# Bayesian Networks: Representation, Variable Elimination

CS 6375: Machine Learning
Class Notes
Instructor: **Vibhav Gogate**
The University of Texas at Dallas

We can view a Bayesian network as a compact representation of a joint probability distribution (the computational view). We can also view it as a graphical representation of a set of conditional independence statements (the conditional independence view). As it turns out, the two views are equivalent. (For more details please refer to [1] and [2]). In this class, we will focus exclusively on the computational view. Namely, we will look at a Bayesian network as a polynomial representation of a multi-dimensional (typically 1000s of dimensions) joint probability distribution.

## 1 Computational View

A Bayesian network is a directed acyclic graph (DAG)[1] $G(\mathbf{X}, \mathbf{E})$ where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of random variables and $\mathbf{E}$ is a set of directed edges. A node $X_j$ is a parent of node $X_i$ if there is a directed edge from $X_j$ to $X_i$. We will denote by $pa(X_i)$, the set of parents of $X_i$. Let $D(X_i)$ denote the set of finite values that $X_i$ can take, i.e., the domain of $X_i$. For example, if $X_i$ is a binary variable, then $D(X_i) = \{True, False\}$. Each node in a Bayesian network is associated with a conditional probability table or CPT, whose functional form is $P(X_i|pa(X_i))$ (i.e., variable given its parents). A Bayesian network $B$ represents the following joint probability distribution

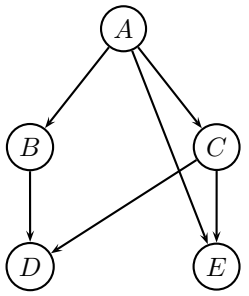
$$\Pr(\mathbf{x}) = \prod_{i=1}^{n} P(x_i|\pi(\mathbf{x}, pa(X_i))) \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_n)$ is an assignment to all variables in $\mathbf{X}$ and $\pi(\mathbf{x}, pa(X_i))$ is the projection of $\mathbf{x}$ on the parents of $X_i$. (For instance, the projection of the assignment $(X_1 = 0, X_2 = 0, X_3 = 1)$ on $\{X_1, X_3\}$ is the assignment $(X_1 = 0, X_3 = 1)$).

**Example 1.** Figure 1 shows a Bayesian network over five binary variables $\{A, B, C, D, E\}$ and its CPTs. From Eq. (1), we can see that the probability of an assignment to all random variables can be calculated by multiplying the probability values obtained by projecting the assignment on each CPT. For example, the probability of $(A = 0, B = 0, C = 0, D = 1, E = 0)$ is $0.6 \times 1 \times 0.2 \times 0.8 \times 1 = 0.096$.

**Exercise 1.** Compute the probability of $(A = 1, B = 1, C = 0, D = 0, E = 0)$ and $(A = 0, B = 1, C = 1, D = 0, E = 0)$.

[1] A DAG is a directed graph that has no directed cycles. Namely, if you follow the arrows, you should not be able to come back to the same point.




(a)

P(A)

| A=0 | A=1 |
|---|---|
| .6 | .4 |

P(B|A)

| A | B=0 | B=1 |
|---|---|---|
| 0 | 1 | 0 |
| 1 | .1 | .9 |

P(C|A)

| A | C=0 | C=1 |
|---|---|---|
| 0 | .2 | .8 |
| 1 | .7 | .3 |

P(D|B,C)

| B | C | D=0 | D=1 |
|---|---|---|---|
| 0 | 0 | .2 | .8 |
| 0 | 1 | .1 | .9 |
| 1 | 0 | .3 | .7 |
| 1 | 1 | .5 | .5 |

P(E|A,C)

| A | C | E=0 | E=1 |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 1 | .5 | .5 |
| 1 | 0 | .7 | .3 |
| 1 | 1 | 0 | 1 |

(b)

Figure 1: A Bayesian network and its CPTs

**Space Complexity** The space required by a CPT is exponential in the number of its parents. Assuming that the Bayesian network has $n$ variables, the space required to store it is $O(n \sum_i \exp(|pa(X_i)| + 1))$. Thus, if the number of parents for each node is bounded by a constant, the Bayesian network can be specified in polynomial space.

