14.15 Consider the variable elimination algorithm in Figure 14.11 (page 528).
a. Section 14.4 applies variable elimination to the query P(Burglary | JohnCalls=true, MaryCalls=true) . Perform the calculations indicated and check that the answer is correct.
b. Count the number of arithmetic operations performed, and compare it with the number performed by the enumeration algorithm.
c. Suppose a network has the form of a chain: a sequence of Boolean variables X1, . . . ,Xn where Parents(Xi)={Xi−1} for i=2, . . . , n. What is the complexity of computing P(X1 | Xn=true) using enumeration? Using variable elimination?
d. Prove that the complexity of running variable elimination on a polytree network is linear in the size of the tree for any variable ordering consistent with the network structure.

```
function ELIMINATION-ASK(X, e, bn) returns a distribution over X
  inputs: X, the query variable
          e, observed values for variables E
          bn, a Bayesian network specifying joint distribution P(X1, ..., Xn)

  factors ← [ ]
  for each var in ORDER(bn.VARS) do
     factors ← [MAKE-FACTOR(var, e)|factors]
     if var is a hidden variable then factors ← SUM-OUT(var, factors)
  return NORMALIZE(POINTWISE-PRODUCT(factors))
```

Figure 14.11 The variable elimination algorithm for inference in Bayesian networks.

a. Section 14.4 applies variable elimination to the query P(Burglary | JohnCalls=true, MaryCalls=true) . Perform the calculations indicated and check that the answer is correct.

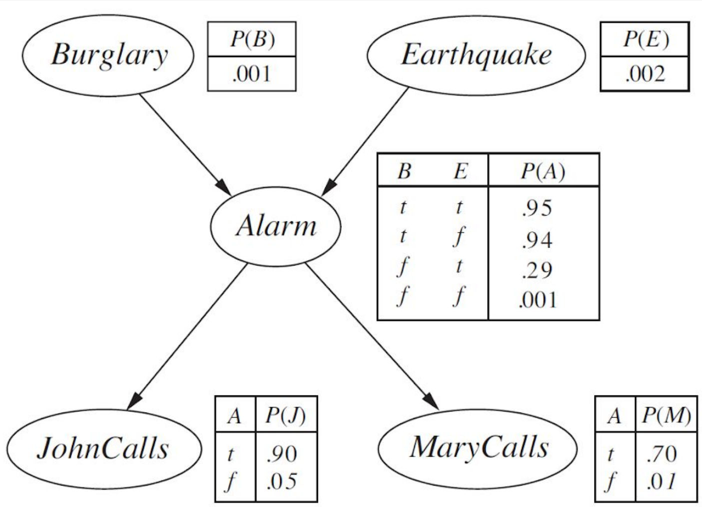

Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for $Burglary$, $Earthquake$, $Alarm$, $JohnCalls$, and $MaryCalls$, respectively.

This is what the algorithm does:

$$P(B \mid j, m) = \alpha \times P(B) \times \sum_e P(e) \times \sum_a P(a \mid B, e) \times P(j \mid a) \times P(m \mid a)$$

$$P(B \mid j, m) = \alpha \times f_1(B) \times \sum_e f_2(E) \times \sum_a f_3(A, B, E) \times f_4(A) \times f_5(A)$$

$$factors = f_1(B),\ f_2(E),\ f_3(A, B, E),\ f_4(A),\ f_5(A)$$

Right to Left

$f_5(A) = (P(m \mid a),\ P(m \mid \neg a)) = (0.70,\ 0.01)$

$f_4(A) = (P(j \mid a),\ P(j \mid \neg a)) = (0.90,\ 0.05)$

[TODO]: draw as 2x2x2 matrix

$f_3(A, B, E) = (P(a|b, e), P(\neg a|b, e), P(a|\neg b, e), P(\neg a|\neg b, e), P(a|b, \neg e), P(\neg a|b, \neg e), P(a|\neg b, \neg e), P(\neg a|\neg b, \neg e))$

Sum Out

$f_6(B, E) = \sum_a f_3(A, B, E) \times f_4(A) \times f_5(A)$

$f_6(B, E) = f_5(a) \times f_4(a) \times f_3(a, B, E) + f_5(\neg a) \times f_4(\neg a) \times f_3(\neg a, B, E)$

[TODO]: draw a 2x2 matrix

$f_6(a, B, E) = (P(a|b, e), P(a|\neg b, e), P(a|b, \neg e), P(a|\neg b, \neg e))$

$f_6(a, B, E) = (0.95,\ 0.29,\ 0.94,\ 0.001)$

$f_6(B, E) = 0.70 \times 0.90 \times (0.95,\ 0.29,\ 0.94,\ 0.001) + 0.01 \times 0.05 \times (0.05,\ 0.71,\ 0.06,\ 0.999)$

$f_6(B, E) = 0.63 \times (0.95,\ 0.29,\ 0.94,\ 0.001) + 0.0005 \times (0.05,\ 0.71,\ 0.06,\ 0.999)$

