phi(x_1,x_4)}{\sum_{x_1} \phi(x_1,x_2)\,\phi(x_1,x_4)}$$

$$= \sum_{x_2,x_4} \phi(x_1,x_2)\,\phi(x_1,x_4) \sum_{x_3} \phi(x_2,x_3)\,\phi(x_3,x_4)$$

$$= \sum_{x_2,x_3,x_4} \phi(x_1,x_2)\,\phi(x_1,x_4)\,\phi(x_2,x_3)\,\phi(x_3,x_4)$$

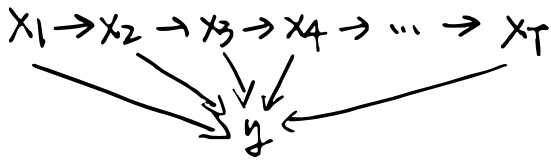
$$= \sum_{x_2,x_3,x_4} p(x_1,x_2,x_3,x_4) = p(x_1)$$

Hence, the absorption procedures show that this gives correct result for $p(x_1)$. ▨

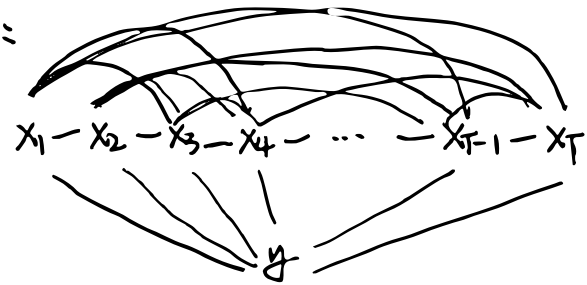
Problem 7.

$p(y|x_1,x_2,\cdots,x_T)p(x_1)\prod_{t=2}^{T}p(x_t|x_{t-1})$, with all binary variables

1. distribution network looks like ⇓



Moralize :



connect with each other.

Since they are connected with each others
No need to triangulate.

Maximum spanning tree :

Since the moralized graph has all the variables connected with each other, this resulted graph is just a single clique of all variables.

computational complexity of computing $p(x_T)$

There are ~~total~~ $T+1$ variables, each variable has two values, so. overall complexity is $\boxed{O(2^T)}$.

2.

$$p(y, x_1, x_2, \cdots, x_T) = p(y|x_1, \cdots, x_T)\, p(x_1) \prod_{t=2}^{T} p(x_t|x_{t-1})$$

$$\sum_y p(y, x_1, \cdots, x_T) = \sum_y p(y|x_1, \cdots, x_T) \cdot p(x_1) \prod_{t=2}^{T} p(x_t|x_{t-1})$$

Thus, $p(x_1, \cdots, x_T) = p(x_1) \prod_{t=2}^{T} p(x_t|x_{t-1})$

This is a simple linear chain: $x_1 \to x_2 \to \cdots \to x_T$

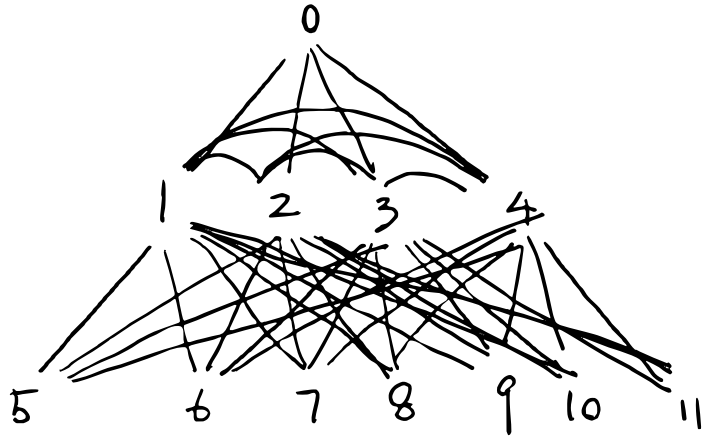
so the complexity is $\boxed{O(T)}$.

Thus, the trick is to sum y first.

Problem 8.

(a). Variable Elimination.

Moralized Graph:



New connections inside moralized graph:

1-2, 1-3, 1-4, 2-3, 2-4, 3-4.

I will show elimination in (both) original graph and moralized graph.

In original BN, $p(\vec{x}) = p(0) \prod_{i=1,2,3,4} p(i|0) \prod_{j=5,\cdots,11} p(j|1,2,3,4)$.