Forget the $O$-notation for the time being. Let us calculate how much memory units are actually needed to represent a CPT on our computer. Assume that we need one memory unit to store a real-number. Then, the number of memory units required to represent a CPT $P(X_i|pa(X_i))$ is:

$$M[P(X_i|pa(X_i)] = (|D(X_i)| - 1) \prod_{X_j \in pa(X_i)} |D(X_j)| \tag{2}$$

where $|D(X_i)|$ denotes the cardinality of $D(X_i)$, i.e., the domain size of $X_i$. How did we arrive at this number? Notice that to represent a marginal distribution over $k$ points, we have to specify probabilities over some arbitrary selected $k-1$ points only; we can calculate the probability of the $k$-th point by subtracting the sum of the known probabilities over the $k-1$ points from 1. Thus, to represent a marginal distribution over a variable $X_i$, we need $|D(X_i)| - 1$ memory units. This gives us the first term in Equation (2). To specify a CPT for a variable, we have to represent a conditional marginal distribution for each possible assignment to its parents. The number of possible assignments to the parents equals the product of the domain sizes of the variables in the parent set. This gives us the second term in Equation (2).

The number of memory units required to represent a Bayesian network $B$ is simply a sum over the memory units required by each of its CPTs:

$$M[B] = \sum_i M[P(X_i|pa(X_i))]$$

**Example 2.** The number of memory units required to represent the CPT $P(C|A)$ in Figure 1 is 2.

**Exercise 2.** Calculate the number of memory units required for representing the remaining CPTs in Figure 1. Also, calculate the number of memory units required to represent the Bayesian network.

## 2 Common Queries over Bayesian networks

Since a Bayesian network represents a joint probability distribution (i.e., complete knowledge about the world), it can be used to answer any query defined over the random variables. Some common query types are:

- **Probability of Evidence (PE)** We define evidence as an assignment to a subset of variables in the Bayesian network. Given evidence, the aim of this task, as the name suggests, is to compute the probability of evidence. Formally, let $\mathbf{E}$ be the set of evidence variables, $\mathbf{E} = \mathbf{e}$ be the evidence and let $\mathbf{Z} = \mathbf{X} \setminus \mathbf{E}$ be the set of non-evidence variables.

$$\Pr(\mathbf{e}) = \sum_{\mathbf{z} \in D(\mathbf{Z})} \Pr(c(\mathbf{z}, \mathbf{e})) = \sum_{\mathbf{z} \in D(\mathbf{Z})} \prod_{i=1}^{n} P(\pi(c(\mathbf{z}, \mathbf{e}), \{X_i\}) | \pi(c(\mathbf{z}, \mathbf{e}), pa(X_i)) \tag{3}$$

  where $c(\mathbf{z}, \mathbf{e})$ denotes the composition of the assignments $\mathbf{z}$ and $\mathbf{e}$ and $D(\mathbf{Z})$ is the Cartesian product of domains of all variables in $\mathbf{Z}$.

  We have to use cumbersome notation in Equation 3 in order to be mathematically precise. To better understand the equation, recall that the probability of an event is a sum over all possible assignments in the joint distribution that are consistent with the event. To calculate it, we fix each evidence variable to the given value and sum over all possible assignments to the non-evidence variables.

- **Posterior Marginal probabilities (MAR)** Given evidence $\mathbf{e}$ and a variable $X_i$, the aim here is to compute $\Pr(X_i = x_i | \mathbf{e})$, which is defined as:

$$\Pr(x_i | \mathbf{e}) = \frac{\Pr(c(x_i, \mathbf{e}))}{\Pr(\mathbf{e})} \tag{4}$$

  Since the posterior marginal probability is a ratio of two probabilities, algorithms used for computing probability of evidence can be used to compute it.

- **Most Probable explanation (MPE)** Given evidence $\mathbf{e}$, the aim here is to find an assignment to all the non-evidence variables that has the highest probability. Formally, we have to find:

$$\arg\max_{\mathbf{z}} \Pr(c(\mathbf{z}, \mathbf{e})) \tag{5}$$

Algorithms used to solve these queries or tasks are called inference algorithms.

## 3 Variable Elimination: An Exact Inference Algorithm

Before, we derive the variable elimination, we will briefly describe three inference operators, summation, multiplication and maximization.

### 3.1 Inference Operators

**Function or Factor** Given a set of variables $\mathbf{Y}$, a function or a factor $\phi(\mathbf{Y})$ is a mapping from all possible assignments to $\mathbf{Y}$ to a positive real number. We will refer to $\mathbf{Y}$ as the scope of the function.

