CSE 555
Assignment 2
Name:Rajiv Ranjan
Id:50249099

Problem 1: Draw the moral graph, triangulated graph and the junction tree. Explain why the "running intersection property" is satisfied in your junction tree.

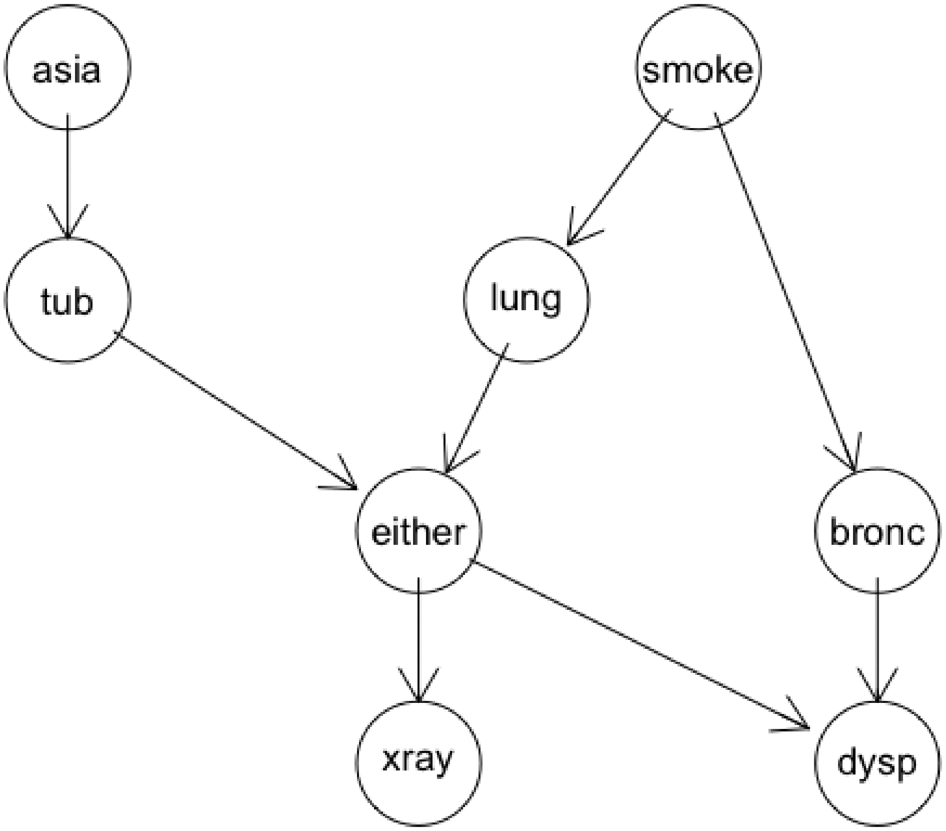
Solution 1:
Part1:
First of all, we try to do a network setup. One simple way of specifying a model for the chest clinic example is as follows. First we specify conditional probability tables with values as given in Lauritzen and Spiegelhalter (1988). This can be done with array() or as here with the cptable() function, which offers some additional features.

```
library(gRain)
require(gRbase)
require(Rgraphviz)
yn <- c("yes","no")
a <- cptable(~asia, values=c(1,99), levels=yn)
t.a <- cptable(~tub+asia, values=c(5,95,1,99), levels=yn)
s <- cptable(~smoke, values=c(5,5), levels=yn)
l.s <- cptable(~lung+smoke, values=c(1,9,1,99), levels=yn)
b.s <- cptable(~bronc+smoke, values=c(6,4,3,7), levels=yn)
e.lt<cptable(~either+lung+tub,values=c(1,0,1,0,1,0,0,1),levels=yn)
x.e <- cptable(~xray+either, values=c(98,2,5,95), levels=yn)
d.be<cptable(~dysp+bronc+either,values=c(9,1,7,3,8,2,1,9),levels=yn)
```

A network is created with the function grain() which returns an object of class grain:

```
plist <- compileCPT(list(a, t.a, s, l.s, b.s, e.lt, x.e, d.be))
grn1 <- grain(plist)
plot(grn1)
```

Output is as follows:



The compileCPT() function does some checking of the specified CPT’s. (For example, it is checked that the graph defined by the CPT’s is acyclic. Furthermore, the specification of t.a gives a table with four entries and the variable tub is specified to be binary. Hence it is checked that the variable asia is also binary.) The object plist is a list of arrays and it is from this list that the grain object is created.

2nd Step : Compilation—Finding the Clique Potentials

A grain object must be compiled and propagated before queries can be made. These steps are performed by the querygrain() function if necessary, but for some purposes it is advantageous to perform them explicitly. Compilation of a network is done with the compile() method for grain objects:

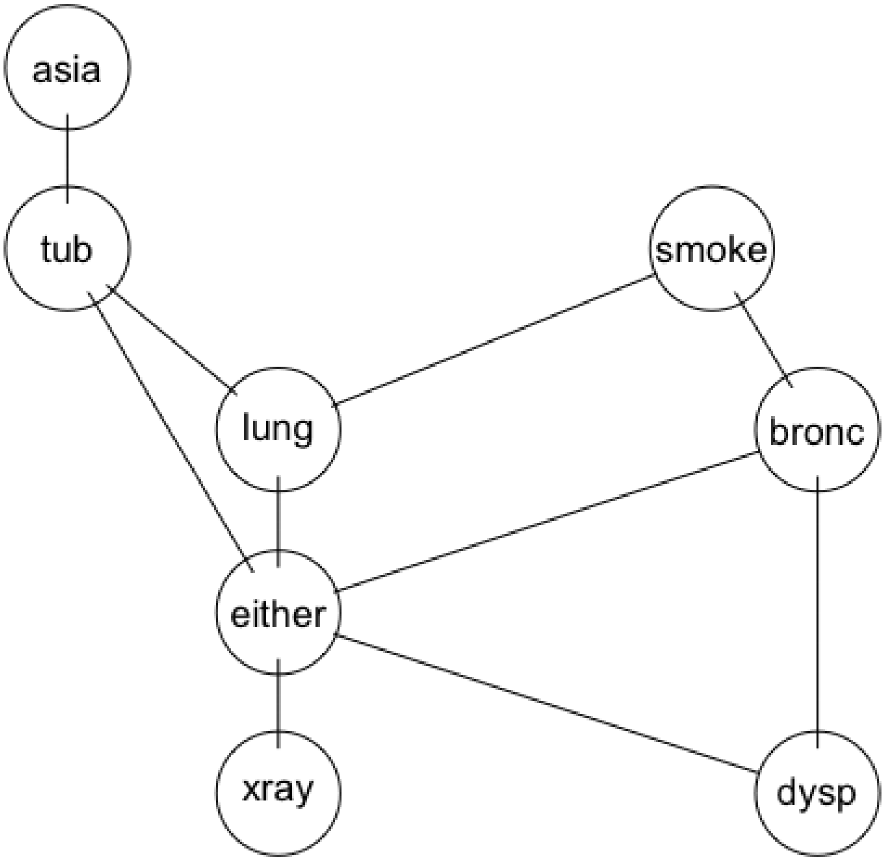
```
grn1c <- compile(grn1)
```

Compilation of a grain object based on CPTs involves the following steps: First it is checked whether the list of CPTs defines a directed acyclic graph (a DAG). If so, then the DAG is created; it is moralized and triangulated to form a chordal (triangulated) graph. The CPTs are transformed into clique potentials defined on the cliques of this chordal graph. The chordal graph together with the corresponding clique potentials are the most essential components of a grain object, and one may indeed construct a grain object directly from a specification of these two components.