Since our query is: (1) with 11 = 8 = true.

$$p(1) = \sum_{0,2,3,\cdots,11} p(\vec{x}) =$$

$$= \sum_0 \left(p(0)\, p(1|0) \sum_{2,3,4} \left(p(2|0)\, p(3|0)\, p(4|0) \sum_{5,\dots,11} \left(\prod_{5,\dots,11}^{5} p(j|1,2,3,4) \right) \right) \right)$$

A good ordering = 5, 6, 7, 8, 9, 10, 11, 2, 3, 4, 0.

Procedure = ① when eliminating 5, 6, 7, 9, 10 ←

$\sum_5 p(5|1,2,3,4) = 1$, so all of them = 1

② when eliminating 8, 11

$\sum_8 p(8 = \text{true}|1,2,3,4) = g_1(1,2,3,4)$

$\sum_{11} p(11 = \text{true}|1,2,3,4) = g_2(1,2,3,4)$

Equation becomes $P(1) = \sum_0 \left(p(0)\, p(1|0) \sum_{2,3,4} \left(p(2|0)\, p(3|0)\, p(4|0) \cdot g_1(1,2,3,4)\, g_2(1,2,3,4) \right) \right)$.

③ when eliminating 2, 3, 4.

$\sum_2 p(2|0)\, g_1(1,2,3,4)\, g_2(1,2,3,4) = m_1(0,1,3,4)$

$\sum_3 p(3|0)\, m_1(0,1,3,4) = m_2(0,1,4)$

$\sum_4 p(4|0)\, m_2(0,1,4) = m_3(0,1)$

④ Eliminating 0 : $\sum_0 p(0)\, p(1|0)\, m_3(0,1) = P(1)$.

Time complexity:

For 5,6,7,8,9,10,11, we do not need to compute.

For the rest, we have $O(n^4)$

Space complexity:

since all nodes are binary, it is $O(2^4)$

In moralized graph $p(\vec{x}) = \phi(0,1)\phi(0,2)\phi(0,3)\phi(0,4)\cdot$

$\phi(1,2)\phi(1,3)\phi(1,4)\phi(2,3)\phi(3,4)\phi(2,4)\cdot$

$\prod_{i=1,2,3,4} \prod_{j=5,6,\cdots,11} \phi(i,j)$.

we know that $11=8=$ true. We need to compute $P(1)$.

Good ordering: 5,6,7,8,9,10,11,2,3,4,0.

Procedure ① when eliminating 5,6,7,9,10

$\sum_5 \phi(1,5)\phi(2,5)\phi(3,5)\phi(4,5) = g_1(1,2,3,4)$

similarly, for 6,7,9,10, we have

$\sum_6 \cdots = g_2(1,2,3,4)$, $\cdots$ $g_3(1,2,3,4)$ $\cdots$ $g_5(1,2,3,4)$.

② For 8, 11 = true.

$\sum_8 \phi(1,8)\phi(2,8)\phi(3,8)\phi(4,8) = g_6(1,2,3,4)$

$\sum_{11} \cdots = g_7(1,2,3,4)$

③ For 2,3,4:

$\sum_2 \phi(0,2)\phi(1,2)\phi(2,3)\phi(2,4) \prod_{i=1,\dots,7} g_i(1,2,3,4)$

$= m_1(0,1,3,4)$

$\sum_3 \phi(0,3)\phi(1,3)\phi(3,4)\, m_1(0,1,3,4) = m_2(0,1,4)$

$\sum_4 \phi(0,4)\phi(1,4)\, m_2(0,1,4) = m_3(0,1)$

④ For 0: $\sum_0 \phi(0,1)\, m_3(0,1) = m_4(1) = p(1)$.

Time complexity:

① For $5,\dots,11$: $7 \times n^4$

Also $\boxed{O(n^4)}$

Space complexity:

$\boxed{O(n^5)}$

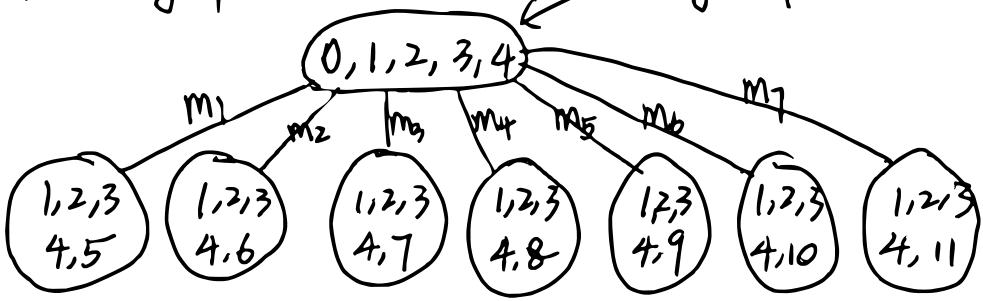
(b) Clique tree:

(i). First, moralize the graph as in previous question.

