ECE GY-6143 ML Homework 5 Solution

Yogya Sharma

(ys5250@nyu.edu)

Answer 1) C: door with car hidden behind.
I: Initially chosen door
O: The door the host opened

$C, I, O \in \{1, 2, 3\}$

Behind Door 1	Behind Door 2	Behind Door 3	Result if sticking with Door 1	Result if switching.
empty	empty	car	No win	wins car
empty	car	empty	No win	No win
car	empty	empty	wins car	No win

Assuming that door 1 is chosen initially, and then the host opens the door with nothing behind it. In this case, if the chosen door is switched, in 2 of 3 equally likely scenarios a car is won.

Explanation:
Given the first choice, the probability of choosing the door with car behind it is 1/3. The probability of the car being behind the other 2 doors is 2/3. As the game show host opens the door that has nothing behind it, this probability doesn't change. Thus, the unopened door still has a probability of 2/3 of having a car behind it and the door that was chosen initially has a 1/3 probability of having a car behind it.

Initially the probability of choosing any door,

$$P(C=1) = P(C=2) = P(C=3) = 1/3$$

Suppose that the host opens the 3rd door and the initially chosen door was 1.

$$P(O=3 \mid C=1, I=1) = 1/2$$
$$P(O=3 \mid C=2, I=1) = 1$$
$$P(O=3 \mid C=3, I=1) = 0$$

using Bayesian rule calculating the probability of winning the car by not switching the door:

$$P(C=1 \mid O=3, I=1) = \frac{P(O=3 \mid C=1, I=1)\, P(C=1 \mid I=1)}{\sum_{i=1}^{3} P(O=3 \mid C=i, I=1)\, P(C=i \mid I=1)}$$

$$= \frac{P(O=3 \mid C=1, I=1)\, P(C=1)}{\sum_{i=1}^{3} P(O=3 \mid C=i, I=1)\, P(C=i)}$$

$$= \frac{P(O=3 \mid C=1, I=1)}{\sum_{1}^{3} P(O=3 \mid C=i, I=1)}$$

$$= \frac{1/2}{1/2 + 1 + 0} = 1/3$$

$$P(C=1 \mid O=3, I=1) + P(C=2 \mid O=3, I=1) + P(C=3 \mid O=3, I=1) = 1$$

We know that $\rightarrow$ $P(C=3 \mid O=3, I=1) = 0$

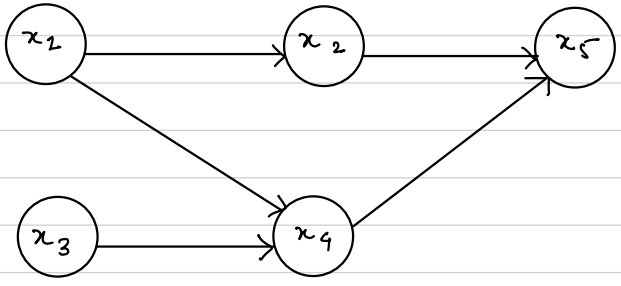
Probability of winning a car by switching to the second door $= P(C=2 \mid O=3, I=1)$

$$P(C=2 \mid O=3, I=1) = 1 - P(C=1 \mid O=3, I=1)$$
$$= 2/3$$

$$P(C=1 \mid O=3, I=1) < P(C=2 \mid O=3, I=1)$$

Thus, there's a better chance to win if I switch to the other door.

Answer 2)

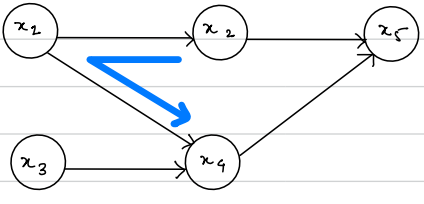


- x_1 is intelligent
- x_2 is good at taking tests
- x_3 is hard working
- x_4 understands the material
- x_5 gets good grade

Factorization of the probability distribution:

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1)\, p(x_2|x_1)\, p(x_3)\, p(x_4|x_2, x_3)\, p(x_5|x_2, x_4)$$

(i) If the ball is in x_2, considering the path x_2, x_5, x_4 (2 causes)

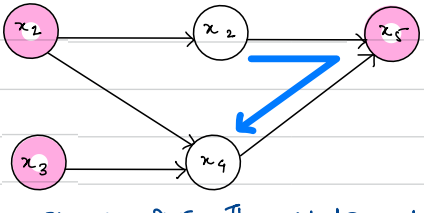


if ball starts from x_2, it will stop at x_5.

