COMP9418: Advanced Topics in Statistical Machine Learning

Variable Elimination

Instructor: Gustavo Batista

University of New South Wales

Introduction

- This lecture introduces ones of the simplest methods for inference
  - It is based on the principle of variable elimination
  - We successively remove variables from the Bayesian network, maintaining its ability to answer queries of interest

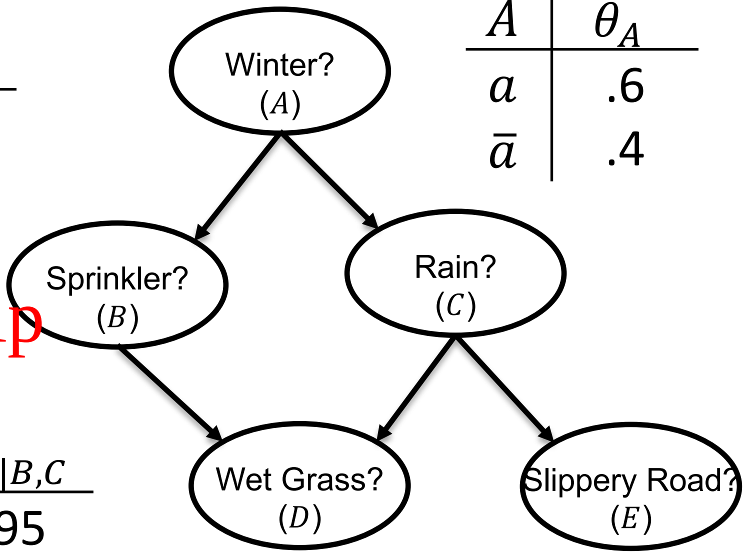
- In previous lectures we identified four types of queries
  - Probability of evidence, prior and posterior marginals, MPE and MAP
  - Variable elimination can be used to answer all these types of queries
  - But we will leave MPE and MAP to a future lecture on this topic
- We will discuss the algorithm of variable elimination
  - Its complexity and how to make it more efficient
  - How to implement it in the tutorials
  - Its variants such as bucket elimination

Process of Elimination

- Given this Bayesian network
  - We are interested in computing the marginal $P(D,E)$

D	E	$P(D,E)$
d	e	.30443
d	$\bar{e}$	.39507
$\bar{d}$	e	.05957
$\bar{d}$	$\bar{e}$	.24093

- The variable elimination (VE) algorithm,
  - Sums out variable A, B and C to construct a marginal over D and E



A	θ_A
a	.6
$\bar{a}$	.4

A	B	$\theta_{B\mid A}$
a	b	.2
a	$\bar{b}$	.8
$\bar{a}$	b	.75
$\bar{a}$	$\bar{b}$	.25

A	C	$\theta_{C\mid A}$
a	c	.8
a	$\bar{c}$	.2
$\bar{a}$	c	.1
$\bar{a}$	$\bar{c}$	.9

B	C	D	$\theta_{D\mid B,C}$
b	c	d	.95
b	c	$\bar{d}$	.05
b	$\bar{c}$	d	.9
b	$\bar{c}$	$\bar{d}$	.1
$\bar{b}$	c	d	.8
$\bar{b}$	c	$\bar{d}$	.2
$\bar{b}$	$\bar{c}$	d	0
$\bar{b}$	$\bar{c}$	$\bar{d}$	1

C	E	$\theta_{E\mid C}$
c	e	.7
c	$\bar{e}$	.3
$\bar{c}$	e	0
$\bar{c}$	$\bar{e}$	1

Process of Elimination

- Consider the joint distribution over all variables. To sum out variable A
  - Merge all rows that agree on values of B, C, D, and E

A	B	C	D	E	$P(.)$
a	b	c	d	e	$.06384$
$\bar{a}$	b	c	d	e	$.01995$

  - Into a single row

B	C	D	E	$P(.)$
b	c	d	e	$.08379 = .06384 + .01995$

  - Resulting in a table with 16 rows that do not mention variable A

A	B	C	D	E	$P(.)$
a	b	c	d	e	$.06384$
a	b	c	d	$\bar{e}$	$.02736$
a	b	c	$\bar{d}$	e	$.00336$
a	b	c	$\bar{d}$	$\bar{e}$	$.00144$
a	b	$\bar{c}$	d	e	0
a	b	$\bar{c}$	d	$\bar{e}$	$.02160$
a	b	$\bar{c}$	$\bar{d}$	e	0
a	b	$\bar{c}$	$\bar{d}$	$\bar{e}$	$.00240$
a	$\bar{b}$	c	d	e	$.21504$
a	$\bar{b}$	c	d	$\bar{e}$	$.09216$
a	$\bar{b}$	c	$\bar{d}$	e	$.05376$
a	$\bar{b}$	c	$\bar{d}$	$\bar{e}$	$.02304$
a	$\bar{b}$	$\bar{c}$	d	e	0
a	$\bar{b}$	$\bar{c}$	d	$\bar{e}$	0
a	$\bar{b}$	$\bar{c}$	$\bar{d}$	e	0
a	$\bar{b}$	$\bar{c}$	$\bar{d}$	$\bar{e}$	$.09600$

A	B	C	D	E	$P(.)$
$\bar{a}$	b	c	d	e	$.01995$
$\bar{a}$	b	c	d	$\bar{e}$	$.00855$
$\bar{a}$	b	c	$\bar{d}$	e	$.00105$
$\bar{a}$	b	c	$\bar{d}$	$\bar{e}$	$.00045$
$\bar{a}$	b	$\bar{c}$	d	e	0
$\bar{a}$	b	$\bar{c}$	d	$\bar{e}$	$.24300$
$\bar{a}$	b	$\bar{c}$	$\bar{d}$	e	0
$\bar{a}$	b	$\bar{c}$	$\bar{d}$	$\bar{e}$	$.02700$
$\bar{a}$	$\bar{b}$	c	d	e	$.00560$
$\bar{a}$	$\bar{b}$	c	d	$\bar{e}$	$.00240$
$\bar{a}$	$\bar{b}$	c	$\bar{d}$	e	$.00140$
$\bar{a}$	$\bar{b}$	c	$\bar{d}$	$\bar{e}$	$.00060$
$\bar{a}$	$\bar{b}$	$\bar{c}$	d	e	0
$\bar{a}$	$\bar{b}$	$\bar{c}$	d	$\bar{e}$	0
$\bar{a}$	$\bar{b}$	$\bar{c}$	$\bar{d}$	e	0
$\bar{a}$	$\bar{b}$	$\bar{c}$	$\bar{d}$	$\bar{e}$	$.09000$

Process of Elimination

- An important property of summing out variables
  - The new distribution is as good as the original one
  - As far as answering queries that do not mention A
  - That is $P'(\alpha) = P(\alpha)$ for any event α that does not involve A
- Therefore, if we want to compute a marginal distribution, say, over D and E
  - We just sum out variables A, B and C from the joint distribution
  - However, this procedure is exponential in the number of variables
- The key insight of VE is that we can sum out variables without constructing the joint probability
  - This allows to sometimes escape the exponential complexity

Factors

- A factor is a function over a set of variables
  - It maps each instantiation of these variables to a non-negative number
  - In some cases the number represents a probability. It may represent a distribution (e.g., f_2) or a conditional distribution (e.g., f_1)
- A factor over an empty set of variables is called trivial
  - It assigns a single number to the trivial instantiation $\top$
- There are two main operations over factors
  - Summing out a variable
  - Multiplying two factors
- These operations are building blocks of many inference algorithms

B	C	D	f_1
b	c	d	.95
b	c	$\bar{d}$	.05
b	$\bar{c}$	d	.9
b	$\bar{c}$	$\bar{d}$	.1
$\bar{b}$	c	d	.8
$\bar{b}$	c	$\bar{d}$	.2
$\bar{b}$	$\bar{c}$	d	0
$\bar{b}$	$\bar{c}$	$\bar{d}$	1

D	E	f_2
d	e	.448
d	$\bar{e}$	.192
$\bar{d}$	e	.112
$\bar{d}$	$\bar{e}$	.248

Summing Out

- Let f be a factor over variables $\boldsymbol{X}$ and let X be a variable in $\boldsymbol{X}$.
  - The result of summing out variable X from the factor f is another factor over variables $\boldsymbol{Y} = \boldsymbol{X} \setminus \{X\}$ which we denote by $\sum_X f$

$$\left(\sum_X f\right)(\boldsymbol{y}) \stackrel{\text{def}}{=} \sum_x f(x, \boldsymbol{y})$$

- To illustrate this process consider the factor f_1
  - Summing out variable D results in a new factor $\sum_D f_1$
  - If we sum out all variables, we get a trivial factor

B	C	D	f_1
b	c	d	$.95$
b	c	$\bar{d}$	$.05$
b	$\bar{c}$	d	$.9$
b	$\bar{c}$	$\bar{d}$	$.1$
$\bar{b}$	c	d	$.8$
$\bar{b}$	c	$\bar{d}$	$.2$
$\bar{b}$	$\bar{c}$	d	0
$\bar{b}$	$\bar{c}$	$\bar{d}$	1

B	C	$\sum_D f_1$
b	c	1
b	$\bar{c}$	1
$\bar{b}$	c	1
$\bar{b}$	$\bar{c}$	1

	$\sum_B \sum_C \sum_D f_1$
⊤	4

Summing Out

- The summing-out operation is commutative
  - Therefore, we can sum out multiple variables without fixing an order
  - This justifies the notation $\sum_{\boldsymbol{X}} f$, where $\boldsymbol{X}$ is a set

$$\sum_{Y}\sum_{X} f = \sum_{X}\sum_{Y} f$$

- This algorithm provides the pseudocode for summing out any number of variables
  - It is $O(\exp(w))$ time and space
  - w is the number of factor variables
- This operation is also known as *marginalization*
  - $\sum_{\boldsymbol{X}} f$ is also known as *projecting factor* f on variables $\boldsymbol{Y}$

Input: $f(\boldsymbol{X})$ and $\boldsymbol{Z}$
Output: $\sum_{\boldsymbol{Z}} f$
$\boldsymbol{Y} \leftarrow \boldsymbol{X} - \boldsymbol{Z}$
$f' \leftarrow$ a factor over variables $\boldsymbol{Y}$ where $f'(\boldsymbol{y}) = 0$ for all $\boldsymbol{y}$
for each instantiation $\boldsymbol{y}$ **do**
    for each instantiation $\boldsymbol{z}$ **do**
        $f'(\boldsymbol{y}) \leftarrow f'(\boldsymbol{y}) + f(\boldsymbol{yz})$
return f'

Multiplication

- The second operation over factors is *multiplication*
  - If we multiply two factors, we construct a new factor over the union of their variables
  - Each instantiation on the new factor is compatible with exactly one instantiation on each original factor
- The result of multiplying two factors $f_1(\boldsymbol{X})$ and $f_2(\boldsymbol{Y})$ is another factor over variables $\boldsymbol{Z} = \boldsymbol{X} \cup \boldsymbol{Y}$, denoted by $f_1 f_2$
  - $(f_1 f_2)(\boldsymbol{z}) \stackrel{\text{def}}{=} f_1(\boldsymbol{x}) f_2(\boldsymbol{y})$
  - Where $\boldsymbol{x}$ and $\boldsymbol{y}$ are compatible with $\boldsymbol{z}$, that is $\boldsymbol{x} \sim \boldsymbol{z}$ and $\boldsymbol{y} \sim \boldsymbol{z}$