The moralized graph does not need to be trangularized.

↓

cluster graph.



All the lines consist of message $\boxed{1,2,3,4}$ ✓.

The messages here correspond to ⇒

↓

For example, $m_1 = \sum_5 \phi(1,2,3,4,5) = g(1,2,3,4)$.

Time complexity: time complexity is proportional to the cost to compute the biggest clique. $\boxed{O(n^5)}$

Space complexity:

$\boxed{O(n^5)}$

Problem 8

c) Estimating Parameters Results
The whole results can be viewed inside code problem8.py, they are copied and pasted here (2 page)

In the following:

- the first row is node number
- ignore the second row
- the third row is the parents of this node
- the forth row is the children of this node
- the fifth row is the probability. For example, the underlined line means the p(0=F) = 0.75 and p(0=T) = 0.25

0 0 [] [1, 2, 3, 4] [[0.750094, 0.24990599999999996]]
1 0 [0] [5, 6, 7, 8, 9, 10, 11] [[0.9500719243188187, 0.049928075681181294], [0.989900202476131, 0.010099797523869025]]
2 0 [0] [5, 6, 7, 8, 9, 10, 11] [[0.8998638837265729, 0.10013611627342711], [0.9000204076732852, 0.09997959232671483]]
3 0 [0] [5, 6, 7, 8, 9, 10, 11] [[0.8001850434745512, 0.19981495652544878], [0.7990784534985155, 0.20092154650148453]]
4 0 [0] [5, 6, 7, 8, 9, 10, 11] [[0.9800208373883806, 0.01997916261161936], [0.9800364937216394, 0.019963506278360632]]
5 0 [1, 2, 3, 4] [] [[0.989880843181861, 0.010119156818139019], [0.24858920561423817, 0.7514107943857619], [0.500451523226886, 0.499548476773114], [0.14772727272727273, 0.8522727272727273], [0.9896824342280096, 0.010317565771990389], [0.2546755570011384, 0.7453244429988616], [0.5001587637595258, 0.49984123624047416], [0.14055144586415602, 0.859448554135844], [0.6010265320414513, 0.3989734679585487], [0.19616204690831557, 0.8038379530916844], [0.4015787405793478, 0.5984212594206522], [0.00964630225080386, 0.9903536977491961], [0.6001738718090571, 0.3998261281909429], [0.2145922746781116, 0.7854077253218884], [0.39054181020983403, 0.609458189790166], [0.0, 1.0]]
6 0 [1, 2, 3, 4] [] [[0.980023395059427, 0.019976604940572962], [0.19720373317899, 0.80279626682101], [0.9800267372577489, 0.019973262742251086], [0.20432692307692307, 0.7956730769230769], [0.7012357161838958, 0.2987642838161042], [0.15287038542852496, 0.8471296145714751], [0.6997512701100762, 0.3002487298899238], [0.15870880968392737, 0.8412911903160727], [0.9802583173030043, 0.019741682696995 72], [0.20426439232409382, 0.7957356076759061], [0.9794263671107619, 0.020573632889238 08], [0.18810289389067525, 0.8118971061093248], [0.698016280723939, 0.30198371927606105], [0.1888412017167382, 0.8111587982832618], [0.6999686814907611, 0.3000313185092389], [0.16326530612244897, 0.8367346938775511]]
7 0 [1, 2, 3, 4] [] [[0.8001029116845735, 0.19989708831542652], [0.7987447547388221, 0.20125524526117788], [0.800276225974095, 0.1997237740259 0502], [0.8032488344988346, 0.19675116550116545], [0.8002757108689875, 0.19972428913101248], [0.7937876077410961, 0.20621239225890386], [0.80169083403895, 0.19830916596104997], [0.8177538668459986, 0.1822461331540014], [0.7998705754075545, 0.20012942459244554], [0.8046908315565032, 0.19530916844349677], [0.7974425831339073, 0.20255741686609274], [0.8279742765273312,

