

Inference in Markov Networks

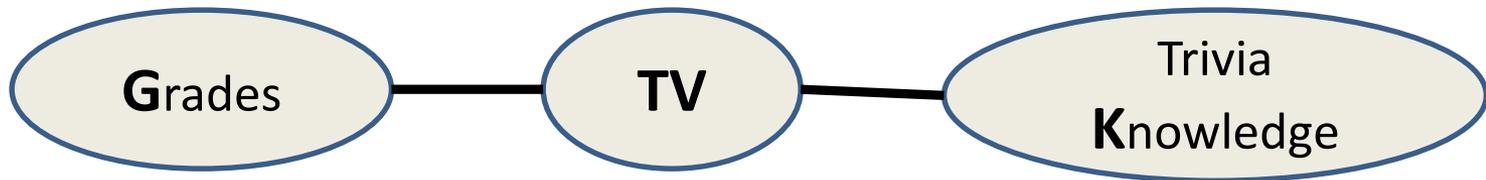
Doug Downey

Northwestern EECS 395/495 Fall 2014

Markov Network Inference

- $$P(\mathbf{x}) = \frac{\prod_c \phi_c(\mathbf{x}_c)}{Z}$$

$$Z = \sum_{\mathbf{x}} \prod_c \phi_c(\mathbf{x}_c)$$

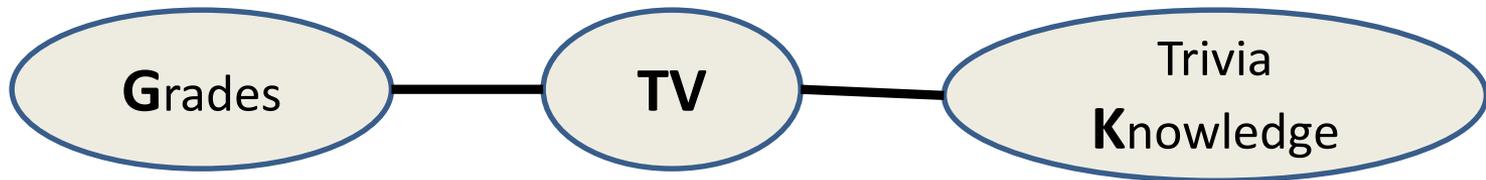


Grades	TV	$\phi_1(G, TV)$
bad	none	2.0
good	none	3.0
bad	lots	3.0
good	lots	1.0

TV	Trivia Knowledge	$\phi_2(TV, K)$
none	weak	2.0
lots	weak	1.0
none	strong	1.5
lots	strong	3.0

Markov Network Inference

- $P(\text{Grades} \mid \text{TV}=\text{none})$? Straightforward: enumerate, then re-normalize

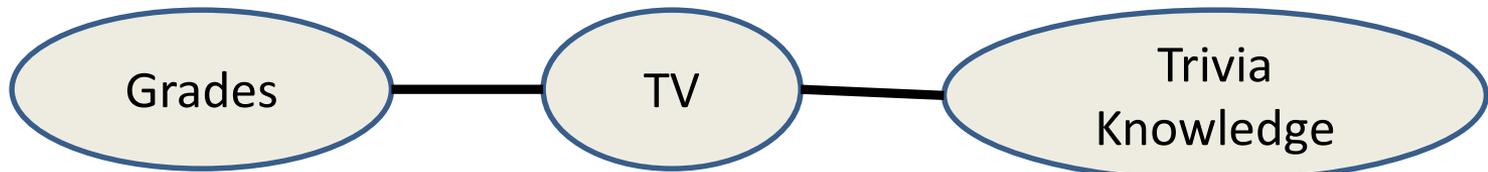


Grades	TV	$\phi_1(G, TV)$
bad	none	2.0
good	none	3.0
bad	lots	3.0
good	lots	1.0

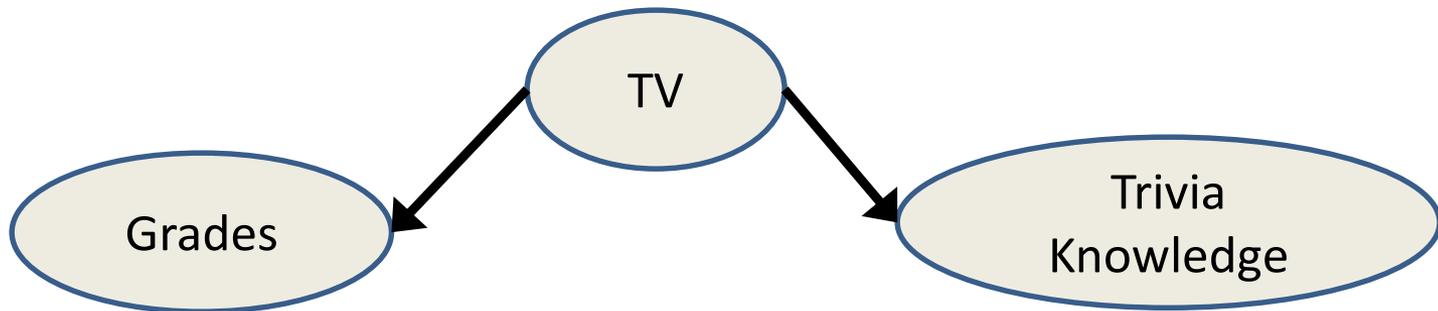
TV	Trivia Knowledge	$\phi_2(TV, K)$
none	weak	2.0
lots	weak	1.0
none	strong	1.5
lots	strong	3.0

But...