We again consider the Bayesian network. The factorization
into a clique potential representation follows by simply noticing that in each of the conditional probability tables can be considered a function of the variables it involves. These potentials are simply non-negative functions. The dependence graph of the Bayesian network is derived from the potentials. For example, the presence of the term p(xD |xE,xB) implies that there must be edges between all pairs in {D,E,B}. Algorithmically, the dependence graph can be formed from the DAG by moralization: The moral graph of a DAG is obtained by first joining all parents of each node by a line and then dropping the directions on the arrows. For the chest clinic example, the edges between tub and lung, and between either and bronc are added.
The next step is to triangulate the dependence graph if it is not already so by adding additional edges, so-called fill-ins. This is done to enable simple computation of marginals from the clique potentials, cf. Finding an optimal triangulation (in terms of a minimal number of fill-ins) of a given graph is NP-complete, but various good heuristics exist. The gRbase package implements a Minimum Clique Weight Heuristic method inspired by Kjærulff (1990). Two possible fill-ins are the edge between lung and bronc, and the edge between
either and smoke. The triangulated graph is also a dependence graph for, the graph just conceals some conditional independence restrictions implied by the model. The steps described above can alternatively be carried out separately.

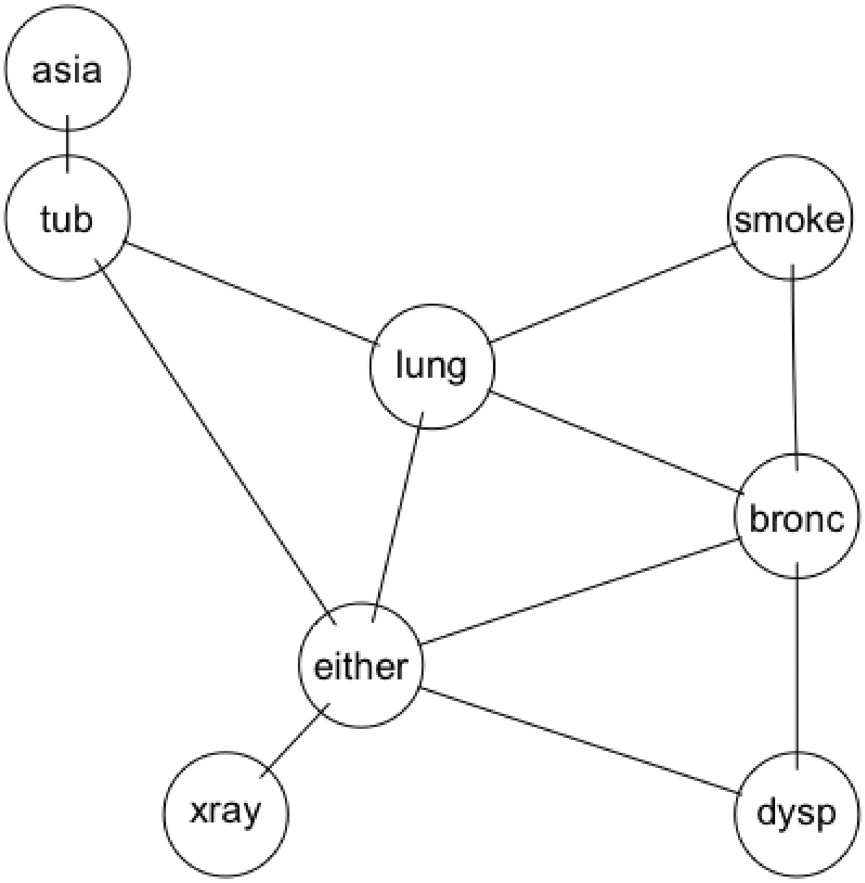
```
g <- grn1$dag
plot(g)
mg <- moralize(g)
plot(mg)
```

The output of the moralized graph is as follows:



```
tmg <- triangulate(mg)
plot(tmg)
```

The output of the finally triangulated graph is as follows:



Triangulated graph

An ordering C1,...,CT of the cliques of a graph is a RIP ordering if Sj = (C1 ∪ ⋯ ∪ Cj−1) ∩ Cj is contained in one (but possibly several) of the cliques C1,...,Cj−1, obtained with:

```
rip(tmg)
```

The output is as follows:

```
cliques
1 : tub asia
2 : either tub lung
3 : bronc lung either
4 : smoke lung bronc
5 : dysp bronc either
6 : xray either
```

```
separators
1 :
2 : tub
3 : lung either
4 : lung bronc
5 : bronc either
6 : either
```

```
parents
1 : 0
2 : 1
3 : 2
4 : 3
5 : 3
6 : 5
```

Picking a particular clique, say Ck, with Sj ⊆ Ck and naming this at the parentclique of Cj , with Cj being the child of Ck, organizes the cliques of the triangulated graph in a rooted tree with the cliques as nodes and arrows from parent to child. We call Sj the separator and Rj = Cj \Sj the residual, where S1 = ∅. The junction tree is formed by ignoring the root and the directions on the edges. It is a tree with the property that for any pair (A,B) of cliques and any clique C on the unique path between A and B it holds that A ∩ B ⊆ C. It can be shown that the cliques of a graph can be organized in a junction tree if and only if the graph is triangulated.

The junction tree can be displayed by plot(), where the numbers on the nodes refer to the clique numbers in the RIP-ordering. Other RIP-orderings of the cliques can be found by choosing an arbitrary clique as the first and then numbering the cliques in any way which is increasing as one moves outward from the first clique in this tree. For example C3,C2,C5,C1,C6,C4 would be another RIP-ordering.

The functions p(iv |ipa(v)) are hence defined on complete sets of the triangulated graph. For each clique C we collect the conditional probability tables p(iv |ipa(v)) into a single term ψC(iC) by multiplying them. Triangulation may have created cliques to which no CPT corresponds. For each such clique the corresponding potential is identically equal to 1. Thus we have obtained the clique potential representation of p(iV) as p(iV) =

product of(j=1(ψCj(iCj))). The representation is the fundamental representation for the subsequent
computations. As such, a DAG and a corresponding factorization as in is just one way of getting to the representation in and one may alternatively specify this directly as shall be illustrated.

```
grn1c <- compile(grn1)
summary(grn1c)
plot(grn1c,type="jt")
```

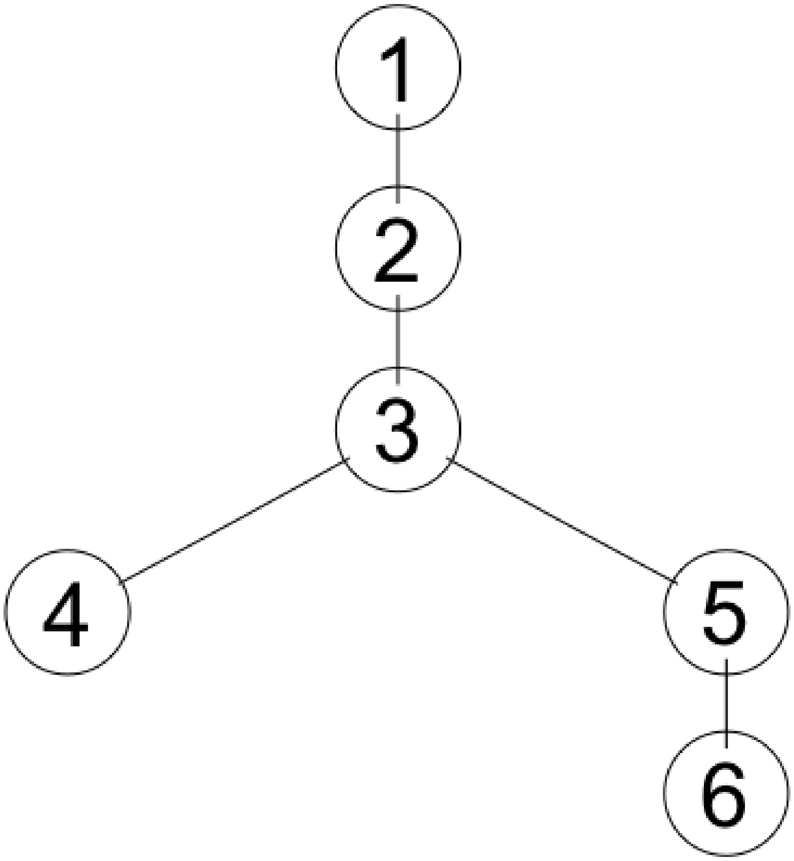


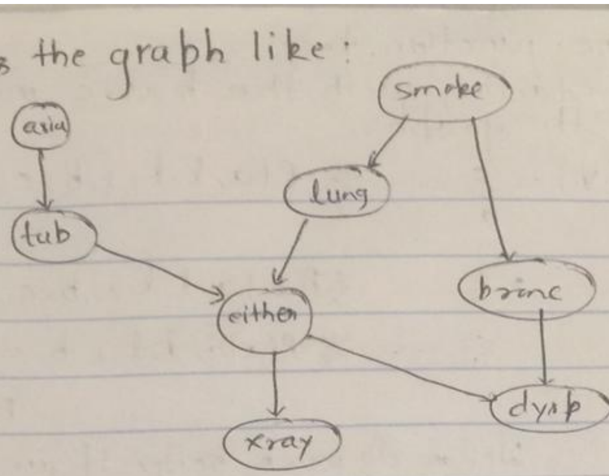
Fig: the junction tree as created by the moralized triangulated graph
Showing the different cliques in the graph