0.17202572347266876], [0.7974393424484312, 0.20256065755156882], [0.7939914163090128, 0.20600858369098718], [0.7892264328217977, 0.21077356717820228], [0.8163265306122449, 0.18367346938775508]]
8 0 [1, 2, 3, 4] [] [[0.8933561058041378, 0.10664389419586218], [0.40949211402112573, 0.5905078859788743], [0.5762911941166229, 0.4237088058833771], [0.34484265734265734, 0.6551573426573427], [0.8938280627159182, 0.10617193728408181], [0.40884696698650186, 0.5911530330134982], [0.5733885478408128, 0.42661145215918717], [0.3382649630127774, 0.6617350369872226], [0.6396145629260596, 0.36038543707394044], [0.373134328358209, 0.6268656716417911], [0.5091259777833339, 0.4908740222166661], [0.2508038585209003, 0.7491961414790997], [0.6392159962064332, 0.3607840037935668], [0.38197424892703863, 0.6180257510729614], [0.5104917005950517, 0.48950829940494833], [0.24489795918367346, 0.7551020408163265]]
9 0 [1, 2, 3, 4] [] [[0.8821376986392994, 0.11786230136070064], [0.21292142960497756, 0.7870785703950225], [0.6409726341510921, 0.35902736584890793], [0.19296328671328672, 0.8070367132867133], [0.7022455487642838, 0.2977544512357162], [0.20344771507562207, 0.7965522849243779], [0.5083615580016935, 0.4916384419830653], [0.1882985877605918, 0.8117014122394082], [0.6926697811306036, 0.3073302188693964], [0.20170575692963752, 0.7982942430703625], [0.5932421330856877, 0.40675786691431226], [0.1607717041800643, 0.8392282958199357], [0.5429542401011618, 0.4570457598988382], [0.18025751072961374, 0.8197424892703863], [0.46382712182900093, 0.5361728781709991], [0.20408163265306123, 0.7959183673469388]]
10 0 [1, 2, 3, 4] [] [[0.9350120832639688, 0.06498791673603121], [0.3504919693242657, 0.6495080306757344], [0.9351931368469514, 0.06480686315304862], [0.35059731934731936, 0.6494026806526807], [0.7268004251926654, 0.2731995748073346], [0.3200520409822735, 0.6799479590177264], [0.7256429932260796, 0.2743570067739204], [0.3113651647612643, 0.6886348352387357], [0.9342682634942867, 0.06573173650571329], [0.34968017057569295, 0.6503198294243071], [0.9353144979819267, 0.0646855020180733], [0.342443729903537, 0.657556270096463], [0.7234647909586659, 0.2765352090413341], [0.36909871244635195, 0.630901287553648], [0.7253366739743188, 0.2746633260256812], [0.2857142857142857, 0.7142857142857143]]
11 0 [1, 2, 3, 4] [] [[0.9798679939862368, 0.020132006013763193], [0.5007596585154103, 0.49924034148458973], [0.980072479806878, 0.01992752019312205], [0.49628496503496505, 0.503715034965035], [0.9798133138453362, 0.02018668615466379], [0.5060985526101806, 0.49390144738981945], [0.9794400931414056, 0.020559906858594434], [0.5010087424344317, 0.4989912575655683], [0.3990709796377884, 0.6009290203622115], [0.041791044776119404, 0.9582089552238806], [0.3967567953709326, 0.6032432046290674], [0.05305466237942122, 0.9469453376205788], [0.40164387892199477, 0.5983561210780053], [0.06437768240343347, 0.9356223175965666], [0.4021296586282493, 0.5978703413717508], [0.061224489795918366, 0.9387755102040817]]

d) Model Accuracy Results

0.33079296508898476

e) Implement Variable Elimination Results

All the states are stored inside the joint result. For example: in query 1, the first number 0.48 is when HasFlu = 0 (which is False in my setting) and the second number 0.52 is when HasFlu=1 (True).

- query 1

From dataset
[0.4828241958324899, 0.5171758041675093]
Time cost
0.21851897239685059
From joint
([0.48377725494618024, 0.516222745053821], [['0'], ['1']])
Time cost
0.013344764709472656

- query 2

Similarly, in the joint results, there are things look like "['0', '0', '0', '0', '0'], ['0', '0', '0', '0', '1']", they are states of queried variables, and are listed in the order [HasRash, Coughs, IsFatigued, Vomits, HasFever]

Sorry for the difficulty of reading them, I do not have time pretty print them.