Considering path x_2, x_2, x_4 (2 effects), starting from x_2, ball passes through x_2 and reaches x_4.

Hence, x_2 and x_4 are not independent, the statement in question is FALSE.

(ii) If the ball is in x_2, considering path x_2, x_5, x_4 (2 causes)

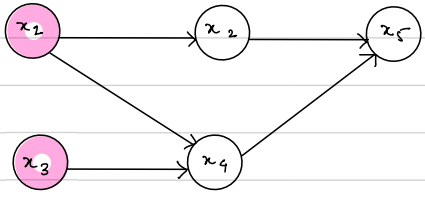


If the ball starts from x_2, it passes through x_5, and reaches x_4.

Hence, x_2 and x_4 are not independent given x_1, x_3, x_5. The statement

is question is FALSE.

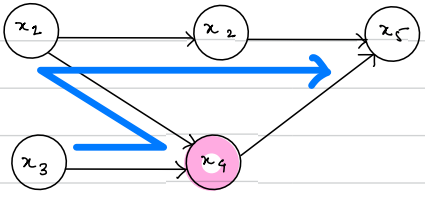
(iii) If the ball is in x_2, considering the path x_2, x_5, x_4, If the ball starts at x_2, it doesn't pass through x_5 (2 causes).



Considering the path x_2, x_1, x_4 (2 effects), if the ball starts at x_2, the ball stops at x_1.

As there no path for the ball to reach from x_2 to x_4, x_2 and x_4 are independent given x_1 and x_3 $(x_2 \perp\!\!\!\perp x_4 \mid x_1, x_3)$, the statement in question is TRUE.

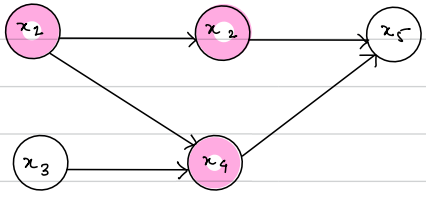
(iv) If the ball is in x_3, considering the path x_3, x_4, x_5 (Markov chains). If the ball starts at x_3, it will not go through x_4.



Considering the path x_3, x_4, x_1, x_2, x_5. Firstly checking for x_3, x_4, x_1 (2 causes), ball goes through x_4. Secondly checking path x_4, x_1, x_2 (2 effects), as x_1 is not shaded, the ball goes through x_2. Thirdly checking x_1, x_2, x_3 (Markov chain), as x_2 is not shaded, the ball goes through x_2, and reaches x_5.

As the ball was able to reach from x_3 to x_5, x_3 and x_5 are not conditionally independent given x_4, and the statement in question is FALSE.

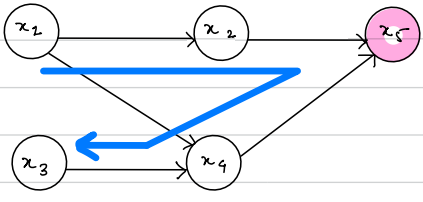
(v) If the ball starts at x_3, considering the path x_3, x_4, x_5 (Markov chain). If the ball starts from x_3, as x_4 is shaded, it does not pass through it.



Considering path x_3, x_4, x_1, x_2, x_5. If ball starts at x_3, firstly considering x_3, x_4, x_1 (2 causes), ball goes through x_4. Second, we consider x_4, x_1, x_2 (2 effects), as x_2 is shaded, ball does not go through it.

As the ball wasn't able to reach from x_3 to x_5, they are conditionally independent given x_1, x_2, x_4 $(x_3 \perp\!\!\!\perp x_5 \mid x_1, x_2, x_4)$. The statement in question is TRUE.

(vi) If the ball starts at x_1, considering path x_1, x_3, x_4 (two causes). If the ball starts from x_2, does not go through x_4, as it's not shaded.

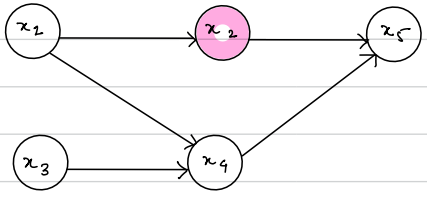


Considering path x_1, x_2, x_5, x_4, x_3. If the ball starts at x_1, first considering x_1, x_2, x_5 (Markov chain), as x_2 isn't shaded, ball goes through. Second we consider x_2, x_5, x_4 (2 causes), ball goes goes through x_5 (as it's shaded). Third, considering x_5, x_4, x_3 (Markov chain), as x_4 is not shaded, the ball goes through it and reaches x_5.