B	C	D	f_1
b	c	d	$.95$
b	c	$\bar{d}$	$.05$
b	$\bar{c}$	d	$.9$
b	$\bar{c}$	$\bar{d}$	$.1$
$\bar{b}$	c	d	$.8$
$\bar{b}$	c	$\bar{d}$	$.2$
$\bar{b}$	$\bar{c}$	d	0
$\bar{b}$	$\bar{c}$	$\bar{d}$	1

D	E	f_2
d	e	$.448$
d	$\bar{e}$	$.192$
$\bar{d}$	e	$.112$
$\bar{d}$	$\bar{e}$	$.248$

B	C	D	E	$f_1(B,C,D)f_2(D,E)$
b	c	d	e	$.4256 = (.95).(.448)$
b	c	d	$\bar{e}$	$.1824 = (.95).(192)$
b	c	$\bar{d}$	e	$.0056 = (.05)(.112)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\bar{b}$	$\bar{c}$	$\bar{d}$	$\bar{e}$	$.2480 = (1)(.2480)$

Multiplication

- Factor multiplication is commutative and associative
  - We can multiply several factors without specifying the order of the multiplication
- This algorithm provides a pseudocode for multiplying m factors
  - It is $O(m \exp(w))$ time and space
  - w is the number of variables in the resulting factor

Input: $f_1(\boldsymbol{X}_1), \dots, f_m(\boldsymbol{X}_m)$

Output: $\prod_{i=1}^{m} f_i$

$\boldsymbol{Z} \leftarrow \bigcup_{i=1}^{m} \boldsymbol{X}_i$

$f \leftarrow$ a factor over variables $\boldsymbol{Z}$ where $f(\boldsymbol{z}) = 1$ for all $\boldsymbol{z}$

for each instantiation $\boldsymbol{z}$ **do**

  for $i = 1$ to m **do**

    $\boldsymbol{x}_i \leftarrow$ instantiation of variables X_i consistent with $\boldsymbol{z}$

    $f(\boldsymbol{z}) \leftarrow f(\boldsymbol{z}) f_i(\boldsymbol{x}_i)$

return f

Variable Elimination (VE)

- Suppose we want to compute the joint probability distribution for this network
  - We can use the chain rule for Bayesian networks
  - We can multiply the CPTs, viewing each CPT as a factor

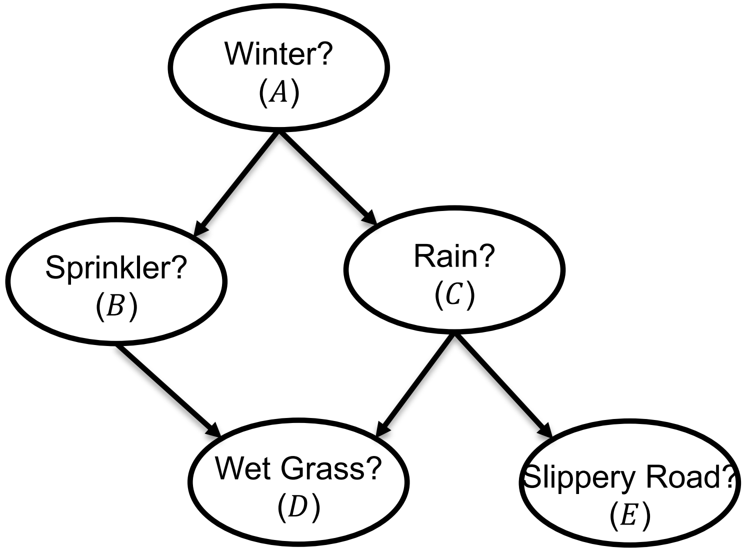
$$P(a,b,c,d,e) = \theta_{e|c}\theta_{d|bc}\theta_{c|a}\theta_{b|a}\theta_a$$

$$\Theta_{E|C}\Theta_{D|BC}\Theta_{C|A}\Theta_{B|A}\Theta_A$$

- Suppose we want to compute the marginals for variables D and E
  - We need to sum out variables A, B and C

$$P(D,E) = \sum_{A,B,C} \Theta_{E|C}\Theta_{D|BC}\Theta_{C|A}\Theta_{B|A}\Theta_A$$

  - This is a combination of marginalization and multiplication
  - However, it still has the problem of complexity



Variable Elimination (VE)

- If f_1 and f_2 are factors and if variable X appears only in f_2, then
  - If $f_1, \dots, f_n$ are the CPTs of a Bayesian network and if we want to sum out variable X
  - We may not need to multiply all these factors first

$$\sum_X f_1 f_2 = f_1 \sum_X f_2$$

- For instance,
  - If variable X appears only in factor f_n
  - But, if variable X appears in two factors f_{n-1} and f_n

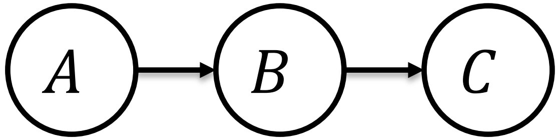
$$\sum_X f_1 \dots f_n = f_1 \dots f_{n-1} \sum_X f_n$$

$$\sum_X f_1 \dots f_n = f_1 \dots f_{n-2} \sum_X f_{n-1} f_n$$

- In general, we need to multiply all factors f_k that include X and then sum out X from $\prod_k f_k$

Variable Elimination (VE)

- Consider this network and assume the goal is to compute $P(C)$
  - We will first eliminate A and then B
  - There are two factors involving A: Θ_A and $\Theta_{B|A}$



A	Θ_A
a	.6
$\bar{a}$	.4

A	B	$\Theta_{B\|A}$
a	b	.9
a	$\bar{b}$	.1
$\bar{a}$	b	.2
$\bar{a}$	$\bar{b}$	.8

B	C	$\Theta_{C\|B}$
b	c	.3
b	$\bar{c}$	.7
$\bar{b}$	c	.5
$\bar{b}$	$\bar{c}$	.5

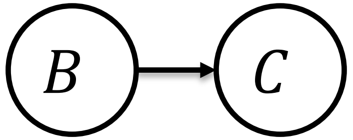
A	B	$\Theta_A\Theta_{B\|A}$
a	b	.54
a	$\bar{b}$	.06
$\bar{a}$	b	.08
$\bar{a}$	$\bar{b}$	.32

  - Summing out variable A, we get

B	$\sum_A \Theta_A\Theta_{B\|A}$
b	.62=.54+.08
$\bar{b}$	.38=.06+.32

Variable Elimination (VE)

- Now, we have two factors, and we want to eliminate variable B



B	C	$\Theta_{C\|B} \sum_A \Theta_A \Theta_{B\|A}$
b	c	.186
b	$\bar{c}$	.434
$\bar{b}$	c	.190
$\bar{b}$	$\bar{c}$	.190

B	$\sum_A \Theta_A \Theta_{B\|A}$
b	.62
$\bar{b}$	.38

- Summing out B

C	$\sum_B \Theta_{C\|B} \sum_A \Theta_A \Theta_{B\|A}$
c	.376
$\bar{c}$	.624

B	C	$\Theta_{C\|B}$
b	c	.3
b	$\bar{c}$	.7
$\bar{b}$	c	.5
$\bar{b}$	$\bar{c}$	.5

Computing Prior Marginals (VE_PR1)

- This algorithm provides the pseudocode for computing the marginal over some variables $\boldsymbol{Q}$
  - How much work does this algorithm do?
  - Note that f and f_i differs only one variable $\pi(i)$

Input: Bayesian network N, query variables $\boldsymbol{Q}$, variable ordering π

Output: prior marginal $P(\boldsymbol{Q})$

$\boldsymbol{S} \leftarrow$ CPTs of network N

for $i = 1$ to length of order π **do**

$\quad f \leftarrow \prod_k f_k$ where f_k belongs to $\boldsymbol{S}$ and mentions variable $\pi(i)$

$\quad f_i \leftarrow \sum_{\pi(i)} f$

$\quad$ replace all factors f_k in $\boldsymbol{S}$ by factor f_i

return $\prod_{f \in \boldsymbol{S}} f$

- For $\pi = \{A, B\}$

$$\underbrace{\sum_B \Theta_{C|B} \overbrace{\sum_A \Theta_A \Theta_{B|A}}^{1}}_{1}$$

- For $\pi = \{B, A\}$

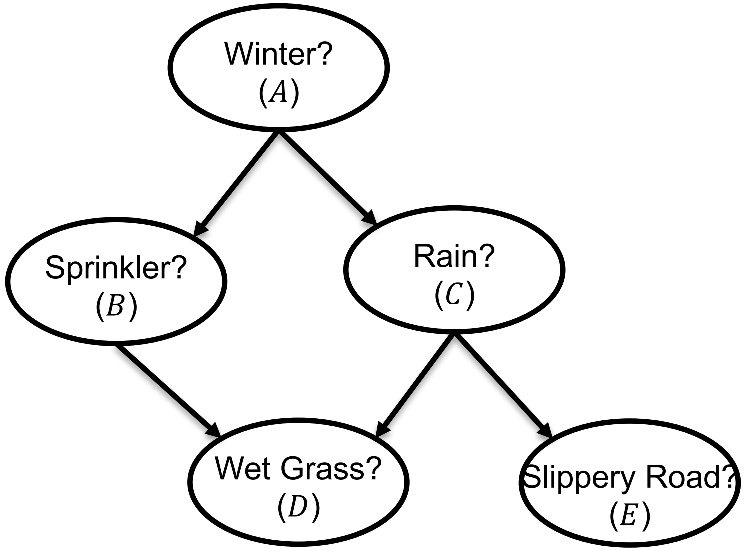
$$\underbrace{\sum_A \Theta_A \overbrace{\sum_B \Theta_{B|A} \Theta_{C|B}}^{2}}_{1}$$

Computing Prior Marginals

- Therefore, although any order will work
  - Some orders are better than others
  - Since they lead to constructing smaller intermediate factors
- We need to find the best order
  - We will address this problem shortly
  - For now, let us try to formalize how to measure the quality of an elimination order
- If the largest factor has w variables, then the complexity of the lines 3-5 is $O(n \exp(w))$
  - This number is known as the *width* of the elimination order
  - We want to find the elimination order with the smallest width
  - The algorithm complexity is $O(n \exp(w) + n \exp(|\boldsymbol{Q}|))$