- $P(\text{Grades})$? Tougher.



- Need to compute Z , requires summing over Trivia Knowledge as well. Compare with Bayes Net:

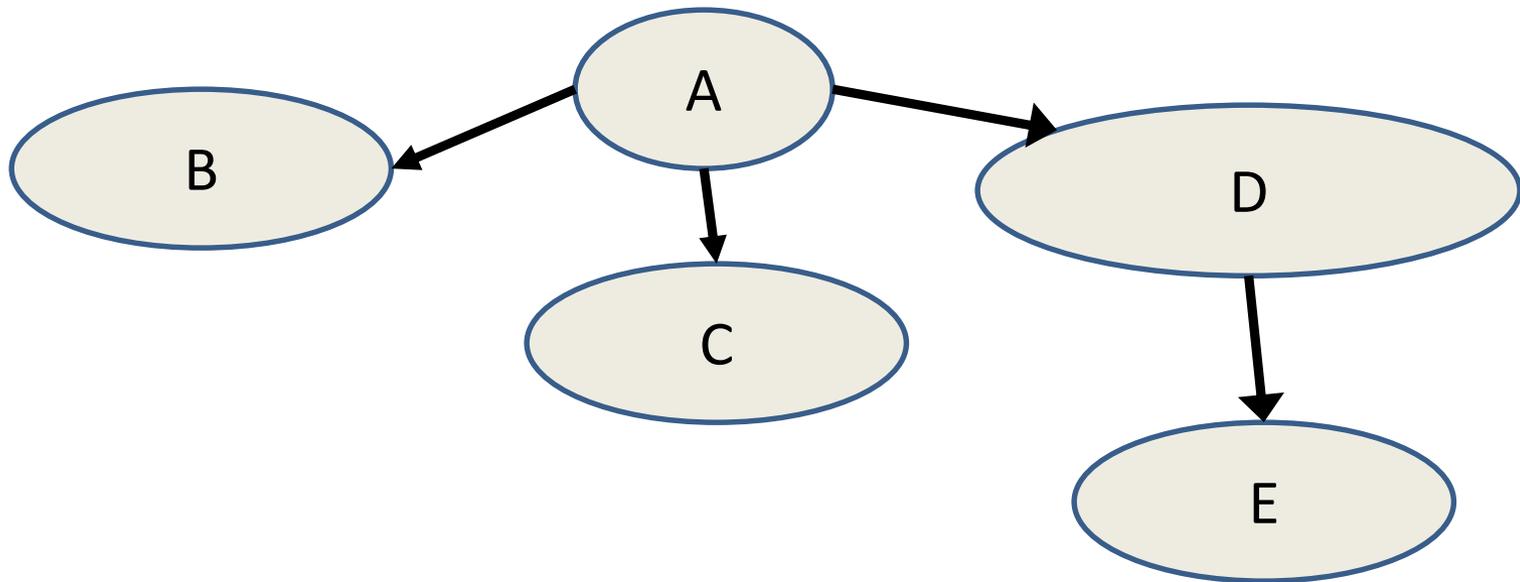


Inference in Markov Networks

- In general, we need to sum over the whole network
- A method for doing so is the *junction-tree* algorithm
 - As a side effect, it computes all the *marginals*
 - $P(\text{Grades}), P(\text{TV}), P(\text{Trivia Knowledge})$
 - Key: can also compute these given evidence
 - We often want to do this for Bayes Nets too
 - Suggests a strategy: convert to Markov Network, then run junction tree algorithm

Junction Tree Algorithm

- High-level Intuition: Computing marginals is straightforward in a tree structure
- Consider a directed Bayes Net for example:

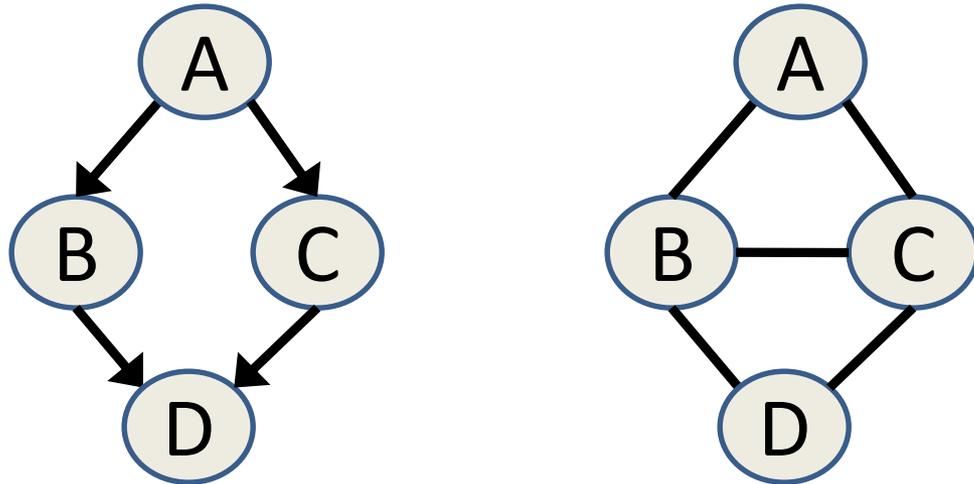


Junction Tree

- Inference of marginals is straightforward in a tree
 - Even if undirected, as we'll see
- Basic idea:
 - **If Bayes Net, convert to Markov Net**
 - Convert Markov Net into a tree structure
 - How?
 - Triangulate, Build Clique Graph, Build Junction Tree
 - Do Inference on Junction Tree

Convert to Markov Net

- Consider this Bayes Net conversion:



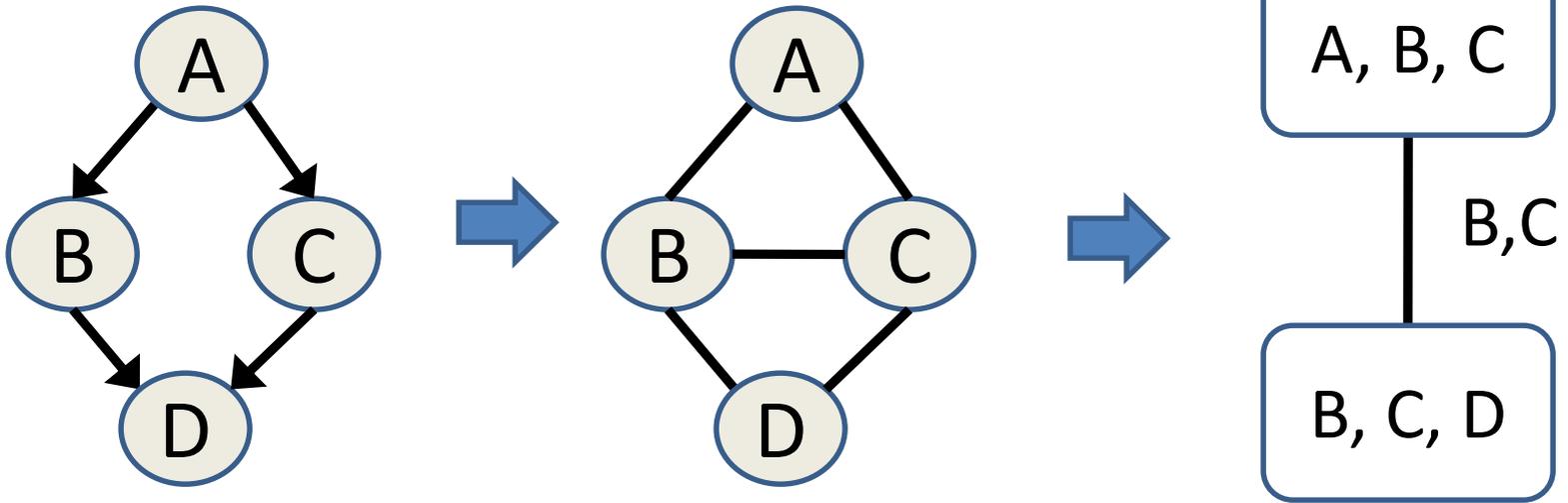
- What are the factors of the Markov Net?

Junction Tree Outline

- If Bayes Net, convert to Markov Net
- **Convert Markov Net into Junction Tree**
 - Triangulate
 - Build Clique Graph
 - Build Junction Tree
- Do Inference using Junction Tree

Convert Markov Net into Junction Tree

- Punchline:



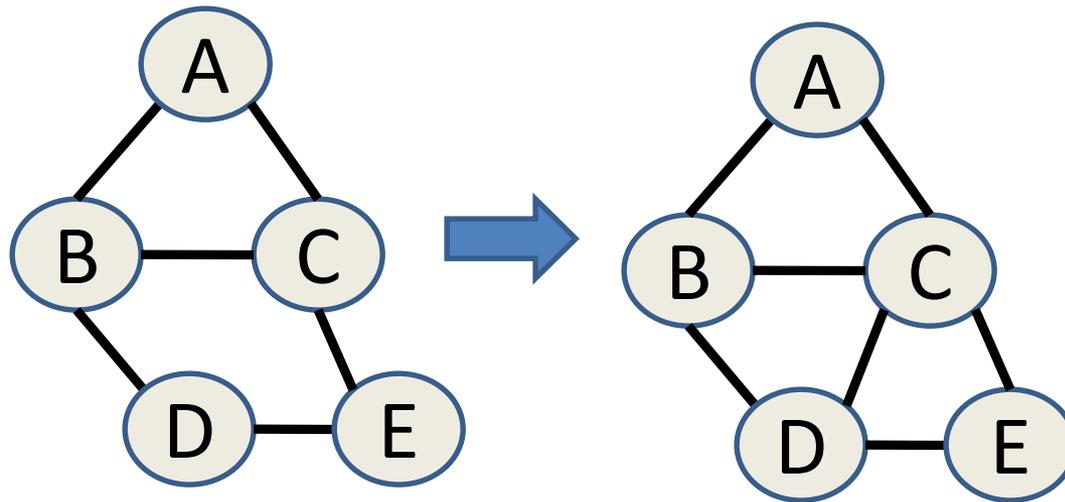
- Details follow

Junction Tree Outline

- If Bayes Net, convert to Markov Net
- Convert Markov Net into Junction Tree
 - **Triangulate**
 - Build Clique Graph
 - Build Junction Tree
- Do Inference using Junction Tree

Triangulation => “Chordal” Graph

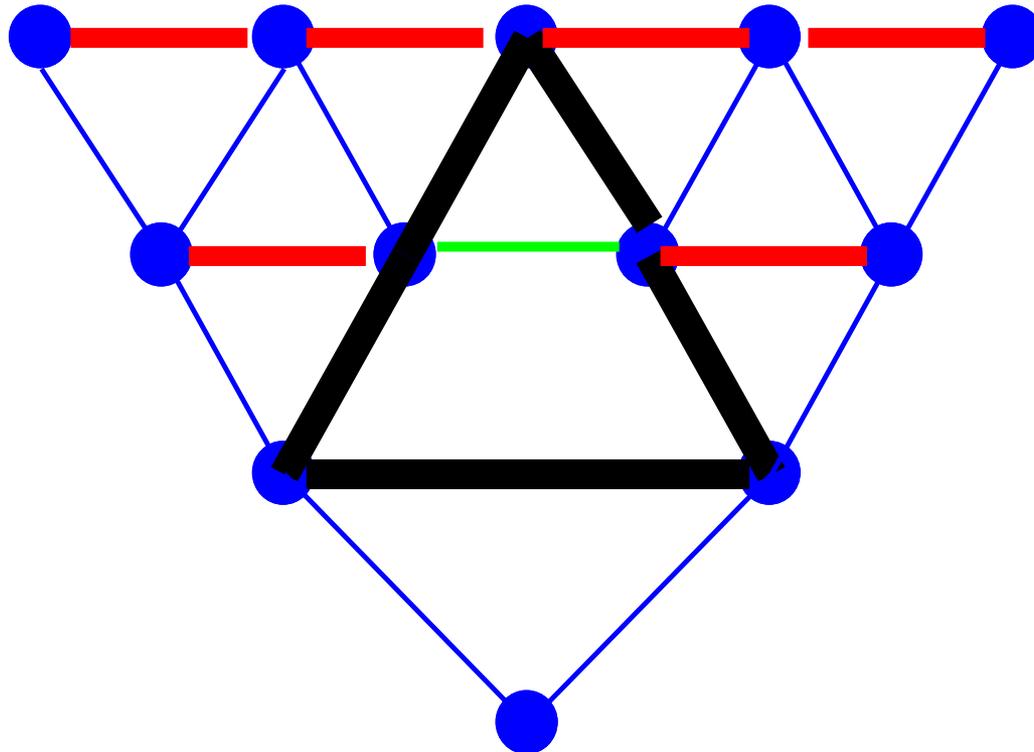
- Goal: Every cycle of length > 3 has a chord



- Why? Stay tuned.