As the ball reaches from x_1 to x_3, they are conditionally independent

given x_5 $(x_2 \perp\!\!\!\perp x_3 \mid x_5)$. The statement in question is FALSE.

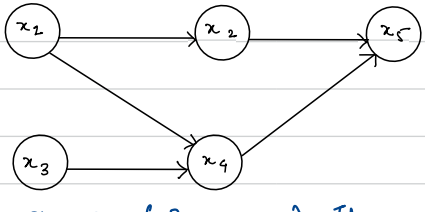
(vii) If the ball starts at x_1, considering the path x_2, x_4, x_3 (2 causes). As x_4 is not shaded the ball stops, doesn't go through x_4.



Considering the path x_1, x_2, x_5, x_4, x_3. If the ball starts from x_1, first we consider x_1, x_2, x_5 (Markov chain), as x_2 is shaded the ball doesn't go through.

As the ball is not able to reach from x_1 to x_3, x_1 and x_3 are conditionally independent given x_2 $(x_1 \perp\!\!\!\perp x_3 \mid x_2)$. The statement in question is TRUE.

(viii) If the ball starts at x_2, considering the path x_2, x_1, x_4, x_3. First consider x_2, x_1, x_4 (2 effects) if ball starts at x_2, it passes through x_1 (as it's not shaded). Second, considering x_1, x_4, x_3 (2 causes). It doesn't pass through x_4 (as its not shaded).

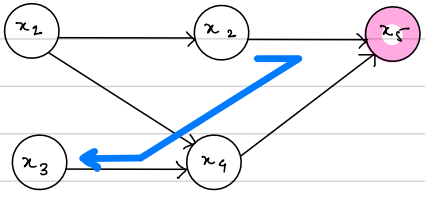


Considering the path x_2, x_5, x_4, x_3. First, considering x_2, x_5, x_4 (2 causes). If the ball starts at x_2, it won't go through x_5 (as it's not shaded)

As the ball is not able to reach from x_2 to x_3, they are independent.

$x_2 \perp\!\!\!\perp x_3$. The statement in question is TRUE.

(ix) If the ball starts at x_2, considering the path x_2, x_1, x_4, x_3

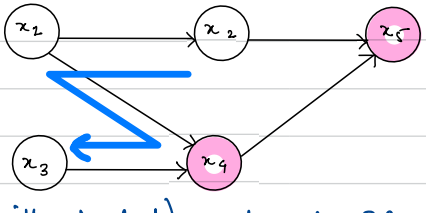


First, consider path x_2, x_1, x_4 (2 effects), if the ball starts at x_2, it does pass through x_1 (as it's not shaded). Second, consider x_1, x_4, x_3 (2 causes), it doesn't pass through x_4 (as it's not shaded).

Considering path x_2, x_5, x_4, x_3. First, consider x_2, x_5, x_4 (2 causes) if the ball starts at x_2, it passes through x_5 (as its shaded). Second, consider, x_5, x_4, x_3 (Markov chain), ball passes through x_4 (as it's not shaded and reaches x_3.

As the ball is able to reach from x_2 to x_3, they are conditionally independent given x_5 ($x_2 \perp\!\!\!\perp x_3 \mid x_5$). The statement in question is FALSE.

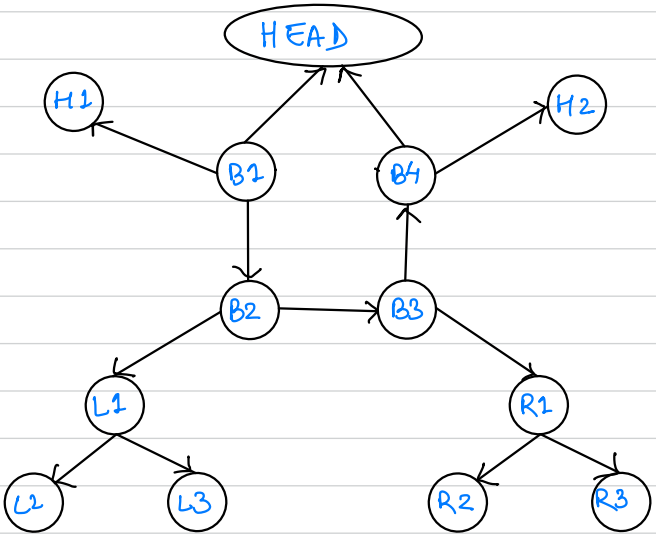
(x) If the ball starts at x_2, considering the path x_2, x_1, x_4, x_3. First consider x_2, x_1, x_4 (2 effects). Ball goes through x_1 (as its not shaded). Second, consider



x_2, x_4, x_3 (2 causes). Ball passes through x_4 (as its shaded) and reaches x_3.