$f_6(B, E) = (0.5985000,\ 0.1827000,\ 0.5922000,\ 0.0006300) + (0.0000250,\ 0.0003550,\ 0.0000300,\ 0.0004995)$

$f_6(B, E) = (0.5985250,\ 0.1830550, 0.5922300,\ 0.0011295)$

Operations until here: 10 multiplications and 4 additions

Right to Left

$f_2(E) = (0.002,\ 0.998)$

Sum Out

$$P(B \mid j, m) = \alpha \times f_1(B) \times \sum_e f_2(E) \times f_6(B, E)$$

$f_7(B) = \sum_e f_2(E) \times f_6(B, E) = f_6(B, e) \times f_2(e) + f_6(B, \neg e) \times f_2(\neg e)$

$f_7(B) = f_6(B, e) \times f_2(e) + f_6(B, \neg e) \times f_2(\neg e)$

$f_7(B) = (0.5985250,\ 0.1830550) \times 0.002 + (0.5922300,\ 0.0011295) \times 0.998$

$f_7(B) = (0.00119705,\ 0.00036611) + (0.59104554, 0.001127241)$

$f_7(B) = (0.59224259,\ 0.001493351)$

Operations until here: 14 multiplications and 6 additions

Pointwise Product

$P(B \mid j, m) = \alpha \times f_1(B) \times f_7(B)$

$P(B \mid j, m) = \alpha \times (0.59224259,\ 0.001493351) \times (0.001,\ 0.999)$

$P(B \mid j, m) = \alpha \times (0.00059224259,\ 0.001491857649)$

Operations until here: 16 multiplications and 6 additions

Normalization:

$P(B \mid j, m) = (0.00059224259/\alpha^{-1},\ 0.001491857649/\alpha^{-1})$

$\alpha^{-1} = 0.00059224259 + 0.001491857649 = 0.002084100239$

$P(B \mid j, m) = (0.00059224259/0.002084100239,\ 0.001491857649/0.002084100239)$

$P(B \mid j, m) \simeq (0.28417,\ 0.71583)$

Operations until here: 16 multiplications, 7 additions and 2 divisions

b. Count the number of arithmetic operations performed, and compare it with the number performed by the enumeration algorithm.

After normalization process there are 16 multiplications, 7 additions and 2 divisions.

```
function ENUMERATION-ASK(X, e, bn) returns a distribution over X
  inputs: X, the query variable
          e, observed values for variables E
          bn, a Bayes net with variables {X} ∪ E ∪ Y   /* Y = hidden variables */

  Q(X) ← a distribution over X, initially empty
  for each value x_i of X do
      Q(x_i) ← ENUMERATE-ALL(bn.VARS, e_{x_i})
          where e_{x_i} is e extended with X = x_i
  return NORMALIZE(Q(X))
---------------------------------------------------------------
function ENUMERATE-ALL(vars, e) returns a real number
  if EMPTY?(vars) then return 1.0
  Y ← FIRST(vars)
  if Y has value y in e
      then return P(y | parents(Y)) × ENUMERATE-ALL(REST(vars), e)
      else return Σ_y P(y | parents(Y)) × ENUMERATE-ALL(REST(vars), e_y)
          where e_y is e extended with Y = y
```

Figure 14.9 The enumeration algorithm for answering queries on Bayesian networks.

The enumeration algorithm will use:

$$P(B \mid j, m) = \alpha \times \langle P(b) \sum_e \sum_a P(j, m, e, a);\ P(\neg b) \sum_e \sum_a P(j, m, e, a) \rangle$$

Left term (when B=true)

$$P(b \mid j, m) = \alpha \times P(b) \times \sum_e P(e) \times \sum_a P(a \mid b, e) \times P(m|a) \times P(j|a)$$

This will use 11 multiplications and 3 additions

Right term (when B=false)

$$P(\neg b \mid j, m) = \alpha \times P(\neg b) \times \sum_e P(e) \times \sum_a P(a \mid \neg b, e) \times P(m|a) \times P(j|a)$$

This will require to add 11 multiplications and 3 more additions operations.

Normalization will add 1 more addition and 2 division operations. Then, the whole algorithm will use 22 multiplications, 7 additions and 2 division operations.

c. Suppose a network has the form of a chain: a sequence of Boolean variables X1, . . . , Xn where Parents(Xi)={Xi−1} for i=2, . . . , n. What is the complexity of computing P(X1 | Xn=true) using enumeration? Using variable elimination?

The enumeration algorithm will open two branches for possible values of X1:

$$P(X_1 \mid X_n = True) = \alpha \times < P(X_1 = True \mid X_n = True); P(X_1 = False \mid X_n = True) >$$

$$Left\ term = P(x_1) \times \sum_{x_2} P(x_2 | x_1) \times \sum_{x_3} P(x_3 | x_2) \times \ldots \times \sum_{x_{n-1}} P(x_{n-1} | x_{n-2}) \times P(x_n \mid x_{n-1})$$

$$Right\ term = P(\neg x_1) \times \sum_{x_2} P(x_2 | \neg x_1) \times \sum_{x_3} P(x_3 | x_2) \times \ldots \times \sum_{x_{n-1}} P(x_{n-1} | x_{n-2}) \times P(x_n \mid x_{n-1})$$

This will require to do 2^{n-2} multiplications for each branch, requiring a total of 2^{n-1} together. So we can say that the complexity of computing $P(X_1 \mid X_n = True)$ is $O(2^n)$.