Triangulation Algorithm

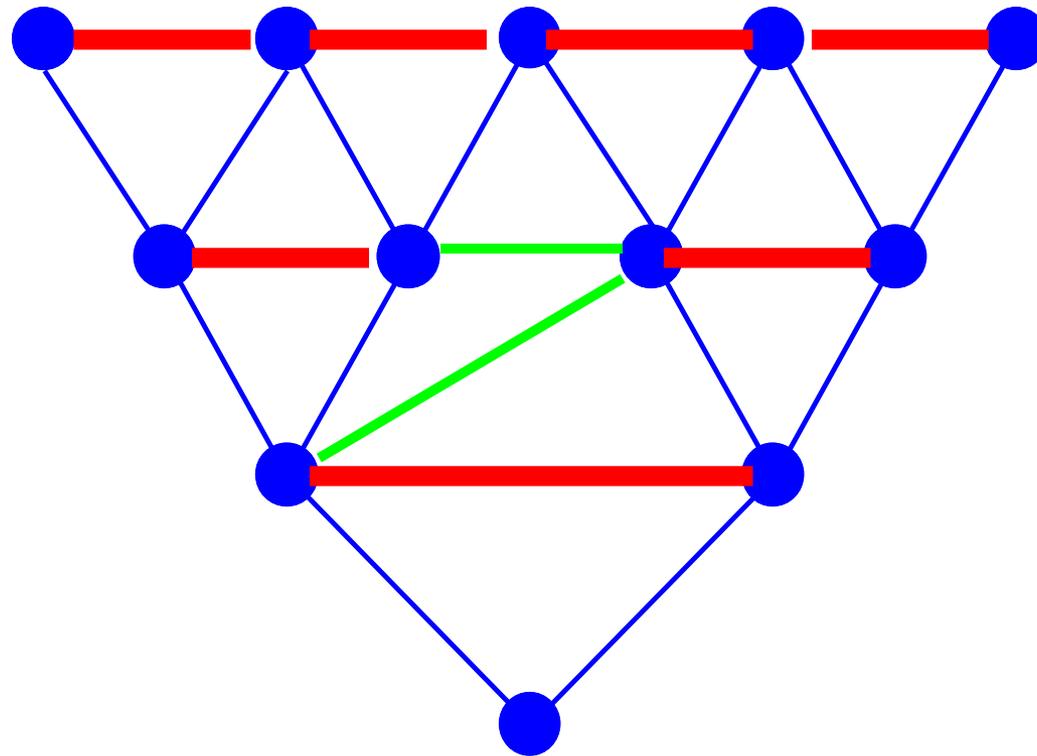
Repeat while there exists a cycle of length > 3 with no chord:
Add a chord (edge between two non-adjacent vertices in such a cycle).



- From David Page, UWisc, pages.cs.wisc.edu/~dpage/cs731/lecture5.ppt

Triangulation Checking (1 of 3)

It appears to be triangulated, but how can we be sure?



- From David Page, UWisc, pages.cs.wisc.edu/~dpage/cs731/lecture5.ppt

Triangulation Checking (2 of 3)

Input: Graph G with n nodes

Output: “Is G triangulated?”

Algorithm:

Choose any node, label it 1

for $i = 2$ to n

 Find node with most labeled neighbors, label it i

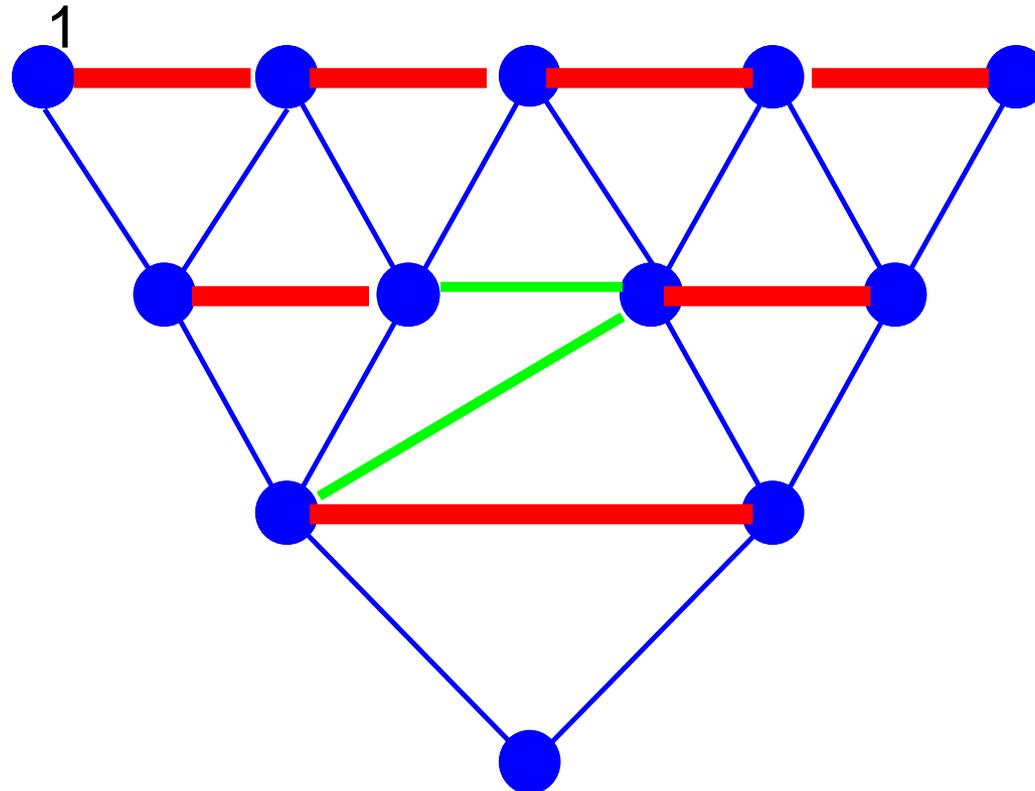
 if i has two non-adjacent labeled neighbors

 return false

return true

Triangulation Checking (3 of 3)