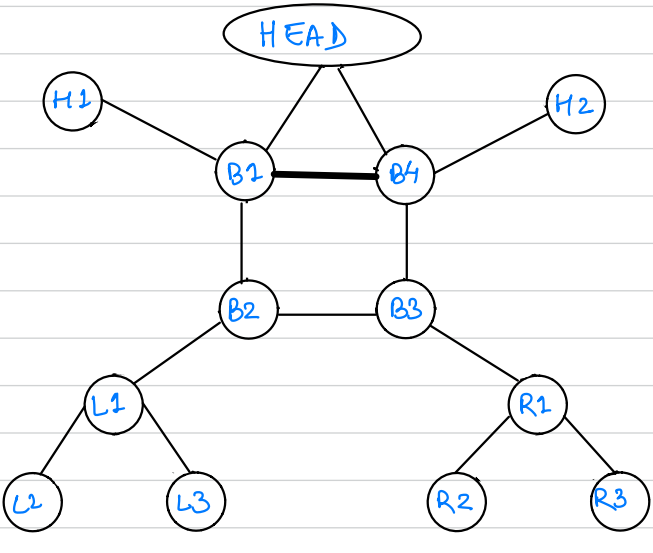
As the ball reaches from x_2 to x_3, they are not conditionally independent given x_4 and x_5. The statement in question is FALSE.

Answer 3)



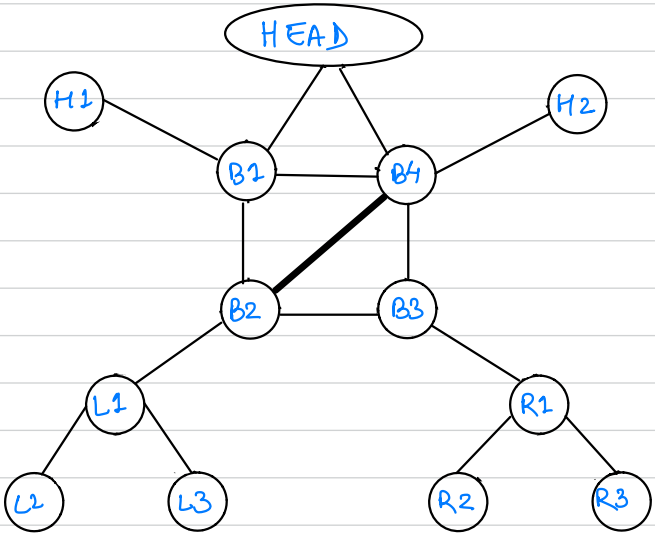
Head is represented as 'H' in the table on the next page.

1. Moralizing



Node HEAD had two parents B1 and B4, they have been joined.