Elimination Order

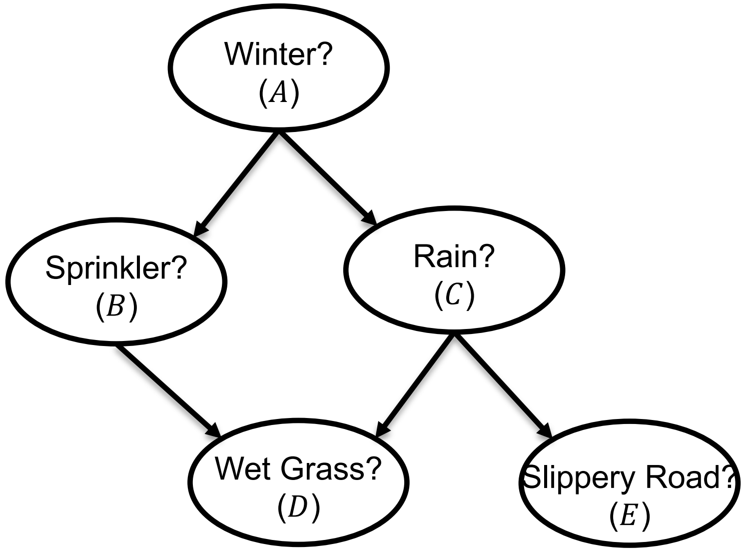
- Suppose we have two elimination orders π_1 and π_2
  - We want to choose the one with smallest width
  - We can modify the VE_PR1 to register the number of variables in line 4
  - The width is maximum number of variables any factor ever contained
- Let us suppose we want com compute $P(C)$
  - With an elimination order B, C, A, D

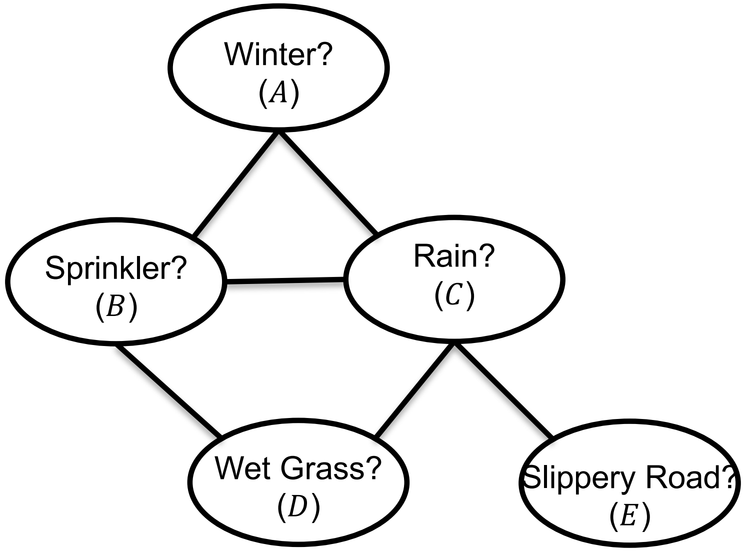


i	$\pi(i)$	S	f_i	w
		$\Theta_A \Theta_{B\|A} \Theta_{C\|A} \Theta_{D\|BC} \Theta_{E\|C}$		
1	B	$\Theta_A \Theta_{C\|A} \Theta_{E\|C} f_1(A, C, D)$	$f_1 = \sum_B \Theta_{B\|A} \Theta_{D\|B,C}$	3
2	C	$\Theta_A f_2(A, D, E)$	$f_2 = \sum_C \Theta_{C\|A} \Theta_{E\|C} f_1(A, C, D)$	3
3	A	$f_3(D, E)$	$f_3 = \sum_A \Theta_A f_2(A, D, E)$	2
4	D	$f_4(E)$	$f_4 = \sum_D f_3(D, E)$	1

Interaction Graph

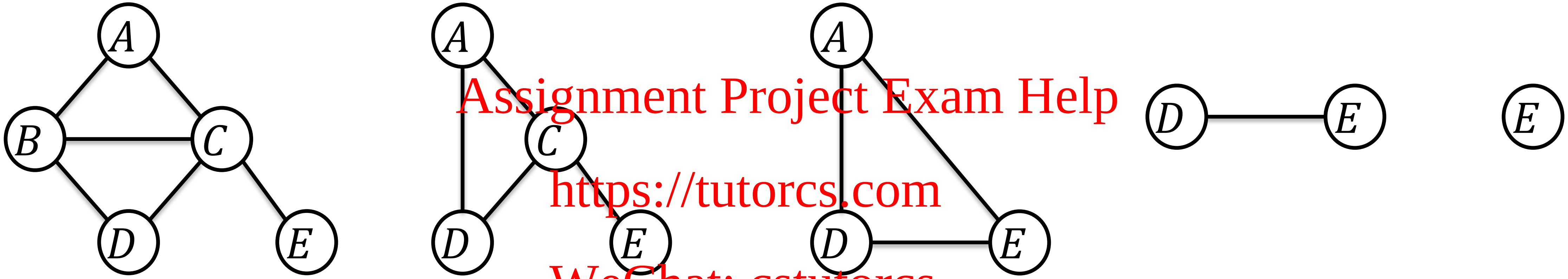
- We can compute the width of an order by simply operating on an undirected graph
  - Let $f_1, \ldots, f_n$ be a set of factors. The *interaction graph* G of these factors is an undirected graph constructed as follows
  - The nodes of G are the variables that appear in factors $f_1, \ldots, f_n$
  - There is an edge between two variables in G iff those variables appear in the same factor
- Another way to visualise the interaction graph is to realise that the variables $\boldsymbol{X}_i$ of f_i form a clique in G
  - For example, $\Theta_A \Theta_{B|A} \Theta_{C|A} \Theta_{D|BC} \Theta_{E|C}$





Interaction Graph

Elimination order: B, C, A, D



$S_1: \Theta_A \Theta_{B|A} \Theta_{C|A} \Theta_{D|BC} \Theta_{E|C}$ $S_2: \Theta_A \Theta_{C|A} \Theta_{E|C} f_1(A, C, D)$ $S_3: \Theta_A f_2(A, D, E)$ $S_4: f_3(D, E)$ $S_5: f_4(E)$

Interaction Graph

- There are two key observations about interaction graphs
  - If G is the interaction graph of factors S, then elimination a variable $\pi(i)$ from S leads to constructing a factor over the neighbours of $\pi(i)$ in G
  - Let S' be the factors that result from eliminating variable $\pi(i)$ from factors S. If G' and G are the interaction graphs of S' and S, respectively, then G' can be obtained from G as follows

a) Add an edge to G between every pair of neighbours of variable $\pi(i)$ that are not already connected

b) Delete variable $\pi(i)$ from G

OrderWidth

Input: Bayesian network N, variable ordering π

Output: the width of π

$G \leftarrow$ interaction graph of the CPTs in network N

$w \leftarrow 0$

for $i = 1$ to length of order π **do**

  $w \leftarrow \max(w, d)$, where d is the number of $\pi(i)$'s neighbours in G

  add an edge between every pair of non-adjacent neighbours of $\pi(i)$ in G

  delete variable $\pi(i)$ from G

return w

- This algorithm provides pseudocode for computing the width of an elimination order
  - One application of OrderWidth is to measure the quality of an ordering before using it
  - However, when the number of orderings is large, we need to do better

Interaction Graph

- Computing the optimal ordering is an NP-hard problem
  - But there are several heuristic approaches that provide good results
  - One of the most popular is also the simplest: *min-degree* heuristic

- The min-degree heuristic eliminates the variable that leads to constructing the smallest factor possible

  - It means we should eliminate the variable with the smallest number of neighbours in the current graph
  - Min-degree is optimal when applied to a network with some elimination order of width ≤ 2

MinDegreeOrder

Input: Bayesian network N with variables $\boldsymbol{X}$

Output: an ordering π of variables $\boldsymbol{X}$

$G \leftarrow$ interaction graph of the CPTs in network N

for $i = 1$ to number of variables in $\boldsymbol{X}$ **do**

    $\pi(i) \leftarrow$ a variable in $\boldsymbol{X}$ with smallest number of neighbours in G

    add an edge between every pair of non-adjacent neighbours of $\pi(i)$ in G

    delete variable $\pi(i)$ from G and from $\boldsymbol{X}$

return π

- There is another popular heuristic that is usually more effective than MinDegreeOrder
  - It consists in eliminating the variable that leads to adding the smallest number of edges in G, called *fill-in edges*
  - This heuristic is called *fill-in heuristic*

MinFillOrder

Input: Bayesian network N with variables $\boldsymbol{X}$

Output: an ordering π of variables $\boldsymbol{X}$

$G \leftarrow$ interaction graph of the CPTs in network N

for $i = 1$ to number of variables in $\boldsymbol{X}$ **do**

    $\pi(i) \leftarrow$ a variable in $\boldsymbol{X}$ that adds the smallest number of edges in G

    add an edge between every pair of non-adjacent neighbours of $\pi(i)$ in G

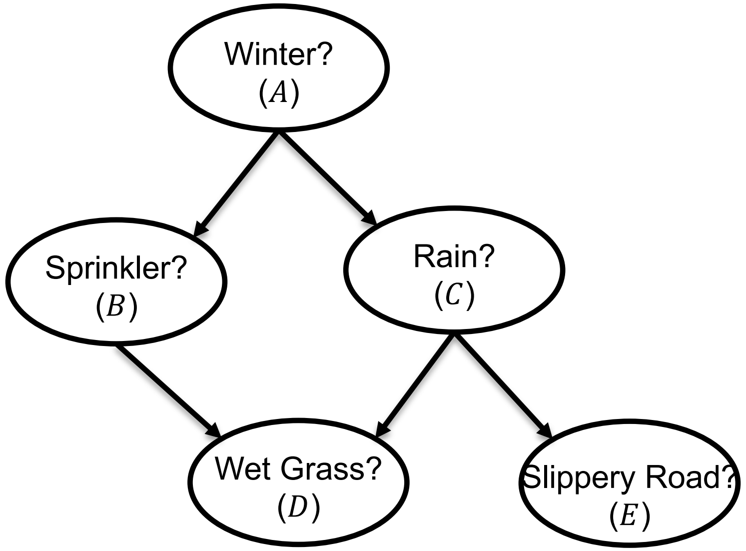
    delete variable $\pi(i)$ from G and from $\boldsymbol{X}$

return π

Computing Posterior Marginals

- We now discuss an algorithm for computing the posterior marginal for a set of variables
  - For instance, $\boldsymbol{Q} = \{D, E\}$ and $\boldsymbol{e}: A = true, B = false$ we get the table on the right side

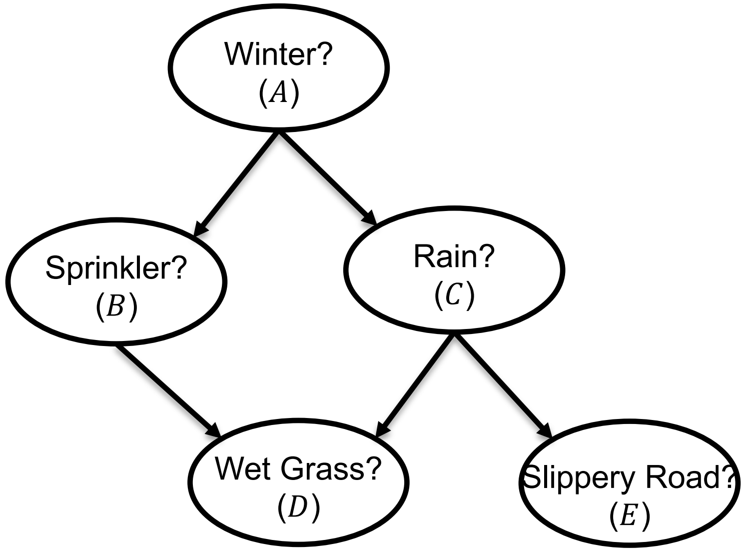
- More generally, given a network N, query $\boldsymbol{Q}$ and evidence $\boldsymbol{e}$
  - We want to compute the posterior marginal $P(\boldsymbol{Q}|\boldsymbol{e})$
  - Prior marginals is a special case of posterior marginals when $\boldsymbol{e}$ is the trivial instantiation