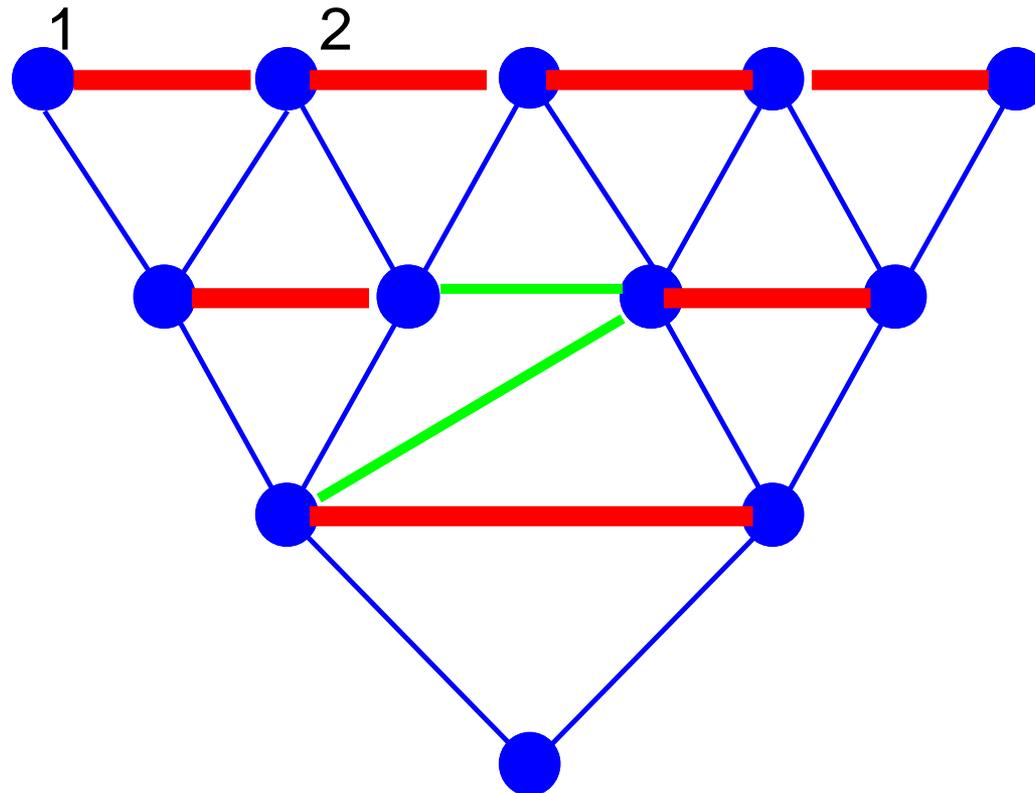
It appears to be triangulated, but how can we be sure?



- From David Page, UWisc, pages.cs.wisc.edu/~dpage/cs731/lecture5.ppt

Triangulation Checking (3 of 3)

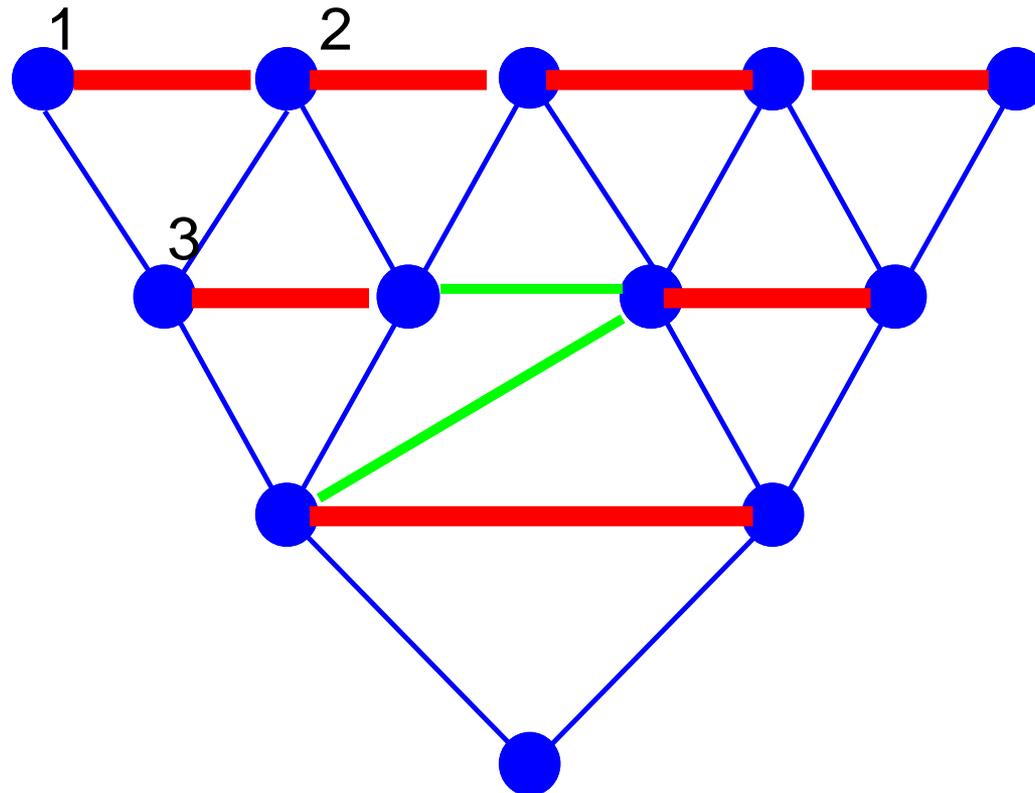
It appears to be triangulated, but how can we be sure?