2. Triangulation

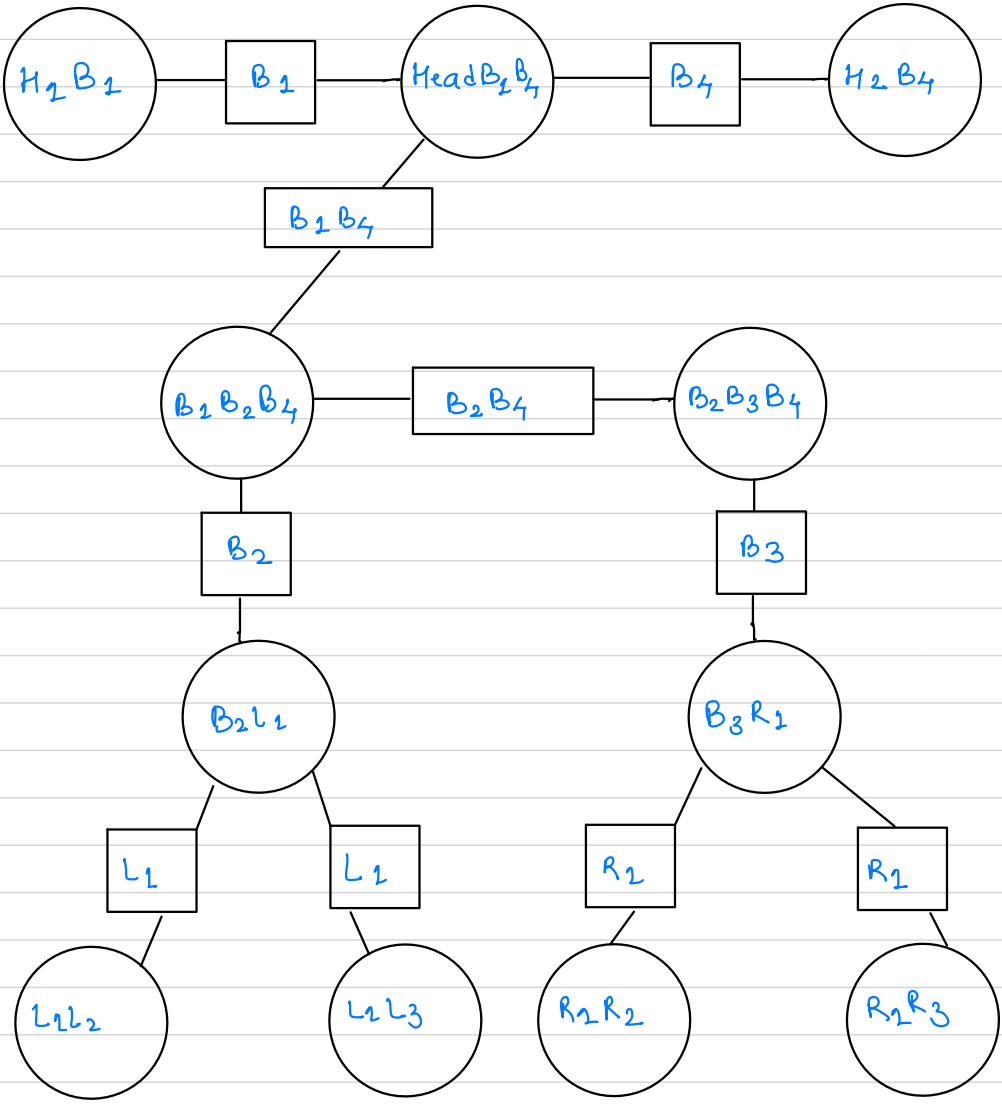


There was only one cycle in the graph B1B2B3B4 with 4 nodes, where B2B4 and B1B3 weren't pairwise adjacent. During triangulation joined B2 and B4.

3.

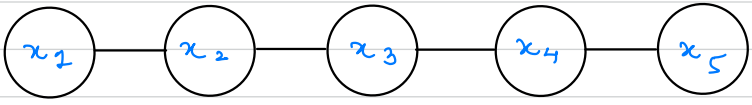
	$B_1B_2B_4$	$B_2B_3B_4$	HB_1B_4	H_1B_1	H_2B_4	B_2L_1	B_3R_1	R_1R_2	R_2R_3	L_1L_2	L_1L_3
$B_1B_2B_4$	-	2	2	1	1	1	0	0	0	0	0
$B_2B_3B_4$	—	—	1	0	1	1	1	0	0	0	0
HB_1B_4	—	—	—	1	1	0	0	0	0	0	0
H_1B_1	—	—	—	—	0	0	0	0	0	0	0
H_2B_4	—	—	—	—	—	0	0	0	0	0	0
B_2L_1	—	—	—	—	—	—	0	0	0	1	1
B_3R_1	—	—	—	—	—	—	—	1	1	0	0
R_1R_2	—	—	—	—	—	—	—	—	1	0	0
R_2R_3	—	—	—	—	—	—	—	—	—	0	0
L_1L_2	—	—	—	—	—	—	—	—	—	—	1
L_2L_3	—	—	—	—	—	—	—	—	—	—	

4. Using KRUSKAL'S ALGORITHM, using the table above creating the Junction Tree with separators.



Answer 4)