**Multiplication or Product** The product of two functions: $\phi_1$ having scope $\mathbf{X}_1$ and $\phi_2$ having scope $\mathbf{X}_2$ is a function $\phi$ having scope $\mathbf{X} = \mathbf{X}_1 \cup \mathbf{X}_2$. The value associated with each assignment $\mathbf{x} \in D(\mathbf{X})$ is given by:

$$\phi(\mathbf{x}) = \phi_1(\pi(\mathbf{x}, \mathbf{X}_1)) \times \phi_2(\pi(\mathbf{x}, \mathbf{X}_2))$$

**Example 3.** The product of two functions $\phi_1(A, B)$ and $\phi(B, C)$ is a function $\phi(A, B, C)$ given below

| A | B | $\phi_1(a,b)$ |
|---|---|---|
| 0 | 0 | 5 |
| 0 | 1 | 7 |
| 1 | 0 | 3 |
| 1 | 1 | 4 |

$\times$

| B | C | $\phi_2(b,c)$ |
|---|---|---|
| 0 | 0 | 2 |
| 0 | 1 | 9 |
| 1 | 0 | 3 |
| 1 | 1 | 1 |

$=$

| A | B | C | $\phi(a,b,c)$ |
|---|---|---|---|
| 0 | 0 | 0 | 5 × 2=10 |
| 0 | 0 | 1 | 5 × 9=45 |
| 0 | 1 | 0 | 7 × 3=21 |
| 0 | 1 | 1 | 7 × 1=7 |
| 1 | 0 | 0 | 3 × 2=6 |
| 1 | 0 | 1 | 3 × 9=27 |
| 1 | 1 | 0 | 4 × 3=12 |
| 1 | 1 | 1 | 4 × 1=4 |

**Sum-out** The sum-out of a variable $X_i$ from a function $\phi(\mathbf{X})$ such that $X_i \in \mathbf{X}$ is a function $\phi'$ defined over the set of variables $\mathbf{X}' = \mathbf{X} \setminus \{X_i\}$. The value of each assignment $\mathbf{x}' \in D(\mathbf{X}')$ is given by:

$$\phi'(\mathbf{x}') = \sum_{x_i} \phi(c(x_i, \mathbf{x}'))$$

**Example 4.** Summing out $A$ from function $\phi(A, B)$ yields a function $\phi'(B)$ given below:

$\sum_a$

| A | B | $\phi(a,b)$ |
|---|---|---|
| 0 | 0 | 5 |
| 0 | 1 | 7 |
| 1 | 0 | 3 |
| 1 | 1 | 4 |

$=$

| B | $\phi'(b)$ |
|---|---|
| 0 | 5+3=8 |
| 1 | 7+4=11 |

**Max-out** The max-out of a variable $X_i$ from a function $\phi(\mathbf{X})$ such that $X_i \in \mathbf{X}$ is a function $\phi'$ defined over the set of variables $\mathbf{X}' = \mathbf{X} \setminus \{X_i\}$. The value of each assignment $\mathbf{x}' \in D(\mathbf{X}')$ is given by:

$$\phi'(\mathbf{x}') = \max_{x_i} \phi(c(x_i, \mathbf{x}'))$$

**Example 5.** Maxing out $A$ from function $\phi(A, B)$ yields a function $\phi'(B)$ given below:

$\max_a$

| A | B | $\phi(a,b)$ |
|---|---|---|
| 0 | 0 | 5 |
| 0 | 1 | 7 |
| 1 | 0 | 3 |
| 1 | 1 | 4 |

$=$

| B | $\phi'(b)$ |
|---|---|
| 0 | max(5,3)=5 |
| 1 | max(7,4)=7 |

### 3.2 Example

We explain the main concepts in variable elimination using an example. Consider the Bayesian network given in Figure 1. Let $E = true$ be evidence. By definition,

$$P(E = true) = \sum_{a,b,c,d,E=true} P(A = a)P(B = b|A = a)P(C = c|B = b) \\ P(D = d|B = b, c = c)P(E = true|A = a, C = c) \quad (6)$$

However, notice that since $E$ is already assigned a value, we can rewrite the CPT $P(E|A, C)$ as:

| A | C | $\phi(a, c)$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | .5 |
| 1 | 0 | .3 |
| 1 | 1 | 1 |