D	E	$P(\boldsymbol{Q}\|\boldsymbol{e})$
d	e	.448
d	$\bar{e}$	.192
$\bar{d}$	e	.112
$\bar{d}$	$\bar{e}$	.248

Computing Posterior Marginals

- It is more useful to construct a variation called *joint marginals*, $P(\boldsymbol{Q}, \boldsymbol{e})$
  - If we take $Q = \{D, E\}$ and $\boldsymbol{e}$: $A = true, B = false$, we get the joint marginal on the right side
  - If we add the probabilities in this factor, we get .48
  - This is the probability of evidence $\boldsymbol{e}$, since $\sum_{\boldsymbol{q}} P(\boldsymbol{q}, \boldsymbol{e}) = P(\boldsymbol{e})$
- This means we can compute $P(\boldsymbol{Q}|\boldsymbol{e})$ by simply normalizing $P(\boldsymbol{Q}, \boldsymbol{e})$
  - We also get the probability of evidence $\boldsymbol{e}$ for free
- VE can be extended to compute joint marginals
  - We need start by zeroing out those rows that are inconsistent with evidence $\boldsymbol{e}$



D	E	$P(\boldsymbol{Q}, \boldsymbol{e})$
d	e	.21504
d	$\bar{e}$	.09216
$\bar{d}$	e	.05376
$\bar{d}$	$\bar{e}$	.11904

Computing Posterior Marginals

- The reduction of factor $f(\boldsymbol{X})$ given evidence $\boldsymbol{e}$ is another factor over variables $\boldsymbol{X}$, denoted by $f^{\boldsymbol{e}}$

$$f^{\boldsymbol{e}}(\boldsymbol{x}) \stackrel{\text{def}}{=} \begin{cases} f(\boldsymbol{x}), & \text{if } \boldsymbol{x} \sim \boldsymbol{e} \\ 0, & \text{otherwise} \end{cases}$$

- For example, given the factor f and evidence $\boldsymbol{e}$: $E = true$, we obtain $f^{\boldsymbol{e}}$

D	E	f
d	e	.448
d	$\bar{e}$	.192
$\bar{d}$	e	.112
$\bar{d}$	$\bar{e}$	.248

D	E	$f^{\boldsymbol{e}}$
d	e	.448
d	$\bar{e}$	0
$\bar{d}$	e	.112
$\bar{d}$	$\bar{e}$	0

D	E	$f^{\boldsymbol{e}}$
d	e	.448
$\bar{d}$	e	.112

Computing Posterior Marginals

- For this network, if $\boldsymbol{Q} = \{D, E\}$ and $\boldsymbol{e}$: $A = true, B = false$. The joint marginal $P(\boldsymbol{Q}, \boldsymbol{e})$ is

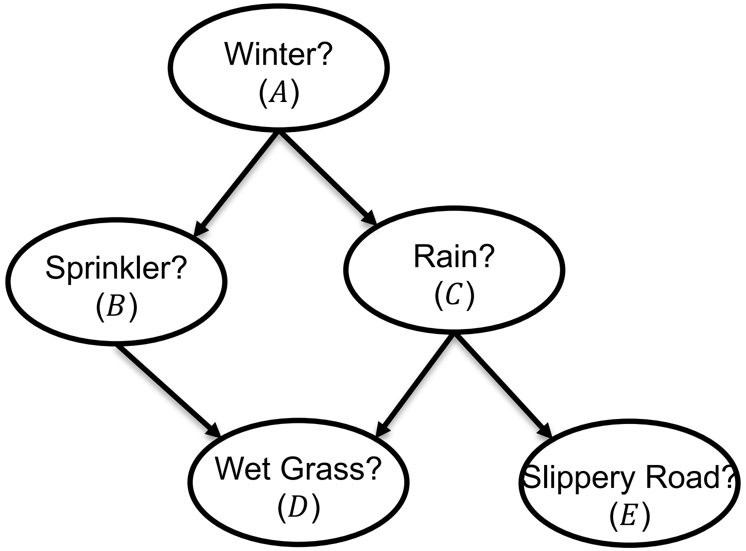
$$P(\boldsymbol{Q}, \boldsymbol{e}) = \sum_{A,B,C} \left(\Theta_A \Theta_{B|A} \Theta_{C|A} \Theta_{D|BC} \Theta_{E|C}\right)^{\boldsymbol{e}}$$

  - We can use the following result

$$(f_1, f_2)^{\boldsymbol{e}} = f_1^{\boldsymbol{e}} f_2^{\boldsymbol{e}}$$

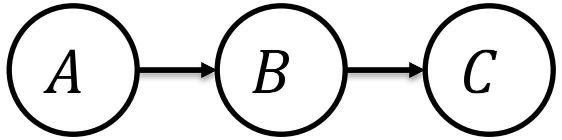
  - Therefore

$$P(\boldsymbol{Q}, \boldsymbol{e}) = \sum_{A,B,C} \Theta_A^{\boldsymbol{e}} \Theta_{B|A}^{\boldsymbol{e}} \Theta_{C|A}^{\boldsymbol{e}} \Theta_{D|BC}^{\boldsymbol{e}} \Theta_{E|C}^{\boldsymbol{e}}$$



Computing Posterior Marginals

- Consider this network. Let $\boldsymbol{Q} = \{C\}$, $\boldsymbol{e}$: $A = true$
  - We want to compute $P(\boldsymbol{Q}, \boldsymbol{e})$, by eliminating A then B
- We first reduce the network CPTs given evidence $\boldsymbol{e}$



A	Θ_A
a	.6
$\bar{a}$	.4

A	B	$\Theta_{B\|A}$
a	b	.9
a	$\bar{b}$	.1
$\bar{a}$	b	.2
$\bar{a}$	$\bar{b}$	.8

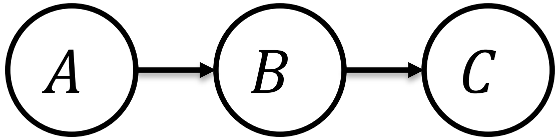
B	C	$\Theta_{C\|B}$
b	c	.3
b	$\bar{c}$	.7
$\bar{b}$	c	.5
$\bar{b}$	$\bar{c}$	.5

Computing Posterior Marginals

- Consider this network. Let $\boldsymbol{Q} = \{C\}$, $\boldsymbol{e}$: $A = true$
  - We want to compute $P(\boldsymbol{Q}, \boldsymbol{e})$, by eliminating A then B
- We first reduce the network CPTs given evidence $\boldsymbol{e}$
  - Then we need to evaluate

$$P(\boldsymbol{Q}, \boldsymbol{e}) = \sum_B \sum_A \Theta_A^{\boldsymbol{e}} \Theta_{B|A}^{\boldsymbol{e}} \Theta_{C|B}^{\boldsymbol{e}}$$

$$= \sum_B \Theta_{C|B}^{\boldsymbol{e}} \sum_A \Theta_A^{\boldsymbol{e}} \Theta_{B|A}^{\boldsymbol{e}}$$



A	$\Theta_A^{\boldsymbol{e}}$
a	.6

A	B	$\Theta_{B\|A}^{\boldsymbol{e}}$
a	b	.9
a	$\bar{b}$	.1

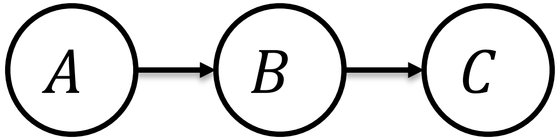
B	C	$\Theta_{C\|B}^{\boldsymbol{e}}$
b	c	.3
b	$\bar{c}$	.7
$\bar{b}$	c	.5
$\bar{b}$	$\bar{c}$	.5

A	B	$\Theta_A^{\boldsymbol{e}} \Theta_{B\|A}^{\boldsymbol{e}}$
a	b	.54
a	$\bar{b}$	.06

B	$\sum_A \Theta_A^{\boldsymbol{e}} \Theta_{B\|A}^{\boldsymbol{e}}$
b	.54
$\bar{b}$	.06

Computing Posterior Marginals

- Consider this network. Let $\boldsymbol{Q} = \{C\}$, $\boldsymbol{e}$: $A = true$
  - We want to compute $P(\boldsymbol{Q}, \boldsymbol{e})$, by eliminating A then B
- We first reduce the network CPTs given evidence $\boldsymbol{e}$
  - Then we need to evaluate



$$P(\boldsymbol{Q}, \boldsymbol{e}) = \sum_B \sum_A \Theta_A^{\boldsymbol{e}} \Theta_{B|A}^{\boldsymbol{e}} \Theta_{C|B}^{\boldsymbol{e}}$$

$$= \sum_B \Theta_{C|B}^{\boldsymbol{e}} \sum_A \Theta_A^{\boldsymbol{e}} \Theta_{B|A}^{\boldsymbol{e}}$$

B	$\sum_A \Theta_A^{\boldsymbol{e}} \Theta_{B|A}^{\boldsymbol{e}}$
b	.54
$\bar{b}$	.06

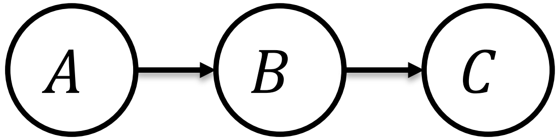
B	C	$\Theta_{C|B}^{\boldsymbol{e}}$
b	c	.3
b	$\bar{c}$	.7
$\bar{b}$	c	.5
$\bar{b}$	$\bar{c}$	.5

B	C	$\Theta_{C|B}^{\boldsymbol{e}} \sum_A \Theta_A^{\boldsymbol{e}} \Theta_{B|A}^{\boldsymbol{e}}$
b	c	.162
b	$\bar{c}$	.378
$\bar{b}$	c	.030
$\bar{b}$	$\bar{c}$	.030

C	$\sum_B \Theta_{C|B}^{\boldsymbol{e}} \sum_A \Theta_A^{\boldsymbol{e}} \Theta_{B|A}^{\boldsymbol{e}}$
c	.192
$\bar{c}$	.408

Computing Posterior Marginals

- Consider this network. Let $\boldsymbol{Q} = \{C\}$, $\boldsymbol{e}$: $A = true$
  - We want to compute $P(\boldsymbol{Q}, \boldsymbol{e})$, by eliminating A then B



- We first reduce the network CPTs given evidence $\boldsymbol{e}$
  - Then we need to evaluate

$$P(\boldsymbol{Q}, \boldsymbol{e}) = \sum_{B} \sum_{A} \Theta_{A}^{\boldsymbol{e}} \Theta_{B|A}^{\boldsymbol{e}} \Theta_{C|B}^{\boldsymbol{e}}$$

$$= \sum_{B} \Theta_{C|B}^{\boldsymbol{e}} \sum_{A} \Theta_{A}^{\boldsymbol{e}} \Theta_{B|A}^{\boldsymbol{e}}$$