- From David Page, UWisc, pages.cs.wisc.edu/~dpage/cs731/lecture5.ppt

Triangulation Checking (3 of 3)

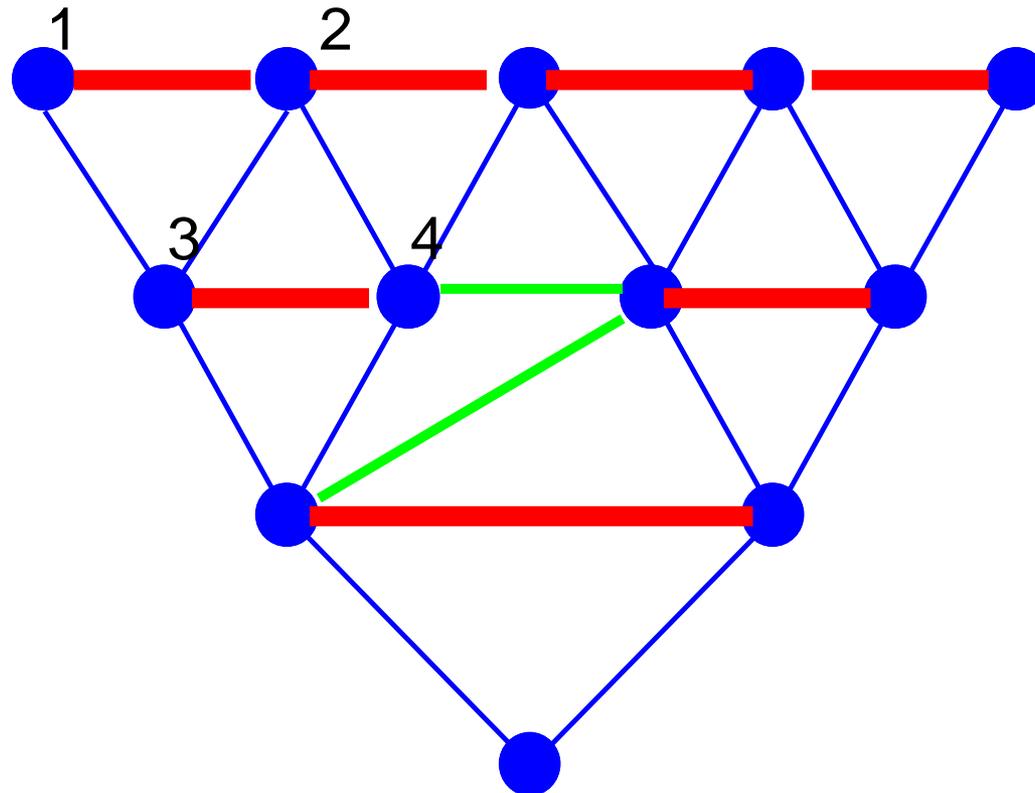
It appears to be triangulated, but how can we be sure?



- From David Page, UWisc, pages.cs.wisc.edu/~dpage/cs731/lecture5.ppt

Triangulation Checking (3 of 3)