Rewriting Equation (6) using $\phi$, we get

$$P(E = true) = \sum_{a,b,c,d} P(A = a)P(B = b|A = a)P(C = c|B = b) \\ P(D = d|B = b, c = c)\phi(A = a, C = c) \quad (7)$$

Distributing sums over products along the order $(A, B, C, D)$, we get

$$P(E = true) = \sum_a P(A = a) \sum_b P(B = b|A = a) \sum_c P(C = c|B = b)\phi(A = a, C = c) \\ \left( \sum_d P(D = d|B = b, c = c) \right) \quad (8)$$

We obtain Equation (8) from Equation (7) by applying the distributive law. Recall that the distributive law states that:

$$ab_1 + ab_2 = a(b_1 + b_2)$$

Summing out $D$ from $P(D = d|B = b, c = c)$ yields a function $\phi(B, C)$ defined over $B$ and $C$.

$\sum_d$

| B | C | D | $P(D = d\|B = b, C = c)$ |
|---|---|---|---|
| 0 | 0 | 0 | .2 |
| 0 | 0 | 1 | .8 |
| 0 | 1 | 0 | .1 |
| 0 | 1 | 1 | .9 |
| 1 | 0 | 0 | .3 |
| 1 | 0 | 1 | .7 |
| 1 | 1 | 0 | .5 |
| 1 | 1 | 1 | .5 |

=

| B | C | $\phi(b, c)$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Substituting ($\sum_d P(D = d|b = b, c = c)$ with $\phi(B = b, C = c)$, we get

$$P(E = true) = \sum_a P(A = a) \sum_b P(B = b|A = a) \sum_c P(C = c|B = b)\phi(A = a, C = c)\phi(B = b, C = c) \quad (9)$$

Multiplying $P(C = c|B = b)$, $\phi(A = a, C = c)$ $\phi(B = b, C = c)$ yields a function $\phi(A, B, C)$ and we can rewrite Equation 9 as:

$$P(E = true) = \sum_a P(A = a) \sum_b P(B = b|A = a) \sum_c \phi(A = a, B = b, C = c) \quad (10)$$

Summing out $C$ from $\phi(A = a, B = b, C = c)$ yields a function $\phi(A, B)$. Substituting it in Equation (10), we get

$$P(E = true) = \sum_a P(A = a) \sum_b P(B = b|A = a)\phi(A = a, B = b) \quad (11)$$

Summing out $B$ from $P(B = b|A = a)\phi(A = a, B = b)$ yields a function $\phi(A)$. Substituting it in Equation (11), we get

$$P(E = true) = \sum_a P(A = a)\phi(A = a) \quad (12)$$

Summing out $A$ from Equation $P(A = a)\phi(A = a)$ yields a number which equals the probability of $E = true$.

### 3.3 The Algorithm (for the PE task)

**Algorithm 1:** Variable Elimination

**Input**: A Bayesian network and Evidence $\mathbf{E} = \mathbf{e}$
**Output**: Probability of Evidence
**begin**
  **for** *each CPT that mentions one or more evidence variables* **do**
    Instantiate evidence and remove evidence variables;
  // Let $\Phi = \{\phi_1, \ldots, \phi_n\}$ be the set of functions. Note that these functions include the original CPTs as well as evidence instantiated CPTs
  Let $o = (X_1, \ldots, X_n)$ be an ordering of non-evidence variables;
  **for** $i = 1$ *to* $n$ **do**
    Multiply all functions in $\Phi$ that mention $X_i$ yielding a function $\phi$;
    Sum-out $X_i$ from $\phi$ yielding a function $\phi_{new}$;
    Remove all functions that mention $X_i$ from $\phi$;
    Add $\phi_{new}$ to $\Phi$;
  **return** $\prod_{\phi \in \Phi} \phi$;

Next, we generalize the process described in the previous subsection, yielding Algorithm 1. The algorithm takes as input a Bayesian network and evidence and outputs the probability of evidence. First, we instantiate evidence in every CPT that mentions one or more evidence variables. Namely, we remove all tuples that are not consistent with the evidence from the CPT and then project the CPT on the non-evidence variables in its scope. Let $\Phi$ denote the set (or database) of all valid functions (which is the union of the set of evidence instantiated CPTs and the CPTs which do not have any evidence variables). Then, given an ordering $o$ of variables, the algorithm eliminates variables one by one, along $o$. A variable $X_i$ is eliminated by computing a product of all the functions that mention $X_i$ in $\Phi$, and then summing out $X_i$ from this product. This creates a new function, whose scope is the union of the scopes of all functions that mention $X_i$, minus $X_i$. The algorithm then deletes the functions involving $X_i$ from $\Phi$, adds the newly created function to it and continues. After all variables are eliminated, the algorithm returns the product of all functions in $\Phi$.