The cliques are as follows:

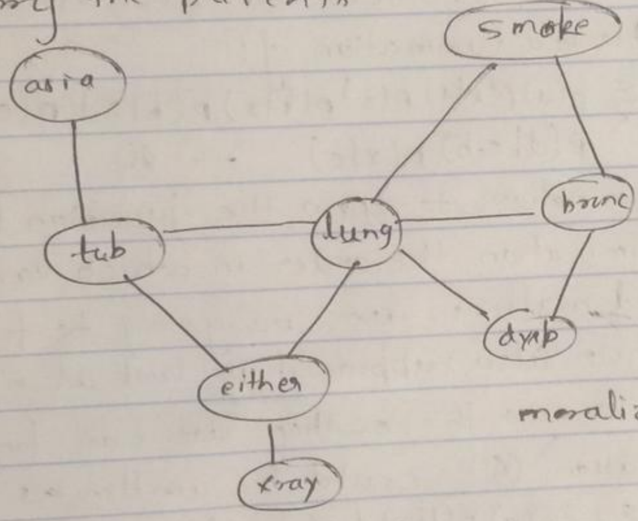
```
str( jTree( bnet$ug )$cliques )
```

```
List of 6
 $ : chr [1:2] "asia" "tub"
 $ : chr [1:3] "either" "lung" "tub"
 $ : chr [1:3] "either" "lung" "bronc"
 $ : chr [1:3] "smoke" "lung" "bronc"
 $ : chr [1:3] "either" "dysp" "bronc"
 $ : chr [1:2] "either" "xray"
```

Given is the graph like:

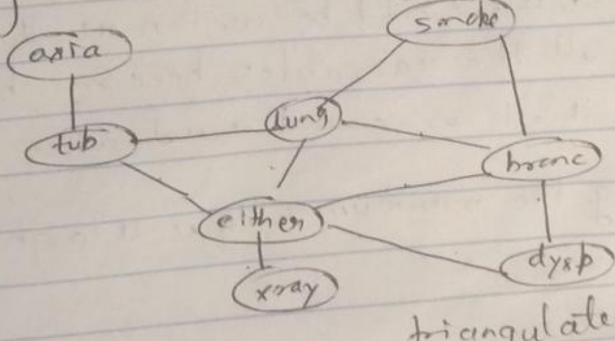


To create a moralized graph:-
remove the directed graph & make it undirected and then marry the parents



moralized graph

triangulated moralized graph is as follows



triangulated moralized graph

Drawing the junction tree

starting with the basic marginal form the graph.

$$P(V) = \sum_a \dots \sum_x P(a,t,l,s,b,e,d,x) \qquad 1$$

Let asia = a

tub = t

lung = l

smoke = s

bronc = b

either = e

dysp = d

xray = x

$$= \sum_a \dots \sum_x P(x|a,t,l,s,b,e,d) \cdot P(a,t,l,s,b,e,d)$$

$$= \sum_a \dots\dots \sum_x P(x|a,t,l,s,b,e,d) \cdot P(d|a,t,l,s,b,e) \dots\dots P(t|a)\, P(a)$$

We can do much better if we assume the conditional independence relations & this is why the notion of d-separation is so important, bcz, then I can simply write this in a summation of:

$$P(V) = \sum_a \dots \sum_x p(a)\, p(t|a)\, p(s)\, p(l|s)\, p(b|s)\, p(e|t,l)\, p(d|e,b)\, p(x|e) \quad \text{— (A)}$$

Using variable elimination to draw the junction tree.

In variable elimination the order in which variables are eliminated matters, we are going to follow a certain order here, also suppose if we look at which all dependencies are there, then we can find, that so, the equation (A) could be written as

$$\sum_t \dots \sum_x \left[\sum_a p(a)\, p(t|a)\right] p(s)\, p(l|s)\, p(b|s)\, p(e|t,l)\, p(d|e,b)\, p(x|e)$$

→ so, this could be written as the integration of all the variables here one by one.

this term could said as a function of $\Phi(t,a)$

Using the minimum Clique Weight heuristic

Using Variable Elimination

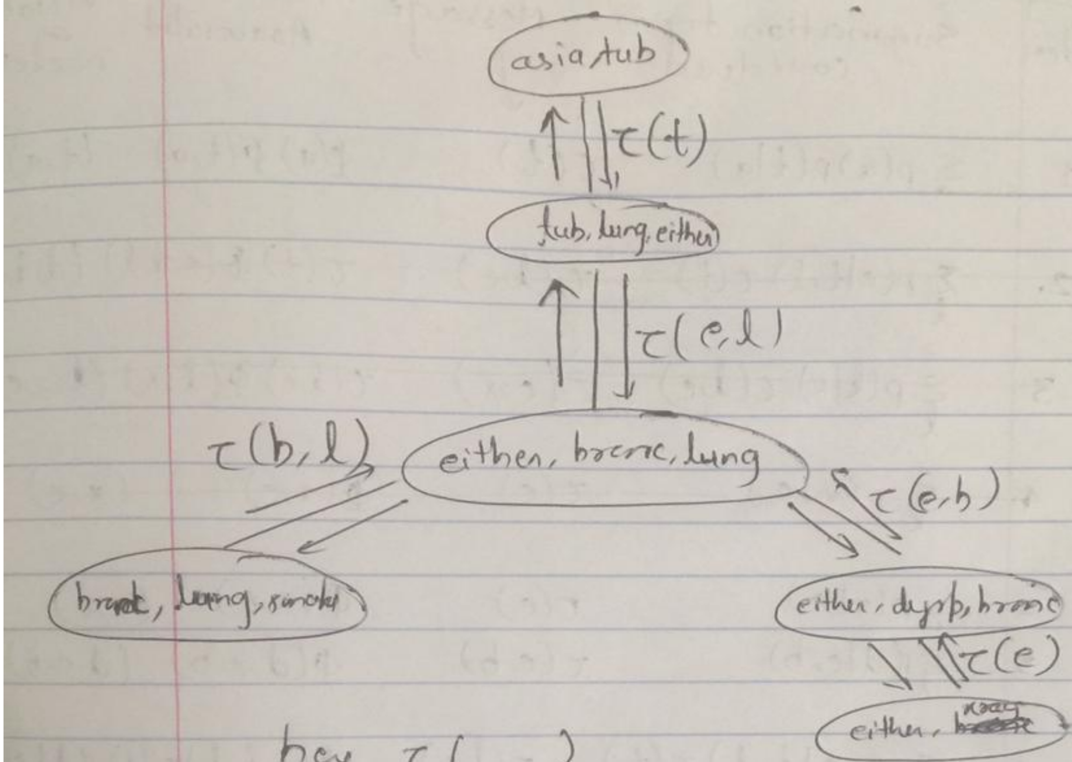
Order	Summation term considered	Message ↓	Factors Associated	Variabla co nodes
1	$\sum_a p(a)p(t\|a)$	$\tau(t)$	$\Phi(a)\Phi(t,a)$	(t,a)
~~2.~~	~~$\sum_t p(e\|t,l)\tau(t)$~~	~~$\tau(l,e)$~~	~~$\tau(t)\Phi(e,l,l)$~~	~~(l,l,e)~~
~~3~~	~~$\sum_l p(l\|s)\tau(l,e)$~~	~~$\tau(e,s)$~~	~~$\tau(l,e)\Phi(l,s)$~~	~~(l,s,e)~~
~~4~~	~~$\sum_x p(x\|e)$~~	~~$\tau(e)$~~	~~$\Phi(x,e)$~~	~~(x,e)~~
2.	$\sum_x p(x\|e)$	$\tau(e)$	$\Phi(x,e)$	(x,e)
3.	$\sum_d p(d\|e,b)$	$\tau(e,b)$	$\Phi(d,e,b)$	(d,e,b)
4.	$\sum_t p(e\|t,l)\tau(t)$	$\tau(l,e)$	$\Phi(e,l,l)\tau(t)$	(t,l,e)
5	$\sum_e \tau(e)\tau(e,b)\tau(l,e)$	$\tau(b,l)$	$\tau(e)\tau(e,b)\tau(l,e)$	(e,b,l)
6	$\sum_s p(s)p(l\|s)p(b\|s)$	$\tau(b,l)$	$\Phi(s)\Phi(l,s)$ $\Phi(b,s)$	(b,l,s)