C	$\sum_B \Theta_{C\|B}^{\boldsymbol{e}} \sum_A \Theta_{A}^{\boldsymbol{e}} \Theta_{B\|A}^{\boldsymbol{e}}$
c	.192
$\bar{c}$	.408

- To compute $P(C|A = true)$
  - We need to normalize this factor, which gives

C	$P(C\|A = true)$
c	.32
$\bar{c}$	.68

Computing Joint Marginals (VE_PR2)

Input: Bayesian network N, query variables $\boldsymbol{Q}$, variable ordering π, evidence $\boldsymbol{e}$

Output: joint marginal $P(\boldsymbol{Q}, \boldsymbol{e})$

```
S ← {f^e : f is a CPTs of network N}
for i = 1 to length of order π do
    f ← ∏_k f_k where f_k belongs to S and mentions variable π(i)
    f_i ← Σ_π(i) f
    replace all factors f_k in S by factor f_i
return ∏_{f∈S} f
```

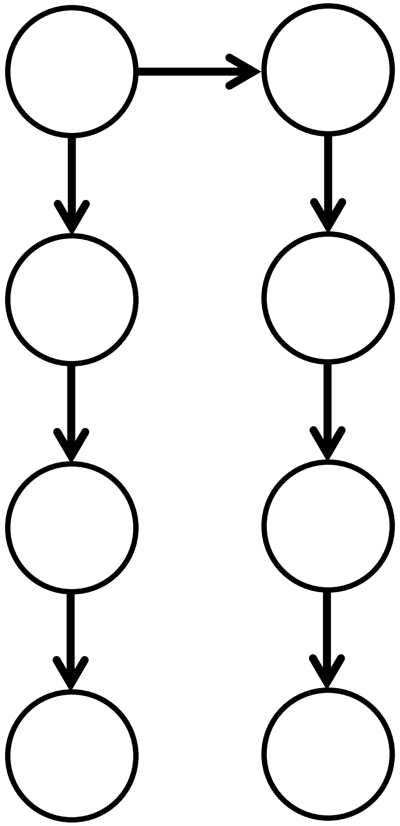

- It is not uncommon to run VE_PR2 with empty $\boldsymbol{Q}$
  - The algorithm will return a trivial factor with the probability of evidence $\boldsymbol{e}$

Network Structure and Complexity

- Suppose we have two Bayesian networks with 100 variables
- We find the best elimination order for both
  - The first one has width 3, and therefore, good performance
  - The second one has width 25, and poor performance regardless of the elimination order
- Why is the second network more difficult given that both have the same number of variables?
  - The answer lies in the notion of *treewidth*
  - It is a number that quantifies the extend a network resembles a tree structure
  - No elimination order can have a width smaller than the network treewidth
  - There is an elimination order whose width equals the network treewidth. Yet determining such an order is NP-hard

Treewidth

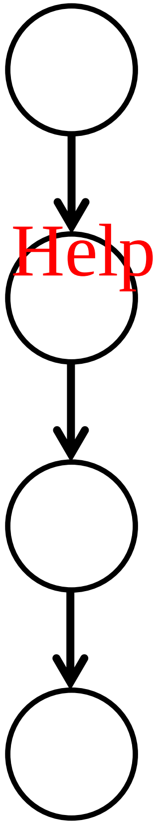
- Intuitively, treewidth is a number that quantifies the extent a network resembles a tree structure
  - The treewidth can be defined as the width of this best *complete elimination order*
  - A complete elimination order has all network variables



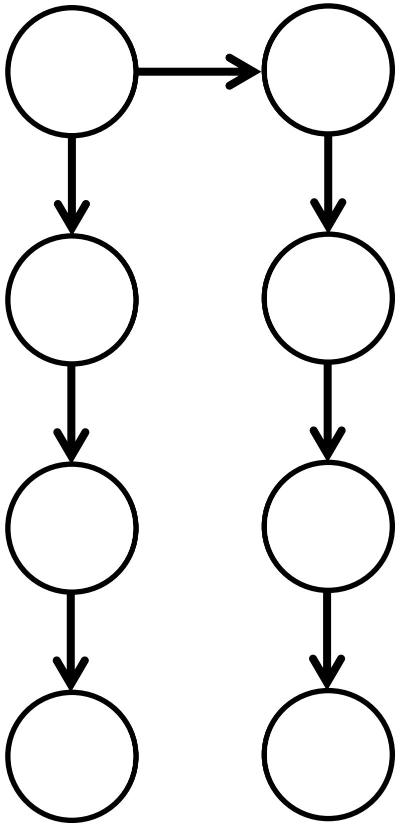
Network with treewidth = 1

Treewidth: Intuition

- The number of nodes has no effect on treewidth
  - For example, these two networks have treewidth = 1
  - But the second one has the twice the number of nodes



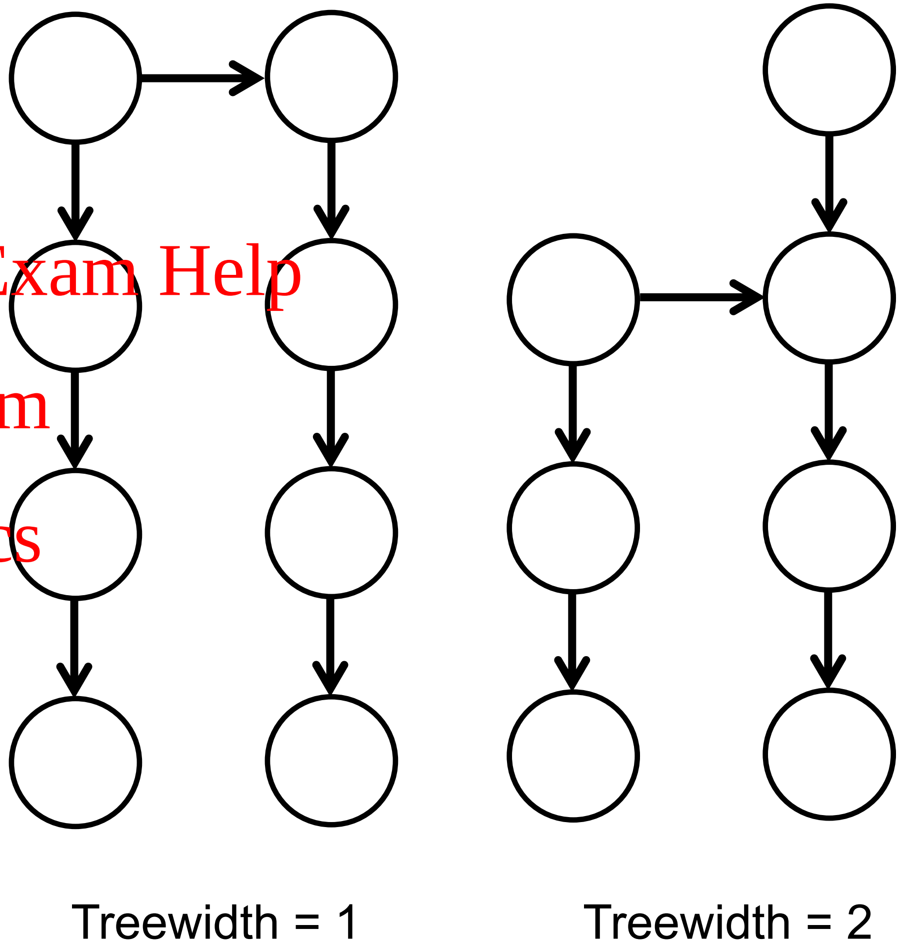
Treewidth = 1



Treewidth = 1

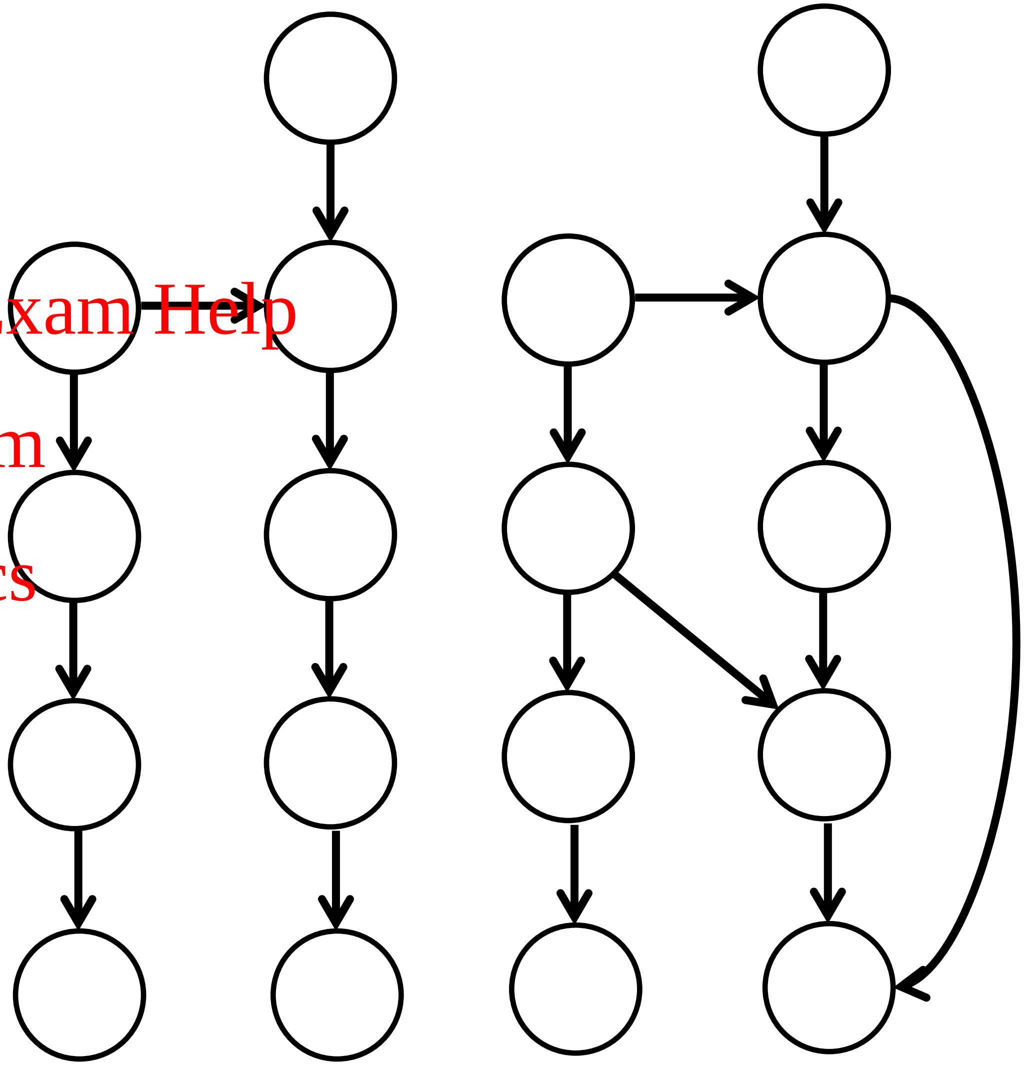
Treewidth: Intuition

- The number of parents per node has a direct effect on treewidth
  - If the number of parents can reach k, the treewidth is no less than k