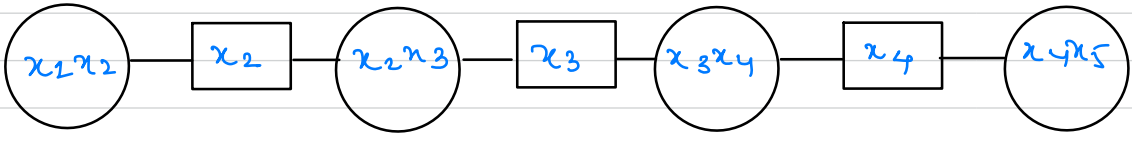
Given graph:



Cliques in this graphs are: x_1x_2 x_2x_3 x_3x_4 x_4x_5

	x_1x_2	x_2x_3	x_3x_4	x_4x_5
x_1x_2	–	1	0	0
x_2x_3	–	–	1	0
x_3x_4	–	–	–	1
x_4x_5	–	–	–	–

using kruskals algorithm, using the table above, creating the junction tree with separators.

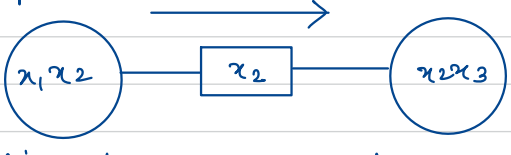


Junction Tree algorithm for the tree on the previous page:

- The probability distribution for the undirected graph is:

$$p(x_1, \ldots x_5) = \frac{1}{Z} \psi(x_1, x_2)\, \psi(x_2, x_3)\, \psi(x_3, x_4)\, \psi(x_4, x_5)$$

In the junction tree consider the cliques x_1x_2 and x_2x_3 with separator x_2.



We start sending the message from x_1x_2, it tells x_2 what the submarginal on the separator is.

$V = \{x_1, x_2\}$ $W = \{x_2, x_3\}$ $S = \{x_2\}$

$*$ → first update $**$ → second update

While sending the message from x_1x_2 to x_2x_3, the potentials will be updated as:

Separator potential: $\phi_S^* = \sum_{V/S} \psi_V$

clique W potential: $\psi_W^* = \frac{\phi_S^*}{\phi_S} \psi_W$

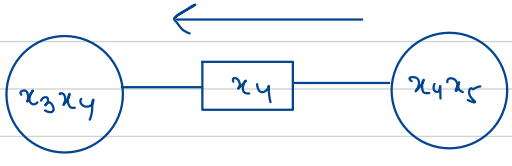
ϕ_S → (initial potential of the separator)

clique V potential: $\psi_V^* = \psi_V$ (remains same)

The above steps will be repeated till the algorithm reaches x_4x_5 (the last clique). This marks the end of the forward messaging.

After the step above, the potentials for all the cliques except the first one (the one that sent the message) and all the separators have been updated.

Then the backward messaging starts. In this case, it starts from x_4x_5 clique. Consider the messaging from x_4x_5 to x_3x_4.



$W = \{x_4, x_5\}$ $\quad V = \{x_3, x_4\}$ $\quad S = \{x_4\}$

while sending the message from x_4x_5 to x_3x_4, the potentials will be updated as:

$$\phi_S^{**} = \sum_{W \setminus S} \psi_W^{*}$$

$$\psi_V^{**} = \frac{\phi_S^{**}}{\phi_S^{*}} \psi_V^{*}$$

$$\psi_W^{**} = \psi_W^{*} \quad \text{(stays same)}$$

These steps happen till the algorithm reaches the first node, in this case x_1, x_2.

Now, they agree, have converged to consistent marginals.

$$\sum_{v/s} \psi_v^{**} = \sum_{v/s} \frac{\phi_s^{**}}{\phi_s^*} \psi_v^*$$

$$= \frac{\phi_s^{**}}{\phi_s^*} \sum_{v/s} \psi_v^*$$

$$= \phi_s^{**} = \sum_{w/s} \psi_w^{**}$$

Now, all the clique potentials are marginals and all the separator potentials are submarginals. $p(X)$ is unchanged by the message passing step.

In the end we normalize:

let $C \rightarrow$ cliques $\quad S \rightarrow$ separators

$$p(X_C) = \frac{\psi_C^{**}}{\sum_C \psi_C^*} \quad \forall C$$

$$p(X_S) = \frac{\phi_S^{**}}{\sum_S \phi_S^*} \quad \forall S$$

Flow of the code:

1. The first file Q4_calling_File.py is run, which has all the clique potentials.
2. The number of cliques are counted and stored in the variable 'm' in file junctionTreeAlgo.py.
3. Separator potentials are initialized as 1.
4. Forward messaging starts and goes on till the last clique.
5. The updated separator potentials after this become the initial separator potentials for back messaging.
6. Back messaging starts and goes on till the first clique.
7. Now the algorithm has converged, finally we normalize these potentials and obtain the marginals.