how the terms are arranged

these are the messages passed

these are nodes

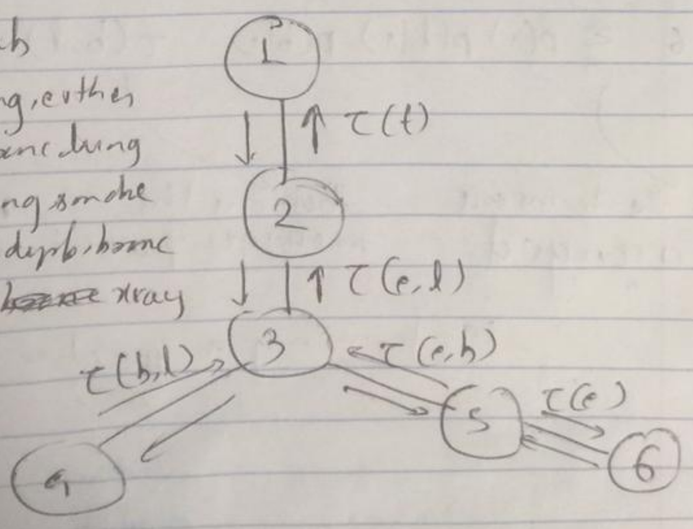
so, forming a junction tree using these



here τ( )
→ is the message passed.

↓ this could be written as

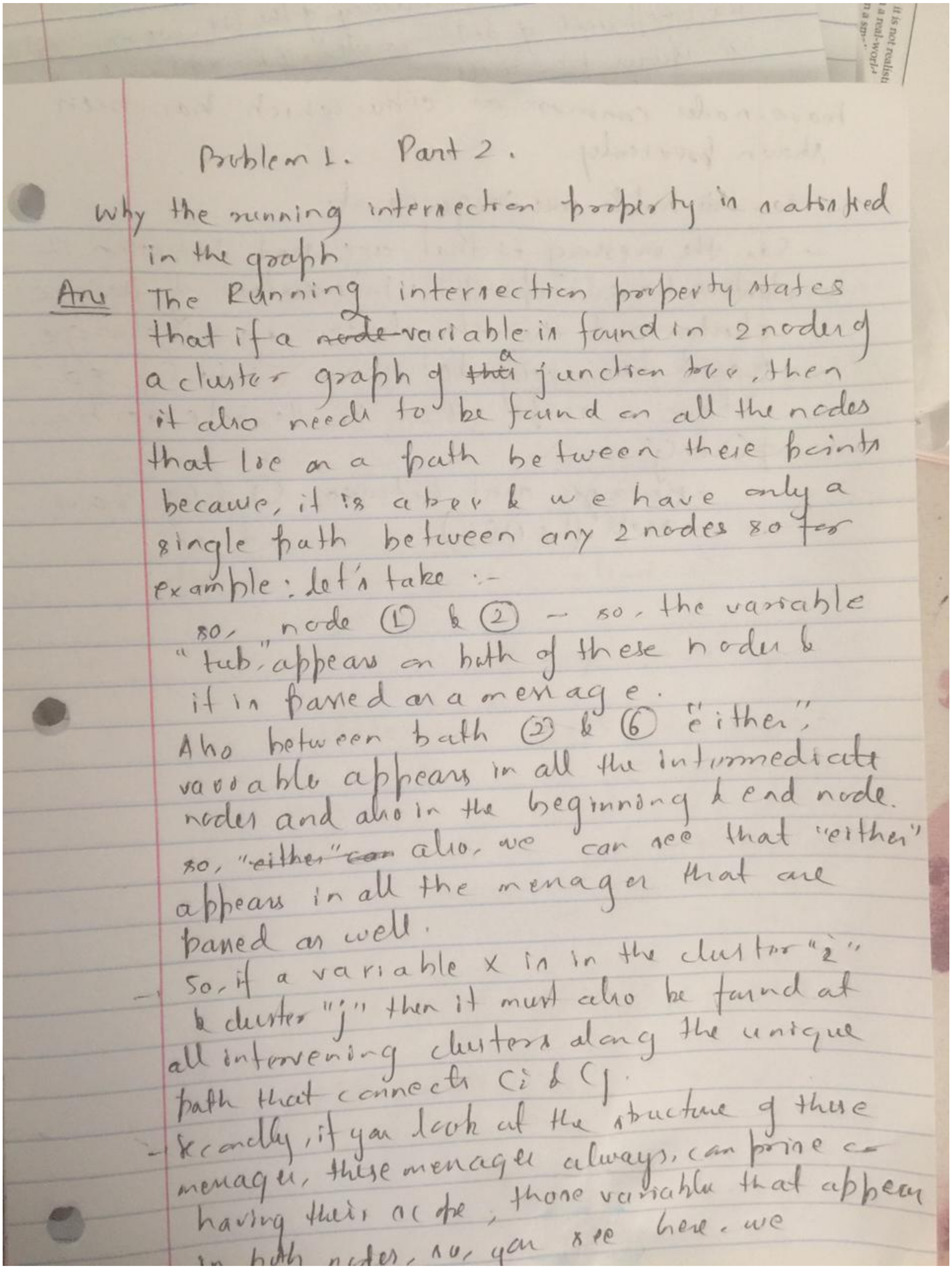
where 1 - asia, tub
2 → tub, lung, either
3 → either, bronc lung
4 bronc, lung smoke
5 → either, dysp, bronc
6 → either, xray



Problem 1 part2: Explain why the "running intersection property" is satisfied in your junction tree.

Solution part2

The Running Intersection Property is satisfied in the junction tree because it states that if a variable is found in 2 nodes of a cluster graph of a junction tree then it also needs to be found on all the nodes that lie on a path between these points because, it is a tree we have a single path between any 2 nodes. This has been explained below:



Problem 1. Part 2.

Why the running intersection property is satisfied in the graph

Ans The Running intersection property states that if a ~~node~~ variable is found in 2 nodes of a cluster graph of ~~the~~ a junction tree, then it also needs to be found on all the nodes that lie on a path between these points because, it is a tree & we have only a single path between any 2 nodes so for example: let's take :-

so, node ① & ② – so, the variable "tub" appears on both of these nodes & it is passed as a message.

Also between both ② & ⑥ "either", variable appears in all the intermediate nodes and also in the beginning & end node.

so, ~~"either" can~~ also, we can see that "either" appears in all the messages that are passed as well.

- So, if a variable x is in the cluster "i" & cluster "j" then it must also be found at all intervening clusters along the unique path that connects Ci & Cj.
- Secondly, if you look at the structure of these messages, these messages always, comprise of having their scope, those variables that appear in both nodes, so, you see here. we

have nodes common as either which has been shown previously.

→ So, this holds true in general.