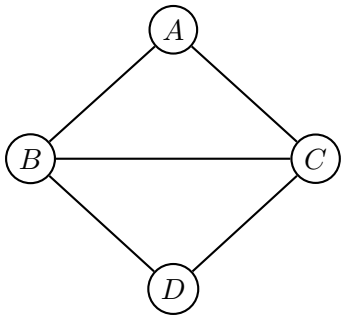
### 3.4 Complexity

The time and space complexity of eliminating a variable $X_i$ is linear in the size of the function obtained by computing the product of all functions that mention $X_i$. Since the size of this function is exponential in the cardinality of the union of the scopes of the functions that mention $X_i$, the time and space complexity of eliminating a variable is exponential in the number of variables that co-occur with $X_i$ in one or more functions. The complexity of variable elimination is the sum of complexity of eliminating variables along the ordering $o$.

It is possible to compute the complexity of variable elimination graphically. More precisely, the complexity can be determined using an undirected graph associated with a Bayesian network, called its primal graph (or moral graph).

**Definition 1** (**Primal Graph).** Given a set of functions $\Phi$ defined over a set of non-evidence variables $\mathbf{X}$, the primal graph is a graph $G(\mathbf{X}, \mathbf{E})$ which has variables of $\mathbf{X}$ as its vertices and an edge between two variables that appear in the scope of a function $\phi \in \Phi$.

**Example 6.** Given evidence $E = e$, the set $\Phi$ associated with the Bayesian network given in Figure 1 equals $\{P(A), P(B|A), P(C|A), P(D|B, C), \phi(A, C)\}$. The primal graph for $\Phi$ is given below:



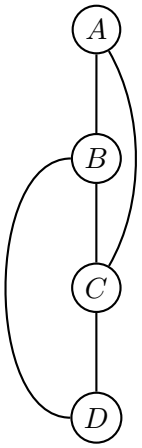

To compute the complexity of variable elimination, we have to convert the primal graph into an induced graph.

**Definition 2** (**Induced Graph).** Given a primal graph $G(\mathbf{X}, \mathbf{E})$ and an ordering $o = (X_1, \ldots, X_n)$ over the vertices of $G$, the induced graph of $G$ along $o$ is the graph $G'(\mathbf{X}, \mathbf{E}')$ (i.e., it has the same vertices as $G$) obtained as follows:

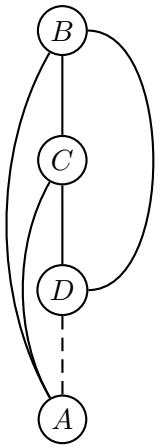
- Initialize $G' = G$
- For $i = 1$ to $n$ do
  - Form a clique between all vertices $X_j$ that are neighbors of $X_i$ but ordered below it in $o$ (i.e., $j > i$)

The **width** of a node in an induced graph is the number of neighbors of $X_i$ that are connected to $X_i$ but ordered below it. The **width** of an induced graph is the maximum width of a node.

**Example 7.** The induced graph along the ordering $(A, B, C, D)$ is given below. It is same as the primal graph.




The induced graph along the ordering $(B, C, D, A)$ is given below. The new edges, which we will call induced edges are drawn using dashed lines.




Intuitively, the scheme used to construct the induced graph schematically follows the variable elimination algorithm. When we eliminate a variable $X_i$, we create a function whose scope equals all nodes that are neighbors of $X_i$ but ordered below it. At each variable $X_i$, the graph over the variables ordered below it is the primal graph w.r.t. the current set of functions $\Phi$.

From the discussion above, it is clear that the complexity of variable elimination along an ordering $o$ is exponential in the width of the induced graph along $o$. Formally,

**Theorem 1.** *The time and space complexity of variable elimination along an ordering $o$ is $O(n \exp(w))$ where $n$ is the number of variables and $w$ is the width of the induced graph along $o$.*

Thus, in order to reduce the complexity of variable elimination, we should try to find an ordering that will yield an induced graph having minimum width. The minimum width is called the **treewidth**. In other words, given a primal graph $G$, the best we can do in terms of complexity is $O(n \exp(w^*))$ where $w^*$ is the treewidth of $G$.