Treewidth: Intuition

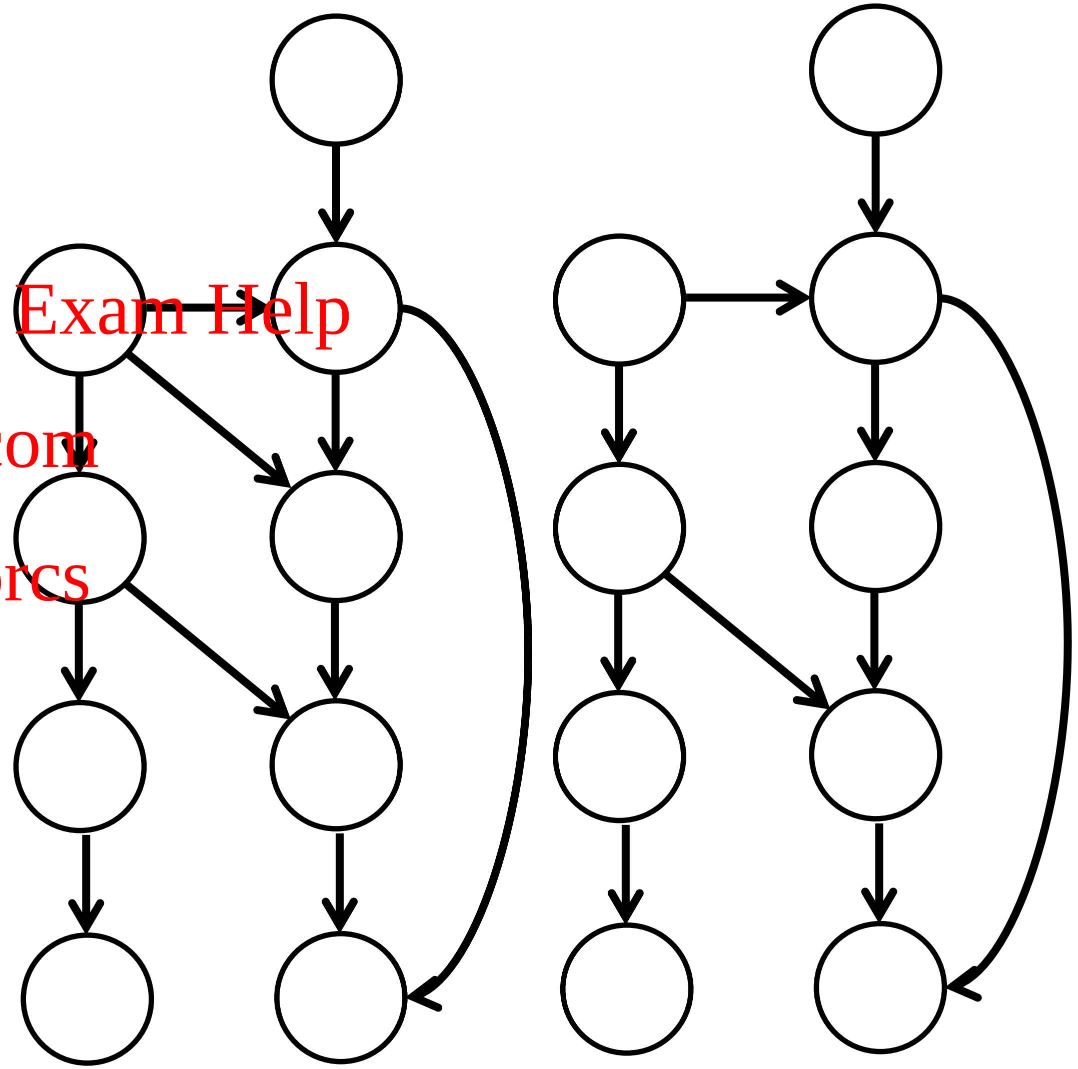
- Loops tend to increase treewidth
  - For example, the second network was obtained from the first by introducing some loops
  - Notice the maximum number of parents per node did not increase



Treewidth = 2 Treewidth = 3

Treewidth: Intuition

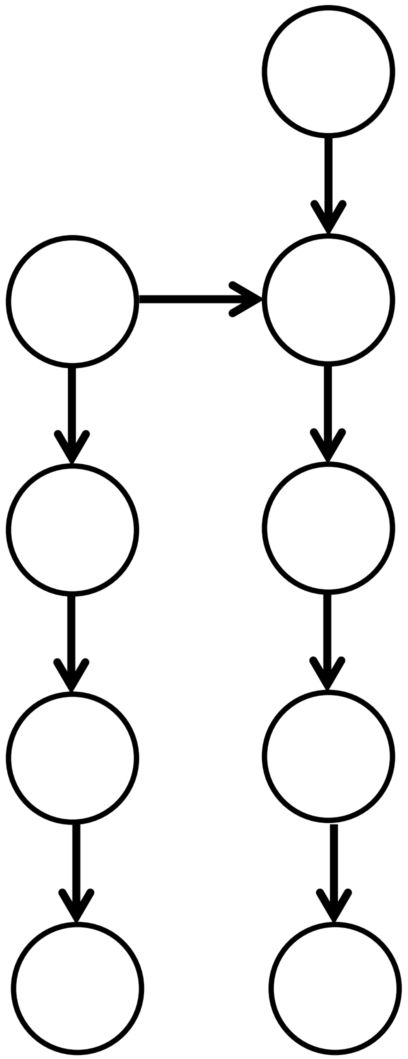
- The number of loops per se does not have a genuine effect on treewidth
  - It is the nature of the loops which does
  - Their interaction in particular
  - These two networks have the same treewidth, although the first has more loops



Treewidth = 3 Treewidth = 3

Classes of Networks with Known Treewidth

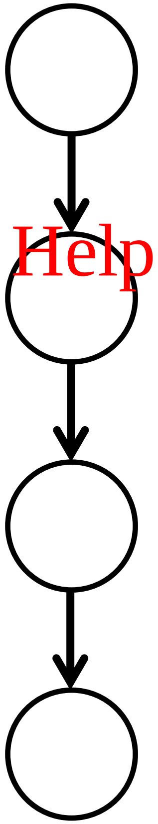
- *Polytrees networks* have at most one (undirected) path between two nodes
  - Also known as *singly connected networks*
  - The treewidth of polytrees is *k*, where *k* is the maximum number of parents of any node.



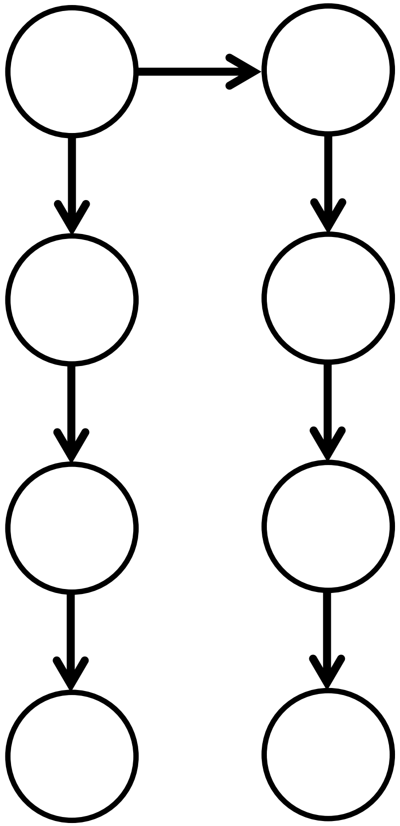
Treewidth = 2

Classes of Networks with Known Treewidth

- *Tree networks* are polytrees where each node has at most one parent
  - Leading to a treewidth of at most 1



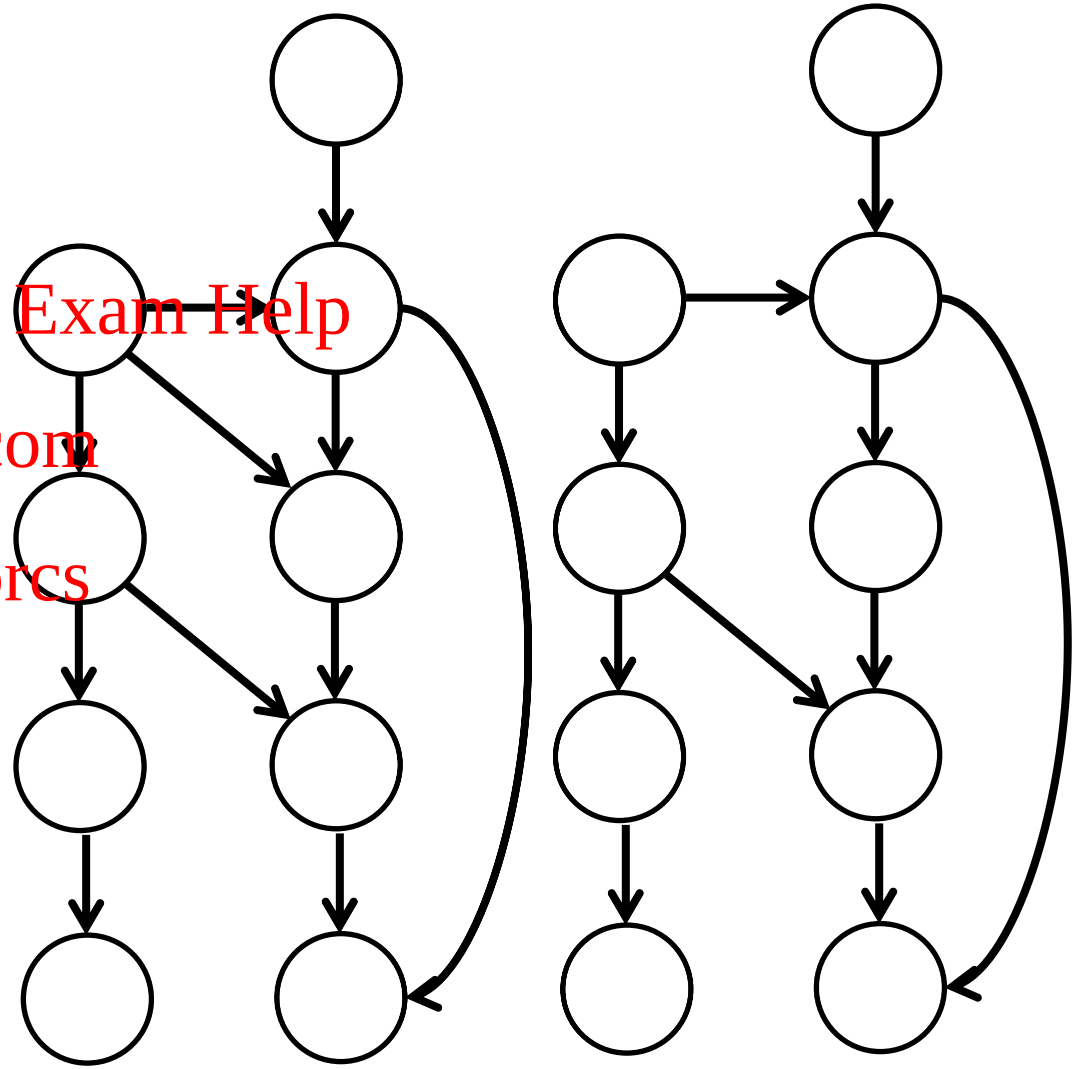
Treewidth = 1



Treewidth = 1

Classes of Networks with Known Treewidth

- *Multiply connected* are networks not singly connected (polytrees)