The elimination algorithm will create factors that will work as a cache from right to left:

$$P(X_1 \mid X_n = True) = \alpha \times P(X_1) \times \sum_{x_2} P(x_2 | X_1) \times \sum_{x_3} P(x_3 | x_2) \times \ldots \times \sum_{x_{n-1}} P(x_{n-1} | x_{n-2}) \times P(x_n \mid x_{n-1})$$

$$P(X_1 \mid X_n = True) = \alpha \times f_1(X_1) \times \sum_{x_2} f_2\,(X_2) \times \sum_{x_3} f_3\,(X_3) \times \ldots \times \sum_{x_{n-1}} f_{n-1}\,(X_{n-1}) \times f_n(X_n)$$

Right to Left: Sum Out

$$f_{n+1} = \sum_{x_{n-1}} f_{n-1}\,(X_{n-1}) \times f_n(X_n)$$

$$P(X_1 \mid X_n = True) = \alpha \times f_1(X_1) \times \sum_{x_2} f_2\,(X_2) \times \sum_{x_3} f_3\,(X_3) \times \ldots \times f_{n+1}$$

Right to Left: Sum Out

$$f_{n+2} = \sum_{x_{n-2}} f_{n-2}\,(X_{n-2}) \times f_{n+1}$$

$$P(X_1 \mid X_n = True) = \alpha \times f_1(X_1) \times \sum_{x_2} f_2\,(X_2) \times \sum_{x_3} f_3\,(X_3) \times \ldots \times f_{n+2}$$

…

Right to Left: Sum Out

$$f_{n+(n-3)} = \sum_{x_{n-(n-3)}} f_{n-(n-3)}\,(X_{n-(n-3)}) \times f_{n+(n-4)} = \sum_{x_3} f_3(X_3) \times f_{n+(n-4)}$$

$$P(X_1 \mid X_n = True) = \alpha \times f_1(X_1) \times \sum_{x_2} f_2\,(X_2) \times f_{n+(n-3)}$$

Right to Left: Sum Out

$$f_{n+(n-2)} = \sum_{x_{n+(n-2)}} f_{n+(n-2)}\,(X_{n+(n-2)}) \times f_{n+(n-3)} = \sum_{x_2} f_2(X_2) \times f_{n+(n-3)}$$

$$P(X_1 \mid X_n = True) = \alpha \times f_1(X_1) \times f_{n+(n-2)}$$

Right to Left:

$$f_{n+(n-1)} = f_{n+(n-1)}\,(X_{n+(n-1)}) \times f_{n+(n-2)} = f_1(X_1) \times f_{n+(n-2)}$$

$$P(X_1 \mid X_n = True) = \alpha \times f_{n+(n-1)}$$

Before normalization process the variable elimination algorithm calculate (n - 1) factors. So, we can say that this algorithm presents a computing linear complexity: $O(n)$

d. Prove that the complexity of running variable elimination on a polytree network is linear in the size of the tree for any variable ordering consistent with the network structure.

[TODO]: Find a nice graphical representation

A polytree is a kind of network where there is at most one undirected path between any two nodes in the network. By ignoring the direction of the edges in a polytree, we get an undirected tree T. Choose any one of the nodes as the root of T and let V be a reverse topological ordering of the nodes in T (i.e. all children appear before parents). We will choose V as our variable elimination ordering. [Note that when we talk about the parent of a node in T, this can be different from a parent of a node in the BN] Observe that when a node X is eliminated, it has no children in T left in the ordering V . Therefore its only remaining neighbour is its parent in T, that we can call Y. Thus the factor corresponding to X will be a function of Y alone and can be computed in time O(|Dom(X)|*|Dom(Y)|). If X is a child of Y in the BN, then clearly this step takes time at most linear in the size of the CPT for X. If X is a parent of Y in the BN, then it could be the case that the CPT for X is smaller than |Dom(X)|*|Dom(Y)|, but we argue as follows: suppose the other parents of Y in the BN are X1, X2, . . . , Xk. Then the total running time for eliminating X, X1, . . . , Xk is:
(|Dom(X)| + |Dom(X1)| + . . . + |Dom(Xk)|)*|Dom(Y)|. However the CPT for Y in the original BN will have size |Dom(X)|*|Dom(X1)|* . . . *|Dom(Xk)|*|Dom(Y)|. Therefore, charging the running time of elimination of the parents to the CPT of the child, still gives sublinear complexity in the size of the tree. The new CPT for Y will present a new size no more than |Dom(X1)|* . . . *|Dom(Xk)|*|Dom(Y)|.