Unfortunately, the problem of finding the treewidth of a graph is NP-hard and therefore in practice we have to resort to various heuristic alternatives. Some notable heuristics are the min-degree heuristic and the min-fill heuristic.

**The min-fill heuristic**

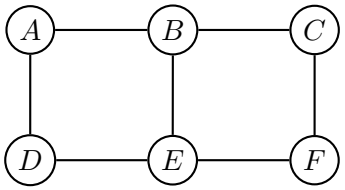
- G=primal graph
- For $i = 1$ to $n$ do
    1. Select a variable $X$ which when eliminated will add the least number of edges in the induced graph. Ties broken randomly. Place $X$ at position $i$ in the ordering.
    2. Update $G$ by creating a clique out of nodes that are neighbors of $X$ and are not yet ordered.

**The min-degree heuristic**

- G=primal graph
- For $i = 1$ to $n$ do
    1. Select a variable $X$ with the smallest degree. Ties broken randomly. Place $X$ at position $i$ in the ordering.
    2. Update $G$ by creating a clique out of nodes that are neighbors of $X$ and are not yet ordered.

Various empirical studies have shown that the min-fill heuristic yields orderings having smaller width than the min-degree heuristic.

**Exercise 3.** Draw the induced graph for the primal graph given below along the ordering $(A, B, C, D, E, F)$. What is the complexity of performing variable elimination along this ordering.



### 3.5 Variable Elimination for MAR and MPE tasks

Variable elimination can be used to compute the conditional marginal probabilities in a straightforward manner. Since, the conditional marginal probability is a ratio of two probabilities, all we have to do is run variable elimination twice, one to compute the numerator and second to compute the denominator. To compute the marginal distribution over a set of variables $\mathbf{Y}$, all we have to do is eliminate all non-evidence variables except $\mathbf{Y}$. You can verify that at termination, the product $\prod_{\phi \in \Phi} \phi$ yields the marginal distribution $P(\mathbf{Y}|\mathbf{e})$.

To compute the MPE value using variable elimination, all we have to do is replace the sum-out operation by the max-out operation. At termination, the algorithm will output the MPE value. To compute the MPE assignment, we have to walk back along the reverse order of elimination and find the value for each variable that yielded the MPE value.

**Remarks.** Since the complexity of variable elimination is exponential in the treewidth of the primal graph associated with the Bayesian network, it is infeasible when the treewidth is large. Unfortunately, in practice, large treewidth Bayesian networks are a norm rather than an exception. Therefore, in practice, we have to rely on approximate inference algorithms. There are two classes of approximate inference algorithms: Belief propagation based and sampling or simulation based. Belief propagation algorithms are loosely based on variable elimination; they iteratively apply the elimination operation in an approximate manner.

## References


[1] A. Darwiche, Modeling and Reasoning with Bayesian Networks, Cambridge university press, 2009.

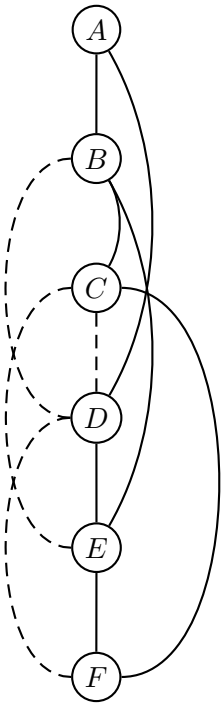
[2] D. Koller and N. Friedman, Probabilistic Graphical Models Principles and Techniques, MIT press, 2009.


## 4 Solutions

**1** The probabilities are $.4 \times .9 \times .7 \times .3 \times .7 = 0.05292$ and $.6 \times 0 \times .8 \times .5 \times .5 = 0$ respectively.

**2** The number of memory units required to represent the CPTs associated with $A, B, C, D, E$ is $1, 2, 2, 4, 4$. The number of memory units required to represent the Bayesian network is $1 + 2 + 2 + 4 + 4 = 13$.

**3** The induced graph along the ordering $(A, B, C, D, E, F)$ is given below.




The width of the nodes $A$, $B$, $C$, $D$, $E$ and $F$ are 2, 3, 3, 2, 1 and 0 respectively. The width of the ordering is the maximum width among the nodes, which is 3. Therefore, the complexity of running variable elimination along this ordering is $O(6\exp(3))$.