→ So, the messages that are sent between 2 clusters have scope of the intersection of the one cluster and the other cluster, so a message is sent here, say between C_i to C_j, so I have a scope that in the intersection of C_i & C_j

- Messages sent between C_i & C_j have scope $(C_i \cap C_j)$

Problem 2: Describe how the different terms on the right hand side of "p(V) = p(a)p(t | a)p(s)p(l | s)p(b | s)p(e | t, l)p(d | e, b)p(x | e)" are distributed among the different juction tree clusters. Write out the messages using these terms and verify that the message passing algorithm indeed gives the cluster marginals.

Describe how the different terms on the right hand side of "p(V) = p(a)p(t | a)p(s)p(l | s)p(b | s)p(e | t, l)p(d | e, b)p(x | e)" are distributed among the different juction tree clusters.

Solution Part 1:

Problem 2

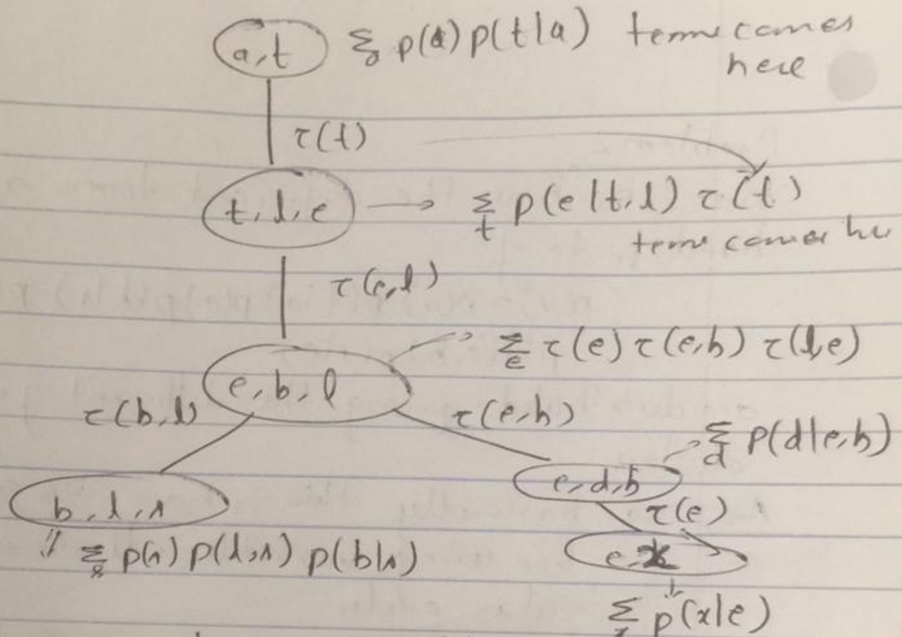
Describe how the different terms on the right hand side of

$$p(v) = p(a)\,p(t|a)\,p(s)\,p(l|s)\,p(b|s)\,p(e|t,l)\,p(d|e,b)\,p(x|e)$$

are distributed among the diffrent junction tree clusters.

Ans so, basically this is how we constructed the tree using variable elimination in this particular order

Order	Summation term	Message	Factors Associated	Variables or node
1	$\sum_a p(a)p(t\|a)$	$\tau(t)$	$\phi(a)\phi(t,a)$	(t,a)
2.	$\sum_x p(x\|e)$	$\tau(e)$	$\phi(x,e)$	(x,e)
3	$\sum_d p(d\|e,b)$	$\tau(e,b)$	$\phi(d,e,b)$	(d,e,b)
4	$\sum_t p(e\|t,l)\tau(t)$	$\tau(l,e)$	$\phi(e,t,l)\tau(t)$	(t,l,e)
5	$\sum_e \tau(e)\tau(e,b)\tau(l,e)$	$\tau(b,l)$	$\tau(e)\tau(e,b)\tau(l,e)$	(e,b,l)
6	$\sum_s p(s)p(l\|s)p(b\|s)$	$\tau(b,l)$	$\phi(s)\phi(l,s)$ $\phi(b,s)$	(b,l,s)


a,t
$\sum_a p(a) p(t|a)$ terms comes here
$\tau(t)$
t,l,e
$\sum_t p(e|t,l)\,\tau(t)$ terms comes here
$\tau(e,l)$
$\sum_e \tau(e)\,\tau(e,b)\,\tau(l,e)$
e,b,l
$\tau(b,l)$
$\tau(e,b)$
$\sum_d p(d|e,b)$
e,d,b
b,l,s
$\tau(e)$
$\sum_s p(s)\,p(l,s)\,p(b|s)$
e,x
$\sum_x p(x|e)$

The various terms on right hand side are associated in this way

Write down the messages using these terms

The Message are as follows

Bidirectional message sending from

[Messages]	[terms on right hand side]
$\tau(t) \leftarrow$	$\sum_a p(a)\,p(t|a)$

(a,t) ← (t,l,e)

from

(t,l,e) ⟷ (e,b,l) $\tau(e,l) \longleftarrow \sum_t p(e|t,l)\,\tau(t)$

from

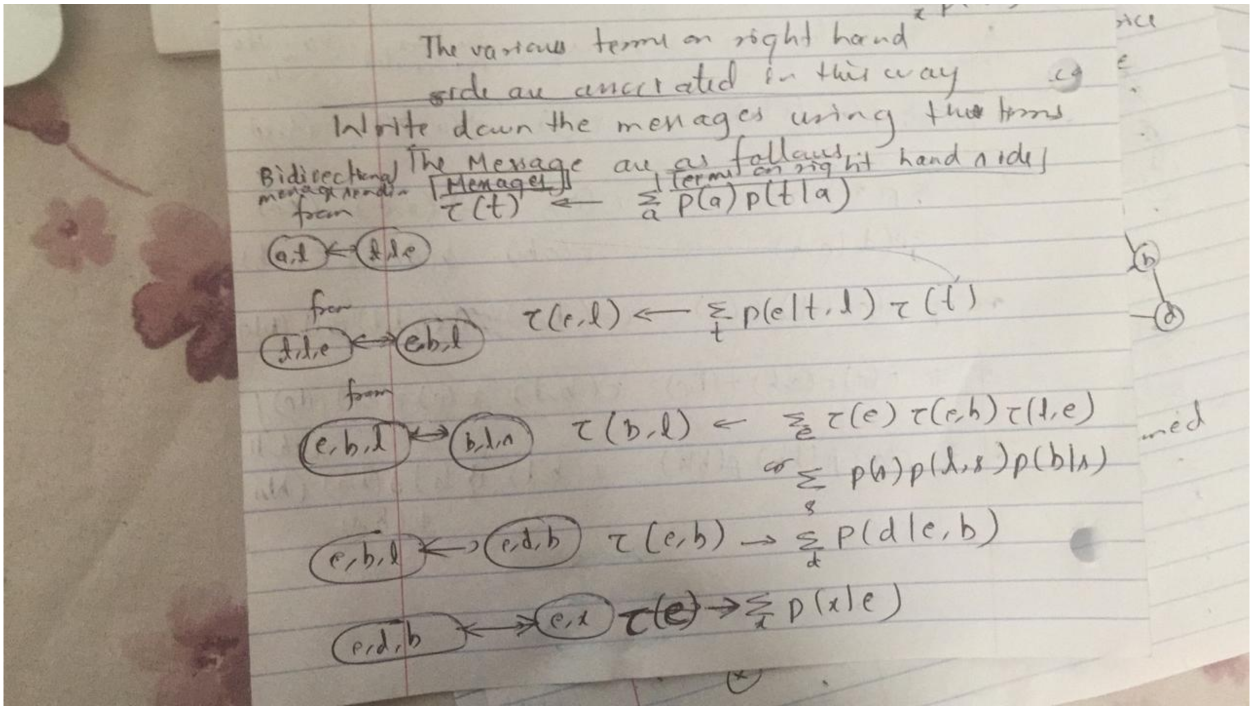
(e,b,l) ⟷ (b,l,s) $\tau(b,l) \leftarrow \sum_e \tau(e)\,\tau(e,b)\,\tau(l,e)$

or $\sum_s p(s)\,p(l,s)\,p(b|s)$

(e,b,l) ⟷ (e,d,b) $\tau(e,b) \rightarrow \sum_d p(d|e,b)$

(e,d,b) ⟷ (e,x) $\tau(e) \rightarrow \sum_x p(x|e)$

Write out the messages using these terms



The various terms on right hand side are associated in this way

Write down the messages using the terms

The Message are as follows [Terms on right hand side]

Bidirectional message sending from

[Message] $\tau(t) \leftarrow \sum_a p(a)p(t|a)$

(a,t) ← (t,l,e)

from

(t,l,e) ↔ (e,b,l) $\tau(e,l) \leftarrow \sum_t p(e|t,l)\tau(t)$

from

(e,b,l) ↔ (b,l,s) $\tau(b,l) \leftarrow \sum_e \tau(e)\tau(e,b)\tau(l,e)$

or $\sum_s p(s)p(l,s)p(b|s)$

(e,b,l) ↔ (e,d,b) $\tau(e,b) \rightarrow \sum_d p(d|e,b)$

(e,d,b) ↔ (e,x) $\tau(e) \rightarrow \sum_x p(x|e)$

Verify that the message passing algorithm indeed gives the cluster marginals.