Treewidth = 3 Treewidth = 3

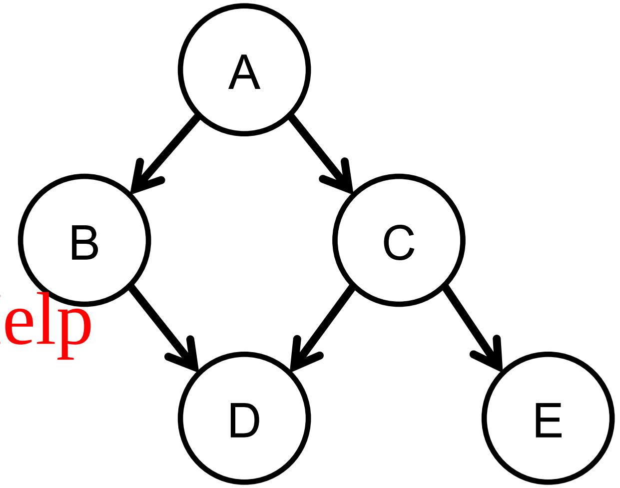
Query Structure

- Treewidth indicates that network structure has a major impact on VE performance
  - Therefore, these algorithms are also known as *structure-based algorithms*
- Network structure can be simplified based on query structure
  - Query is a pair $(\boldsymbol{Q}, \boldsymbol{e})$

- Goal is to compute $P(\boldsymbol{Q}, \boldsymbol{e})$
  - If $\boldsymbol{E} = \emptyset$ then we are interested in $P(\boldsymbol{Q})$
  - If $\boldsymbol{Q} = \emptyset$ then we are interested in $P(\boldsymbol{e})$
- The complexity of inference can be affected by the number and location of variables in $\boldsymbol{Q}$ and $\boldsymbol{e}$
  - For a given query, we provide two transformations that simplify the network, making it more amenable to inference
  - Yet preserving its ability to compute the joint marginal $P(\boldsymbol{Q}, \boldsymbol{e})$

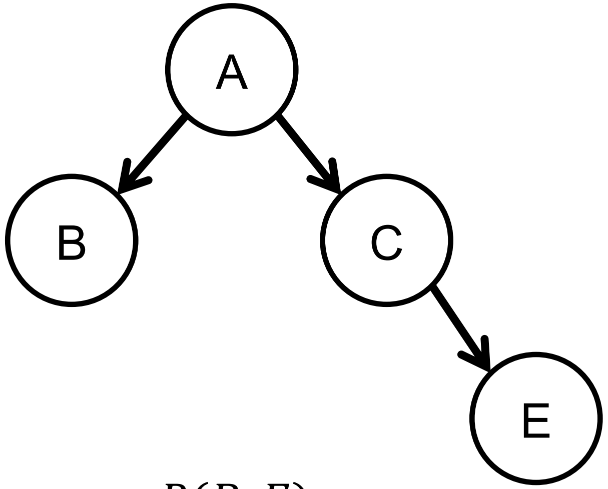
Pruning Nodes

- Given a Bayesian network G and a query $(\boldsymbol{Q}, \boldsymbol{e})$, we can remove any leaf node if it does not belong to $\boldsymbol{Q} \cup \boldsymbol{E}$
  - This operation can be applied iteratively, possibly removing many nodes

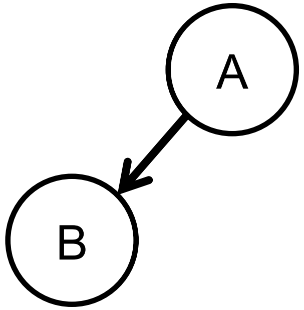


Network structure

- The network structure on the top has treewidth = 2
  - After pruning, both networks have treewidth = 1



$P(B, E)$

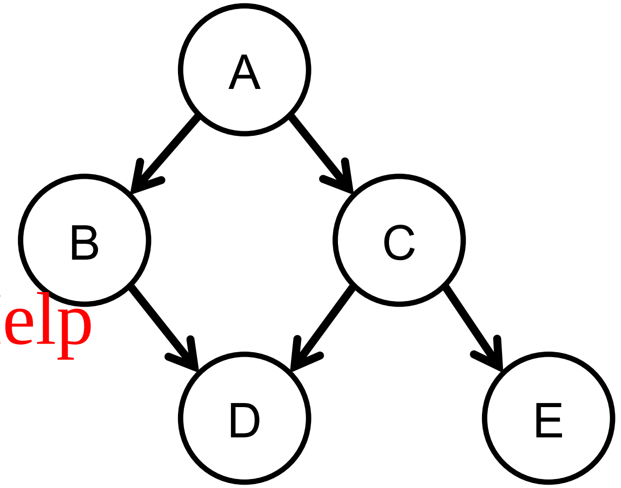


$P(B)$

Pruning Nodes

- Why does pruning leaf nodes work?
  - For any CPT $P(X|\boldsymbol{U})$, the result of summing out variable X is a factor that assigns 1 to each instantiation
  - Multiplying this factor by any other factor f that includes variables $\boldsymbol{U}$ will give factor f back

- The figure gives a numerical example

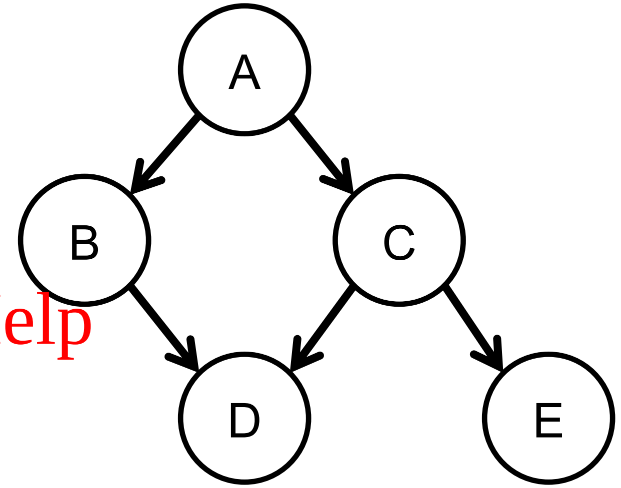


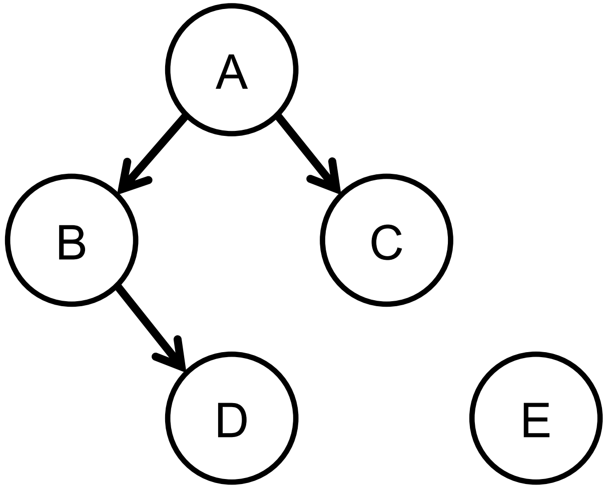
B	C	D	$\theta_{D\|B,C}$
b	c	d	.95
b	c	$\bar{d}$	.05
b	$\bar{c}$	d	.9
b	$\bar{c}$	$\bar{d}$	.1
$\bar{b}$	c	d	.8
$\bar{b}$	c	$\bar{d}$	.2
$\bar{b}$	$\bar{c}$	d	0
$\bar{b}$	$\bar{c}$	$\bar{d}$	1

B	C	$\sum_D \theta_{D\|B,C}$
b	c	1
b	$\bar{c}$	1
$\bar{b}$	c	1
$\bar{b}$	$\bar{c}$	1

Pruning Edges

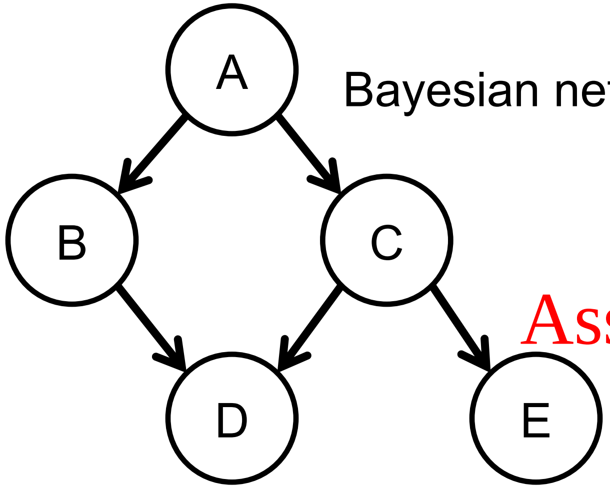
- Given a Bayesian network G and a query $(\boldsymbol{Q}, \boldsymbol{e})$, we can remove each edge $U \rightarrow X$ that originates from a node $U \in \boldsymbol{E}$
  - Remove the edge $U \rightarrow X$
  - Replace the CPT $\Theta_{X|\boldsymbol{U}}$ by a smaller CPT, which is obtained from $\Theta_{X|\boldsymbol{U}}$ by assuming the value $\boldsymbol{u}$ of $\boldsymbol{U}$ given in evidence $\boldsymbol{e}$





Pruned edges for $\boldsymbol{e}: C = false$

Pruning Edges



Bayesian network

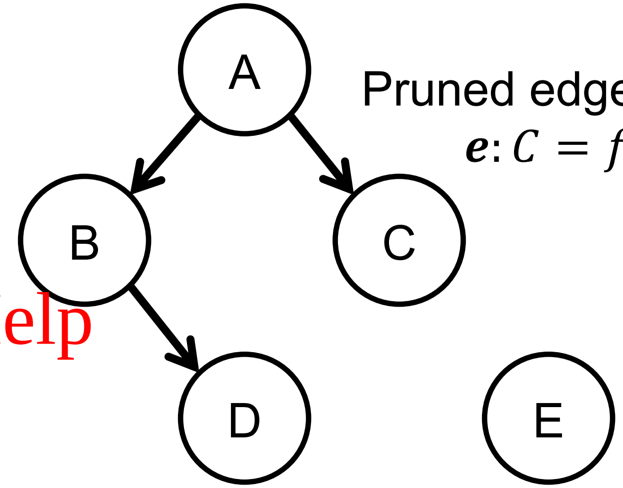
A	θ_A
a	.6
$\bar{a}$	.4

A	B	$\theta_{B\|A}$
a	b	.2
a	$\bar{b}$	.8
$\bar{a}$	b	.75
$\bar{a}$	$\bar{b}$	.25

B	C	D	$\theta_{D\|B,C}$
b	c	d	.95
b	c	$\bar{d}$	.05
b	$\bar{c}$	d	.9
b	$\bar{c}$	$\bar{d}$	.1
$\bar{b}$	c	d	.8
$\bar{b}$	c	$\bar{d}$	.2
$\bar{b}$	$\bar{c}$	d	0
$\bar{b}$	$\bar{c}$	$\bar{d}$	1