Test case given in the question and output of the code:

$$\psi(x_1, x_2) = \left\{ \begin{array}{c|cc} & x_2 = 0 & x_2 = 1 \\ \hline x_1 = 0 & 0.1 & 0.7 \\ x_1 = 1 & 0.8 & 0.3 \end{array} \right\}$$

$$\psi(x_2, x_3) = \left\{ \begin{array}{c|cc} & x_3 = 0 & x_3 = 1 \\ \hline x_2 = 0 & 0.5 & 0.1 \\ x_2 = 1 & 0.1 & 0.5 \end{array} \right\}$$

$$\psi(x_3, x_4) = \left\{ \begin{array}{c|cc} & x_4 = 0 & x_4 = 1 \\ \hline x_3 = 0 & 0.1 & 0.5 \\ x_3 = 1 & 0.5 & 0.1 \end{array} \right\}$$

$$\psi(x_4, x_5) = \left\{ \begin{array}{c|cc} & x_5 = 0 & x_5 = 1 \\ \hline x_4 = 0 & 0.9 & 0.3 \\ x_4 = 1 & 0.1 & 0.3 \end{array} \right\}$$

```
[[[0.04046243 0.44508671]
  [0.32369942 0.19075145]]

 [[0.26011561 0.10404624]
  [0.05780347 0.57803468]]

 [[0.11921965 0.19869942]
  [0.63945087 0.04263006]]

 [[0.56900289 0.18966763]
  [0.06033237 0.18099711]]]
```

Output:

$P(x_1, x_2)$	$x_2 = 0$	$x_2 = 1$	
$x_1 = 0$	0.04046243	0.44508671	0.48554914
$x_1 = 1$	0.32369942	0.19075145	0.51445087
	0.36416185	0.63583816	1.00000001 ≈ 1
			1.00000001 ≈ 1

$P(x_2, x_3)$	$x_3 = 0$	$x_3 = 1$	
$x_2 = 0$	0.26011561	0.10404624	0.36416185
$x_2 = 1$	0.25780347	0.57803468	0.63583815
	0.31791908	0.68208092	1
			1

$P(x_3, x_4)$	$x_4 = 0$	$x_4 = 1$	
$x_3 = 0$	0.11921965	0.19769942	0.31791907
$x_3 = 1$	0.63945087	0.04263006	0.68208093
	0.75867052	0.24132948	1
			1

$P(x_4, x_5)$	$x_5 = 0$	$x_5 = 1$	
$x_4 = 0$	0.56900289	0.18966713	0.75867052
$x_4 = 1$	0.06033237	0.18099711	0.24132948
	0.62933526	0.37066474	1
			1

All of these marginals sum up to 1.

Calling file:

```
import numpy as np
from junctionTreeAlgo import junction_tree

#here pot is initialized with potentials of the test case but can be
replaced by any other cliques potentials
pot =
np.array([[[0.1,0.7],[0.8,0.3]],[[0.5,0.1],[0.1,0.5]],[[0.1,0.5],[0.5,0.1]
],[[0.9,0.3],[0.1,0.3]]])
junction_tree(pot)
```

Algorithm File: junctionTreeAlgo.py

```
import numpy as np

def junction_tree(clique_pot):

    #calculating the size of the junction tree (number of cliques)
    m = np.size(clique_pot[0])

    #initializing separator potentials
    separator_pot_init = np.ones((3, 2))
    separator_pot_updated = np.zeros((3, 2))

    #forward message passing
    for i in range(m-1):
      separator_pot_updated[i] = np.sum(clique_pot[i], axis=0)
      sep_ratio = np.divide(separator_pot_updated[i],
separator_pot_init[i]).reshape(2,1)
      clique_pot[i+1] = np.multiply(sep_ratio, clique_pot[i+1])

    separator_pot_init_back = separator_pot_updated.copy()

    #backward message passing
    for i in range(m-2, -1 , -1):
      separator_pot_updated[i] = np.sum(clique_pot[i+1], axis=1)
      sep_ratio = np.divide(separator_pot_updated[i],
separator_pot_init_back[i]).reshape(1, 2)
      clique_pot[i] = np.multiply(sep_ratio, clique_pot[i])

    #normalizing
    for i in range(m):
```

```
        clique_pot[i] = np.divide(clique_pot[i], np.sum(clique_pot[i]))

    print(clique_pot)
```

Answer 5) Hidden Markov Model (HMM) is a statistical Markov model in which the system being modelled is assumed to be a Markov process with unobserved states.