From dataset
[0.022277568491007524, 0.023967159359707536, 0.01919875846359862, 0.01956170761317121, 0.03001214002327868, 0.03364163151900463, 0.08150085731092192, 0.08590630905745825, 0.022352661418505303, 0.02331635398806012, 0.01702106356616305, 0.01837273626112306, 0.058334689177857335, 0.06389156581269292, 0.13421609241436222, 0.14631856922942077, 0.005932341272324477, 0.0060950426152363325, 0.00444299821028521, 0.0045306066256992844, 0.007346591406865967, 0.008272737512671897, 0.020312636888149, 0.021526639216029746, 0.005256504924844474, 0.0061701355427341105, 0.004568153089448172, 0.004405451746536322, 0.014392811103740816, 0.01610743294827342, 0.03474299445563871, 0.03600705873518463]
Time cost
3.6297829151153564
From joint
([0.02319813096747807, 0.024611457991446276, 0.018606310471581386, 0.019731297367556078, 0.030116225231322437, 0.03224594980727452, 0.08205581332615387, 0.08762041483002869, 0.02192682511150668, 0.02365608352746778, 0.01671874384762385, 0.018034238711594253, 0.058272818686688946, 0.06297250867058321, 0.1347051323489693, 0.1455280490847258, 0.005799532743869518, 0.006152864499361568, 0.004651577618895346, 0.00493282434288902, 0.007529056309830611, 0.008061487453818633, 0.020513953332538467, 0.021905103708507175, 0.005481706279876671, 0.005914020883366943, 0.004179685962905962, 0.004508559678898564, 0.014568204673672237, 0.015743127169145804, 0.03367628308824233, 0.03638201227218145], [['0', '0', '0', '0', '0'], ['0', '0', '0', '0', '1'], ['0', '0', '0', '1', '0'], ['0', '0', '0', '1', '1'], ['0', '0', '1', '0', '0'], ['0', '0', '1', '0', '1'], ['0', '0', '1', '1', '0'], ['0', '0', '1', '1', '1'], ['0', '1', '0', '0', '0'], ['0', '1', '0', '0', '1'], ['0', '1', '0', '1', '0'], ['0', '1', '0', '1', '1'], ['0', '1', '1', '0', '0'], ['0', '1', '1', '0', '1'], ['0', '1', '1', '1', '0'], ['0', '1', '1', '1', '1'], ['1', '0', '0', '0', '0'], ['1', '0', '0', '0', '1'], ['1', '0', '0', '1', '0'], ['1', '0', '0', '1', '1'], ['1', '0', '1', '0', '0'], ['1', '0', '1', '0',

'1'], ['1', '0', '1', '1', '0'], ['1', '0', '1', '1', '1'], ['1', '1', '0', '0', '0'], ['1', '1', '0', '0', '1'], ['1', '1', '0', '1', '0'], ['1', '1', '0', '1', '1'], ['1', '1', '1', '0', '0'], ['1', '1', '1', '0', '1'], ['1', '1', '1', '1', '0'], ['1', '1', '1', '1', '1']])
Time cost
0.25368380546569824

- query 3

From dataset
[0.9027254247597102, 0.09727457524029086]
Time cost
0.22149205207824707
From joint
([0.9026449999997637, 0.09735500000023867], [['0'], ['1']])
Time cost
0.01698589324951172

Summary:

1) More detailed results are listed above. Everything that is a direct answer is listed above.
2) Everything is inside the problem8.py code. Key things are: the Node class, the Model class, and the binary transfer function. Via binary transfer function, we can easily count every possible state. We first define the structure of the model via a 12*12 np array, and initialize all the nodes, and by feeding the numbers read from dataset we update the nodes and hence get results.
3) The accuracy is 0.33, which is a really good number. Because we have 4096 numbers in total, and 0.33/4096 = 0.00008, which means for each state, the accuracy of our model is around 0.00008 different from the true state. This may due to some disease has very low occurrence possibility, but the overall accuracy is still very impressive.
4) Generally speaking, getting results from joint.dat is much faster than getting results from the algorithm. But the final numbers are similar, for example, for query 1, the difference of patient having Flu is only 0.000953.