Soln: To be able to answer queries, the grain object must be propagated, which means that the clique potentials must be calibrated (adjusted) to each other. Propagation is done with the propagate() method for grain object

grn1c <- propagate(grn1c)

The propagation algorithm works by turning the clique potential representation into a representation in which each potential ψ_{C_j} is replaced by the marginal distribution $p(i_{C_j})$. This representation is called a *clique marginal representation*. This is done by working twice through the set of cliques and passing 'messages' between neighbouring cliques: first from the last clique in the RIP-ordering towards the first, i.e. inwards in the junction tree, and subsequently passing messages in the other direction.

In detail, the process is as follows. We start with the last clique C_T in the RIP ordering where $C_T = S_T \cup R_T$, $S_T \cap R_T = \emptyset$. The factorization together with the factorization criterion implies that $R_T \perp\!\!\!\perp (C_1 \cup \cdots \cup C_{T-1}) \setminus S_T \mid S_T$. Marginalizing over i_{R_T} gives

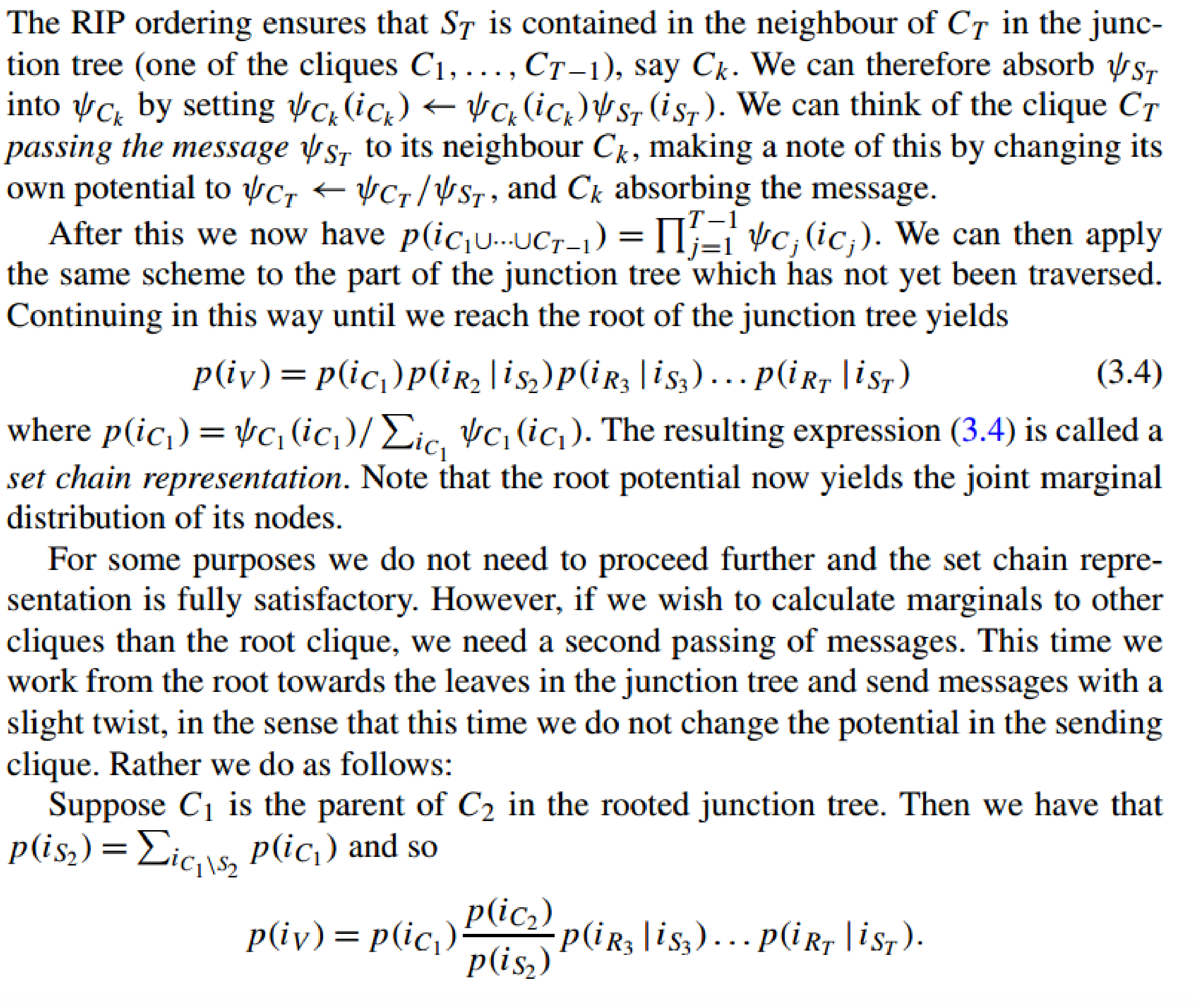
$$p(i_{C_1 \cup \ldots \cup C_{T-1}}) = \left(\prod_{j=1}^{T-1} \psi_{C_j}(i_{C_j}) \right) \sum_{i_{R_T}} \psi_{C_T}(i_{S_T}, i_{R_T}).$$

Let $\psi_{S_T}(i_{S_T}) = \sum_{i_{R_T}} \psi_{C_T}(i_{S_T}, i_{R_T})$. Then from the expression above we have

$$p(i_{R_T} \mid i_{S_T}) = \psi_{C_T}(i_{S_T}, i_{R_T}) / \psi_{S_T}(i_{S_T})$$

and hence

$$p(i_V) = p(i_{C_1 \cup \cdots \cup C_{T-1}}) p(i_{R_T} \mid i_{S_T}) = \left\{ \left(\prod_{j=1}^{T-1} \psi_{C_j}(i_{C_j}) \right) \psi_{S_T}(i_{S_T}) \right\} \frac{\psi_{C_T}(i_{C_T})}{\psi_{S_T}(i_{S_T})}.$$



The RIP ordering ensures that S_T is contained in the neighbour of C_T in the junction tree (one of the cliques $C_1, \dots, C_{T-1}$), say C_k. We can therefore absorb ψ_{S_T} into ψ_{C_k} by setting $\psi_{C_k}(i_{C_k}) \leftarrow \psi_{C_k}(i_{C_k})\psi_{S_T}(i_{S_T})$. We can think of the clique C_T *passing the message* ψ_{S_T} to its neighbour C_k, making a note of this by changing its own potential to $\psi_{C_T} \leftarrow \psi_{C_T}/\psi_{S_T}$, and C_k absorbing the message.