Assumptions for HMM:

1) Given the state at time t, the state at time t+1 is independent to all previous states.

2) Given the state at time t, corresponding observation is independent to all other states.

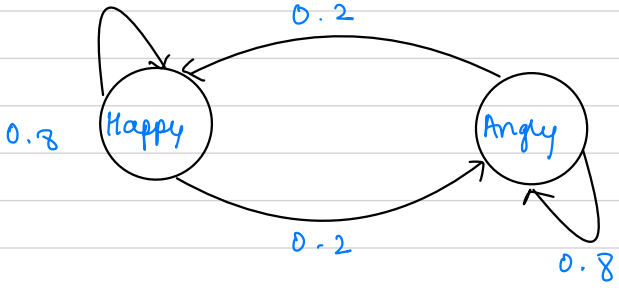
We need to addup the likelihood of the data given every possible series of hidden states. This process is split between 2 procedures, the forward and the backward.

In the forward procedure we use the given observations to calculate the probability of occurence of observation.

In the backward procedure we use these probabilities to espimate the hidden states.

Learning in HMM involves estimating the ~~state~~ transition probabilities A and output emmision probabilities B that make the observed

sequence most likely. For this purpose Expectation-Maximization algorithms are used.



Transition Matrix →

	Happy	Angry
Happy	0.8	0.2
Angry	0.2	0.8

Emmision Matrix →

	Smile	Frown	Laugh	Yell
Happy	0.4	0.1	0.3	0.2
Angry	0.1	0.4	0.2	0.3

O → Observations →

Day 1	Day 2	Day 3	Day 4	Day 5
Smile	Yell	Frown	Frown	Laugh

pdf can be given as:

$$P(q, y) = p(q_0) \prod_{t=1}^{T} p(q_t \mid q_{t-1}) \prod_{t=1}^{T} p(y_t \mid q_t)$$

There can be many sequences of {Happy, Angry} that lead to the observations. We need to maximize the joint probability of y and q.

We use the ArgMax Function Tree algorithm here. We run the algorithm but replace the sums with max and then find the biggest entry in separators. ITA has been explained in the previous question.

The most likely sequence of the emotional states of Mario is:

Day 1	Day 2	Day 3	Day 4	Day 5
Happy	Angry	Angry	Angry	Angry

Code output:

```
Super Mario's mood accross 5 days ['Happy', 'Angry', 'Angry', 'Angry', 'Angry']
```

HMM using Junction Tree Algorithm:

```
import numpy as np

#declaring all the matrices
transitionMat = np.array([[0.8,0.2], [0.2,0.8]])
emissionMat = np.array([[0.4,0.1,0.3,0.2],[0.1,0.4,0.2,0.3]])
observationMat = np.array([1,4,2,2,3])

#function for all the calculations of HMM
def argMax(transitionMat, emissionMat, observationMat):

    transtionProb = len(transitionMat)
    emissionProb = len(observationMat)
    psi = np.zeros((emissionProb,transtionProb,transtionProb))
    phi = np.ones((transtionProb,emissionProb))

    #forward
    for i in range(1,emissionProb):
        obs = observationMat[i]
        psi[i,:,:] = np.matmul(np.diag(phi[:,i-
1]),np.matmul(transitionMat,np.diag(emissionMat[:,obs-1])))
        phi[:,i] = np.amax(psi[i,:,:], axis = 0)

    #backward
    for i in reversed(range(emissionProb-2,0)):
        phiUpdated = np.amax(psi[i,:,:], axis = 0)
        psi[i,:,:] =
np.matmul(psi[i,:,:],np.diag(np.divide(phiUpdated,phi[:,i])))
        phi[:,i] = phiUpdated

    #Predicting mood
    moodMat = np.zeros(5)
    for i in range(5):
        moodMat[i] = np.argmax(phi[:,i])

    mood = list()
    for i in range(len(moodMat)):
        if (moodMat[i] == 0):
           mood.append('Happy')
        elif (moodMat[i] == 1):
           mood.append('Angry')

    print("Super Mario's mood accross 5 days", mood)
```

```
#function call
argMax(transitionMat, emissionMat, observationMat)
```