A	C	$\theta_{C\|A}$
a	c	.8
a	$\bar{c}$	.2
$\bar{a}$	c	.1
$\bar{a}$	$\bar{c}$	.9

C	E	$\theta_{E\|C}$
c	e	.7
c	$\bar{e}$	.3
$\bar{c}$	e	0
$\bar{c}$	$\bar{e}$	1



Pruned edges for $\boldsymbol{e}: C = false$

A	θ_A
a	.6
$\bar{a}$	.4

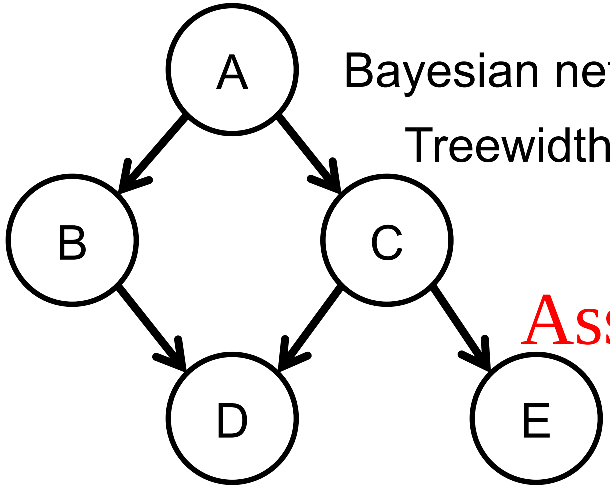
A	B	$\theta_{B\|A}$
a	b	.2
a	$\bar{b}$	.8
$\bar{a}$	b	.75
$\bar{a}$	$\bar{b}$	.25

B	D	$\sum_C \theta^{\bar{c}}_{D\|B,C}$
b	d	.9
b	$\bar{d}$	.1
$\bar{b}$	d	0
$\bar{b}$	$\bar{d}$	1

A	C	$\theta_{C\|A}$
a	c	.8
a	$\bar{c}$	.2
$\bar{a}$	c	.1
$\bar{a}$	$\bar{c}$	.9

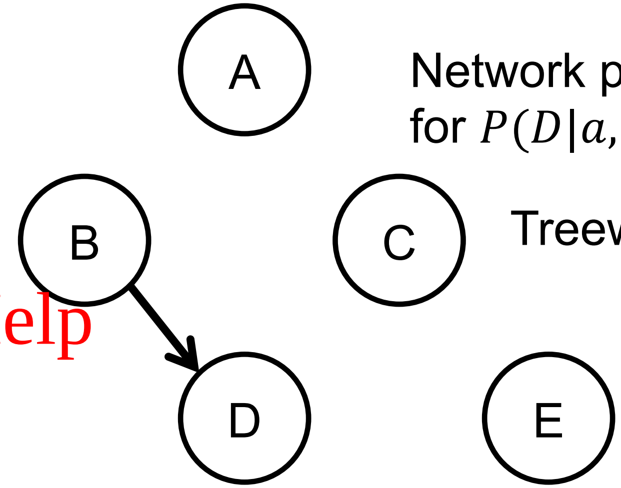
E	$\sum_C \theta^{\bar{c}}_{E\|C}$
e	0
$\bar{e}$	1

Pruning Nodes + Edges = Network Pruning



Bayesian network

Treewidth = 2



Network pruning for $P(D|a,\bar{c})$

Treewidth = 1

A	θ_A
a	.6
$\bar{a}$	.4

A	B	$\theta_{B\|A}$
a	b	.2
a	$\bar{b}$	.8
$\bar{a}$	b	.75
$\bar{a}$	$\bar{b}$	.25

B	C	D	$\theta_{D\|B,C}$
b	c	d	.95
b	c	$\bar{d}$	.05
b	$\bar{c}$	d	.9
b	$\bar{c}$	$\bar{d}$	.1
$\bar{b}$	c	d	.8
$\bar{b}$	c	$\bar{d}$	.2
$\bar{b}$	$\bar{c}$	d	0
$\bar{b}$	$\bar{c}$	$\bar{d}$	1

A	C	$\theta_{C\|A}$
a	c	.8
a	$\bar{c}$	.2
$\bar{a}$	c	.1
$\bar{a}$	$\bar{c}$	.9

C	E	$\theta_{E\|C}$
c	e	.7
c	$\bar{e}$	.3
$\bar{c}$	e	0
$\bar{c}$	$\bar{e}$	1

A	θ_A
a	.6
$\bar{a}$	.4

B	$\sum_A \theta^a_{B\|A}$
b	.2
$\bar{b}$	.8

B	D	$\sum_C \theta^{\bar{c}}_{D\|B,C}$
b	d	.9
b	$\bar{d}$	.1
$\bar{b}$	d	0
$\bar{b}$	$\bar{d}$	1

C	$\sum_A \theta^a_{C\|A}$
c	.8
$\bar{c}$	.2

Network Pruning (VE_PR)

Input: Bayesian network N, query variables $\boldsymbol{Q}$, evidence $\boldsymbol{e}$

Output: joint marginal $P(\boldsymbol{Q}, \boldsymbol{e})$

N' ← pruneNetwork($N, \boldsymbol{Q}, \boldsymbol{e}$)

π ← an ordering of variables not in $\boldsymbol{Q}$ computed relative to N'

$\boldsymbol{S} \leftarrow \{f^{\boldsymbol{e}} : f$ is a CPTs of network $N\}$

for $i = 1$ to length of order π **do**

$\quad f \leftarrow \prod_k f_k$ where f_k belongs to $\boldsymbol{S}$ and mentions variable $\pi(i)$

$\quad f_i \leftarrow \sum_{\pi(i)} f$

$\quad$ replace all factors f_k in $\boldsymbol{S}$ by factor f_i

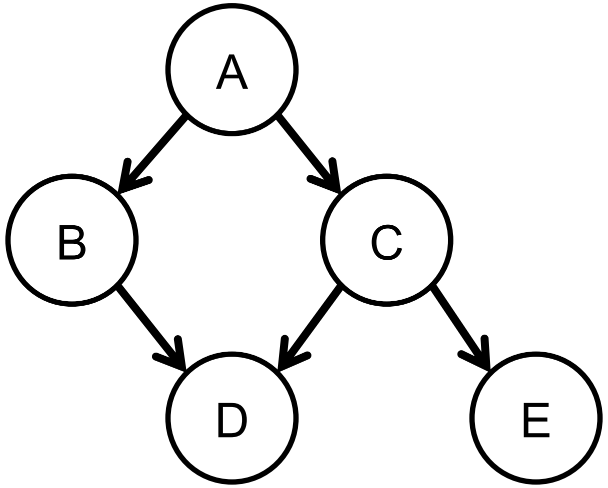
return $\prod_{f \in \boldsymbol{S}} f$

- The effective treewidth for a network N with respect to query $(\boldsymbol{Q}, \boldsymbol{e})$ is the treewidth of pruneNetwork($N, \boldsymbol{Q}, \boldsymbol{e}$)
- Pruning is done in linear time in the size of the network. Therefore it is usually worthwhile to prune the network before answering the query

Bucket Elimination

- VE requires to identify all factors that mention a particular variable
  - Best complexity if factors are found in linear time in the number of factors

- One method is to arrange the factors into *buckets*
  - One bucket for each network variable
- For instance, assume the elimination order: E, B, C, D, A



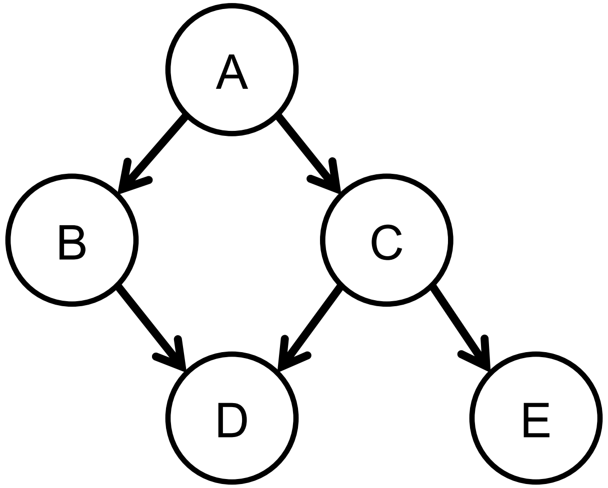
Bucket Label	Bucket Factors
E	$\Theta_{E\|C}$
B	$\Theta_{B\|A}, \Theta_{D\|BC}$
C	$\Theta_{C\|A}$
D	
A	Θ_A

Bucket Elimination

- Each factor is placed in the first bucket whose label appear in the factor
  - $\phi_C(C, A)$ goes to bucket C instead of A

- Given these buckets. We eliminate variables by processing the buckets from top to bottom

- When processing variable $\pi(i)$
  - Multiply all factors in the bucket $\pi(i)$
  - Sum out $\pi(i)$
  - Place the resulting factor f_i in the first next bucket whose label appears in f_i



Bucket Label	Bucket Factors
E	$\Theta_{E\|C}$
B	$\Theta_{B\|A}, \Theta_{D\|BC}$
C	$\Theta_{C\|A}$
D	
A	Θ_A

Bucket Elimination

- For instance, after processing bucket E, the resulting factor is placed in bucket C

- If our goal is to obtain marginals for variables D and A
  - We should process the first three buckets
  - The buckets for D and A will contain a factored representation of the marginal
  - We can multiply these factors or simply keep them in factored form

Bucket Label	**Bucket Factors**
E	$\Theta_{E\|C}$
B	$\Theta_{B\|A}, \Theta_{D\|BC}$
C	$\Theta_{C\|A}$
D	
A	Θ_{A}

Bucket Label	**Bucket Factors**
E	
B	$\Theta_{B\|A}, \Theta_{D\|BC}$
C	$\Theta_{C\|A}, \sum_E \Theta_{E\|C}$
D	
A	Θ_{A}

Bucket Elimination

- The original bucket elimination algorithm handles evidence differently
  - It does not reduce the factors explicitly
  - Instead, it creates new factors
- For instance, given evidence $\boldsymbol{e}$: $B = true, E = false$, it creates the factors

B	λ_B
b	1
$\bar{b}$	0

E	λ_E
e	0
$\bar{e}$	1

- λ_B and λ_E are called *evidence indicators*
  - They are added to their corresponding buckets

Bucket Label	Bucket Factors
E	$\Theta_{E\|C}$
B	$\Theta_{B\|A}, \Theta_{D\|BC}$
C	$\Theta_{C\|A}$
D	
A	Θ_A

Bucket Label	Bucket Factors
E	$\Theta_{E\|C}, \lambda_E$
B	$\Theta_{B\|A}, \Theta_{D\|BC}, \lambda_B$
C	$\Theta_{C\|A}$
D	
A	Θ_A

Conclusion

- Network pruning removes edges and nodes according to a query $(\boldsymbol{Q}, \boldsymbol{e})$
  - Pruning may lead to significant reduction in treewidth
  - *Effective treewidth* is defined as the treewidth after pruning
- Bucket elimination
  - Algorithm that organizes the factors in buckets by variables
  - Quick access to all factors that references a variable