After this we now have $p(i_{C_1 \cup \dots \cup C_{T-1}}) = \prod_{j=1}^{T-1} \psi_{C_j}(i_{C_j})$. We can then apply the same scheme to the part of the junction tree which has not yet been traversed. Continuing in this way until we reach the root of the junction tree yields

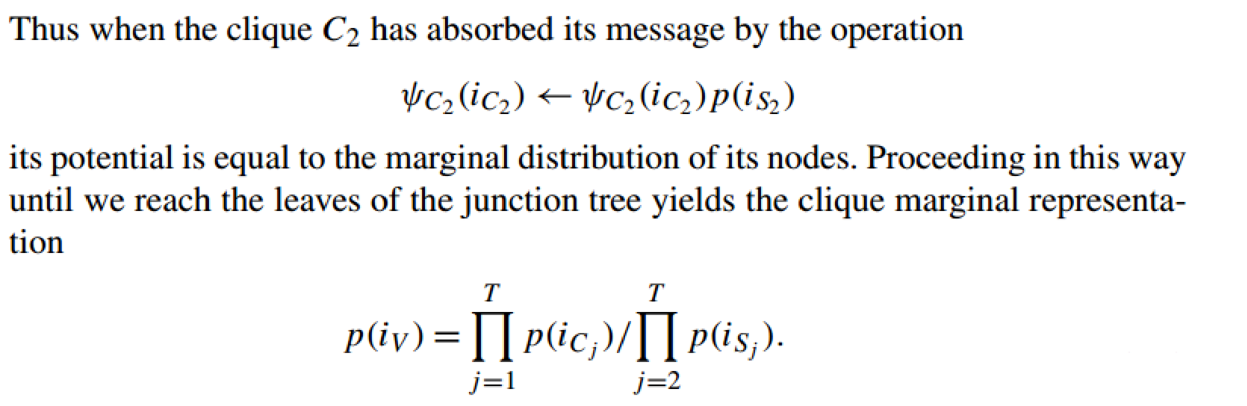
$$p(i_V) = p(i_{C_1})p(i_{R_2} \mid i_{S_2})p(i_{R_3} \mid i_{S_3}) \dots p(i_{R_T} \mid i_{S_T}) \tag{3.4}$$

where $p(i_{C_1}) = \psi_{C_1}(i_{C_1}) / \sum_{i_{C_1}} \psi_{C_1}(i_{C_1})$. The resulting expression (3.4) is called a *set chain representation*. Note that the root potential now yields the joint marginal distribution of its nodes.

For some purposes we do not need to proceed further and the set chain representation is fully satisfactory. However, if we wish to calculate marginals to other cliques than the root clique, we need a second passing of messages. This time we work from the root towards the leaves in the junction tree and send messages with a slight twist, in the sense that this time we do not change the potential in the sending clique. Rather we do as follows:

Suppose C_1 is the parent of C_2 in the rooted junction tree. Then we have that $p(i_{S_2}) = \sum_{i_{C_1 \setminus S_2}} p(i_{C_1})$ and so

$$p(i_V) = p(i_{C_1})\frac{p(i_{C_2})}{p(i_{S_2})}p(i_{R_3} \mid i_{S_3}) \dots p(i_{R_T} \mid i_{S_T}).$$



Thus when the clique C_2 has absorbed its message by the operation

$$\psi_{C_2}(i_{C_2}) \leftarrow \psi_{C_2}(i_{C_2})p(i_{S_2})$$

its potential is equal to the marginal distribution of its nodes. Proceeding in this way until we reach the leaves of the junction tree yields the clique marginal representation

$$p(i_V) = \prod_{j=1}^{T} p(i_{C_j}) / \prod_{j=2}^{T} p(i_{S_j}).$$

Hence proved and verified mathematically that message passing algorithm indeed gives the cluster marginals.

Problem:

Use message-passing algorithm to find the joint probability of "tub=yes, lung=yes, bronc=yes", given evidence that "asia=yes, xray=yes"?

Ans: We will use the junction tree constructed to find the joint probability of "tub=yes, lung=yes, bronc=yes", given evidence that "asia=yes, xray=yes".

Evidence is entered with setFinding() which creates a new grain object:
We set the evidence as asia=yes, xray=yes

```
gn1c.ev<setFinding(grn1c,nodes=c("asia","xray"),states=c("yes","yes"))
```

Then we can use the following query to find the joint probability of "tub=yes, lung=yes, bronc=yes", given evidence that "asia=yes, xray=yes" which is gn1c.ev

```
querygrain(grn1c.ev,nodes=c("lung","bronc",”tub”), type="joint")
```

The answer is : 0.01063804

However, if it is known beforehand that interest will be in the joint distribution of a specific set U of variables, one can ensure that the set U is contained in a single clique in the triangulated graph. This can for example be done by first moralizing, then adding edges between all nodes in U, and finally triangulating the resulting graph. The price to be paid is that the cliques may become larger and since computational complexity is exponential in the largest clique, this can be prohibitive.
But it would give us the graph that we need:

To do this in practice we first need to compile the grain again

```
grn1c2 <- compile(grn1, root=c("lung", "bronc", "tub"),
propagate=TRUE)
grn1c2.ev<setFinding(grn1c2,nodes=c("asia","xray"),states=c("yes","yes"))


querygrain(grn1c2.ev,nodes=c("tub","bronc","lung"), type="joint")
```

output is same: 0.01063804

But the process is much faster.

```
system.time({for (i in 1:50)
+ querygrain(grn1c.ev,nodes=c("lung","bronc","tub"),
+ type="joint")
+ })
user system elapsed
1.5 0.0 1.5
```

```
system.time({for (i in 1:50)
+ querygrain(grn1c2.ev,nodes=c("lung","bronc","tub"),
+ type="joint")
+ })
user system elapsed
0.02 0.00 0.01
```

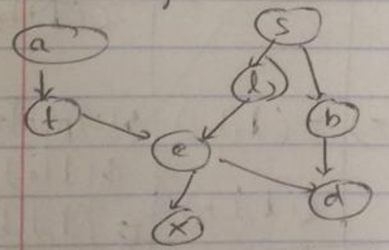
As we see the second step is much faster.
A detailed explanation is below:

Problem-2 ~~Verify that the message passing algorithm~~

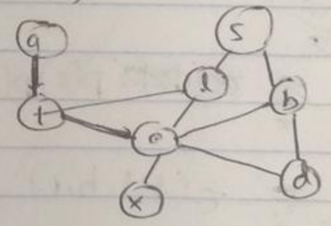
1 Use the message passing algorithm to find the joint probability of "tub = yes, lung = yes, bronc = yes", given evidence that "asia = yes, xray = yes.

Ans To calculate ~~it~~ we have to make sure that the variables tub, lung, bronc are in a single node of a i.e. a single clique in the triangulated graph. This can ~~be done~~ for example by first moralizing, then adding edges between all nodes in U, & then finally triangulating the resulting graph. The price to be paid is that cliques may become larger.

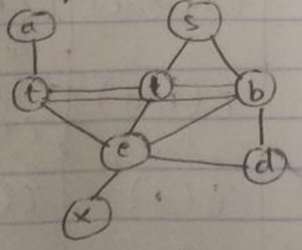
something like this



→ moralizing



→ triangulating l, t, b



a new graph is formed

So to find it we can eliminate other variables
to get a term in t, l, b

$$p(V) = \sum_a \sum_s \sum_t \sum_l \sum_e \sum_b \sum_x \sum_d p(a)\, p(t|a)\, \underline{p(s)}\, p(l|s)\, \underline{p(b|s)}\, \underline{p(e|t,l)}\, \underline{p(d|e,b)}\, \underline{p(x|e)}$$

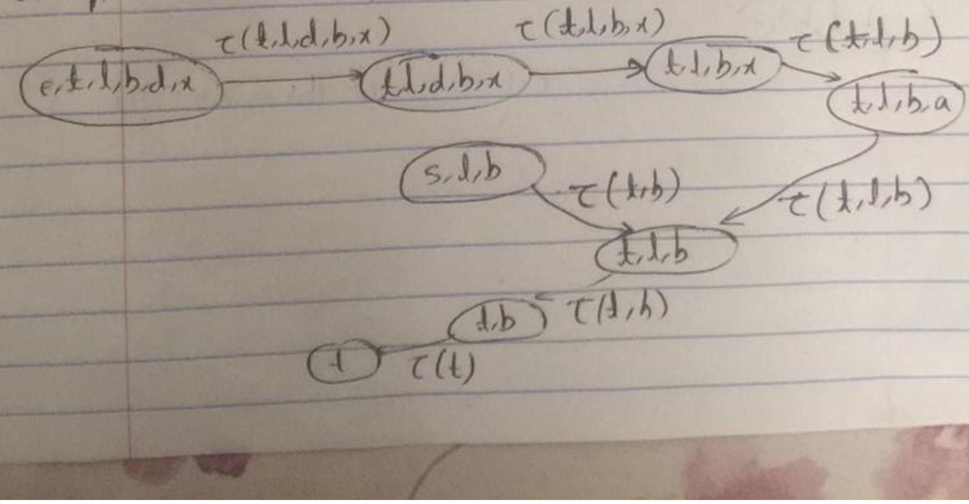
$\sum_e$:

$\sum_e p(e|t,l)\, p(d|e,b)\, p(x,e)$ $\quad \tau(t,l,d,b,x)$ $\quad \Phi(e,t,l)\, \Phi(d,e,b)\, \Phi(x,e)$ $\quad (e,t,l,d,b,x)$

$\sum_t \tau(t,l,d,b,x)$ $\quad \tau(t,l,b,x)$ $\quad \Phi(t,l,d,b,x)$ $\quad (t,l,d,b,x)$

$\sum_e p(e\mid t,l)\,p(d\mid e,b)\,p(x\mid e)$	$\tau(t,l,d,b,x)$	$\Phi(e,t,l)$ $\Phi(d,b,e)\,\Phi(x,e)$	(e,t,l,b,d,x)
$\sum_d \tau(t,l,d,b,x)$	$\tau(t,l,b,x)$	$\tau(t,l,d,b,x)$	(t,l,d,b,x)
$\sum_x \tau(t,l,b,x)$	$\tau(t,l,b)$	$\tau(t,l,b,x)$	(t,l,b,x)
$\sum_a \tau(t,l,b)\,p(a)\,p(t\mid a)$	$\tau(t,l,b)$	$\tau(t,l,b)\cdot$ $\Phi(a)\,\Phi(t,a)$	(t,l,b,a)
$\sum_s p(s)\,p(l\mid s)\,p(b\mid s)$	$\tau(l,b)$	$\Phi(s)\,\Phi(l,s)$ $\Phi(b,s)$	(s,l,b)
$\sum_l \tau(t,l,b)\,\tau(l,b)$	$\tau(t,b)$	$\tau(t,l,b)$ $\tau(l,b)$	(t,l,b)
$\sum_b \tau(t,b)$	$\tau(t)$	$\tau(t,b)$	(t,b)
$\sum_t \tau(t)$	1	$\tau(t)$	(t)

Graph could be as follows:



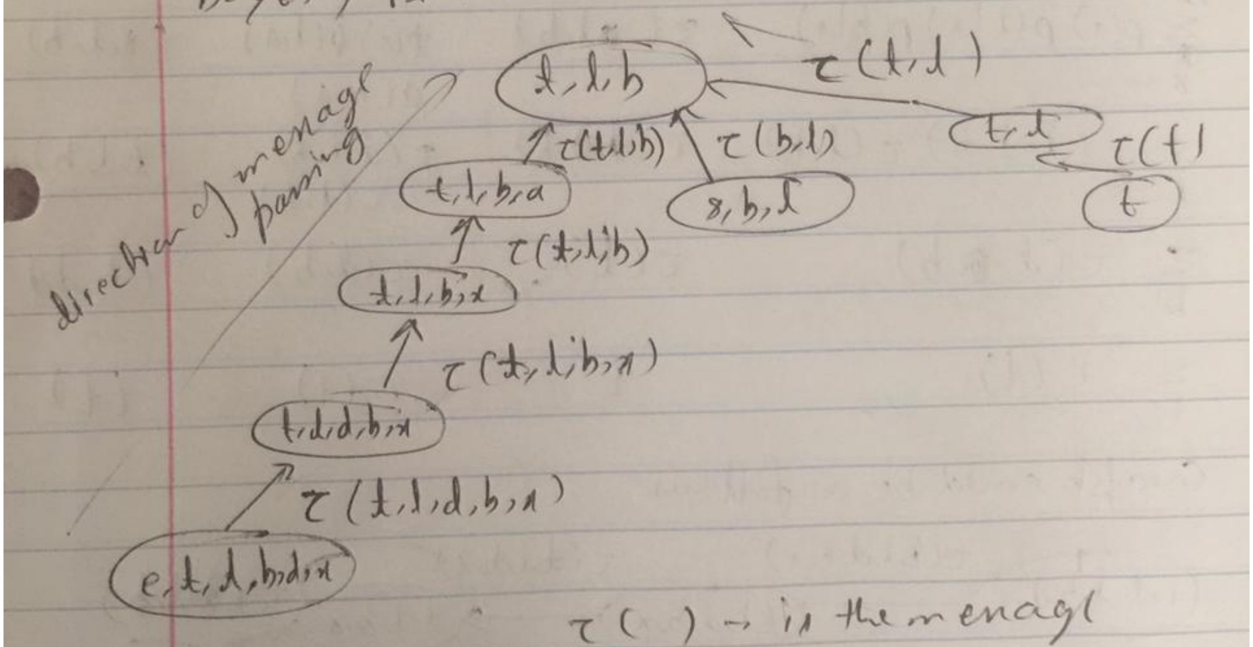
a=10, b=20

For Mid Term Review

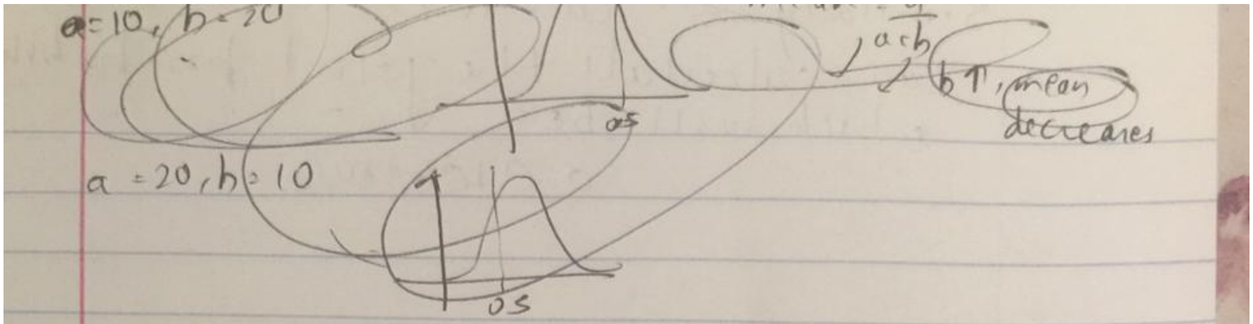
Linear Regression

- How do we learn weights for linear regression
- What does the bias term do?
- What is basic function expansion?
- What is robust regression?

The graph could be formed as follows, now to find 'tub=yes", "lung=yes" & bronc=yes' we make the node (t,l,b) as root & pass message to this node from all the directions & that will give the joint probability of P(t=yes, l=yes b=yes) i.e.



τ( ) → is the message



Using this graph & the messages which can be calculated as follow:-

$$\tau(t,l,d,b,x) = \sum_{e} p(e|t,l)\, p(d|e,b)\, p(x|e)$$

$$\tau(t,l,b,x) = \sum_{d} \tau(t,l,d,b,x)$$

$$\tau(t,l,b) = \sum_{x} \tau(t,l,b,x)$$

$$\tau(t,l,b) = \sum_{a} \tau(t,l,b)\, p(a)\, p(t|a)$$

$$\tau(l,b) = \sum_{s} p(s)\, p(l|s)\, p(b|s)$$

$$\tau(t,b) = \sum_{l} \tau(t,l,b)\, \tau(l,b)$$

$$\tau(t) = \sum_{b} \tau(t,b)$$

using these formulas, we can calculate the joint probability of "tub = yes, bronc = "yes" & lung = "yes" similaily, we can set evidence & then can, calculate the joint probability which will be

0.01063804