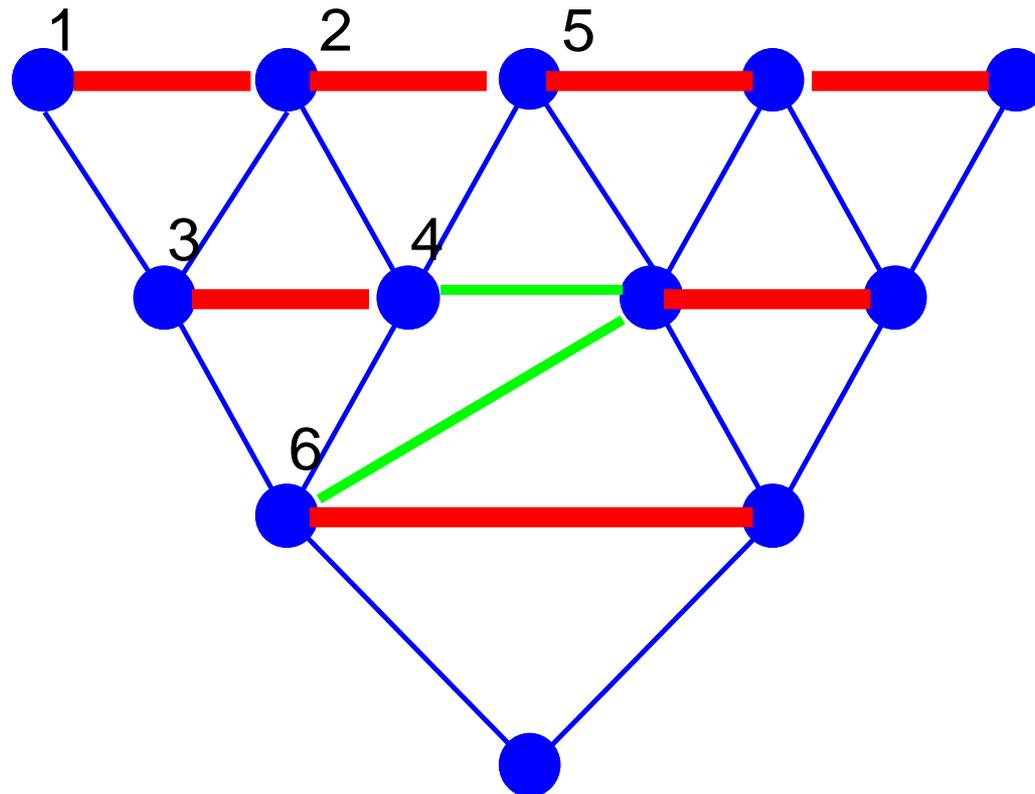
It appears to be triangulated, but how can we be sure?



- From David Page, UWisc, pages.cs.wisc.edu/~dpage/cs731/lecture5.ppt

Triangulation Checking (3 of 3)

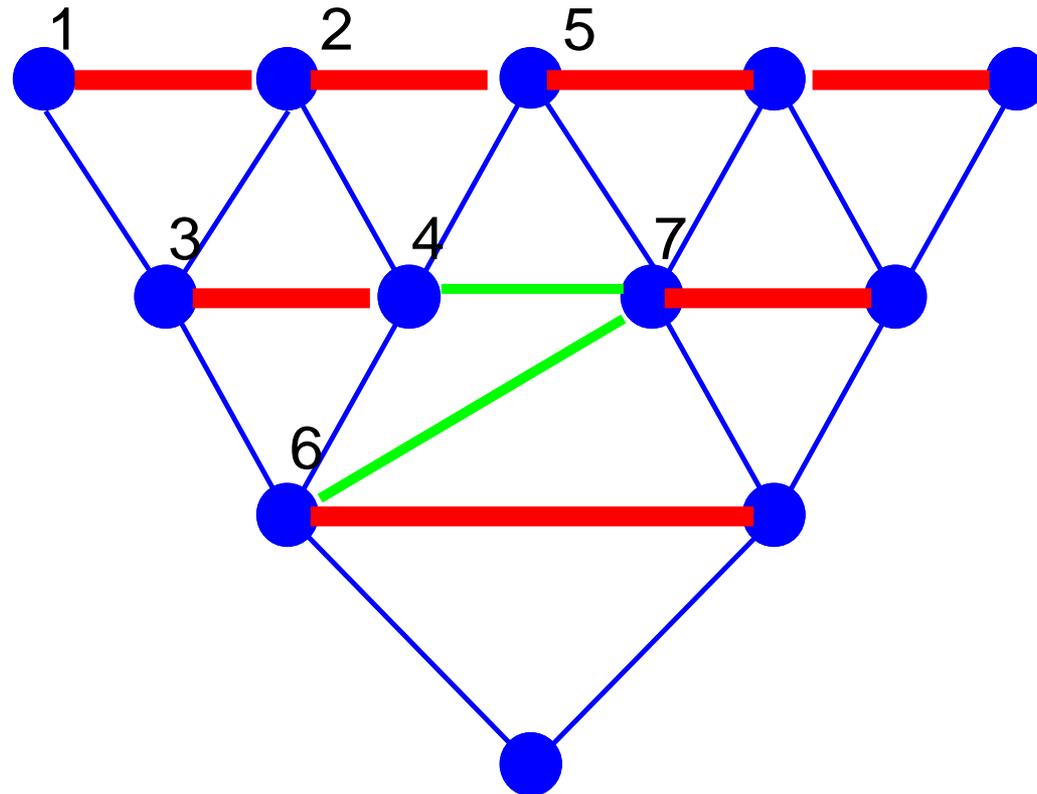
It appears to be triangulated, but how can we be sure?



- From David Page, UWisc, pages.cs.wisc.edu/~dpage/cs731/lecture5.ppt

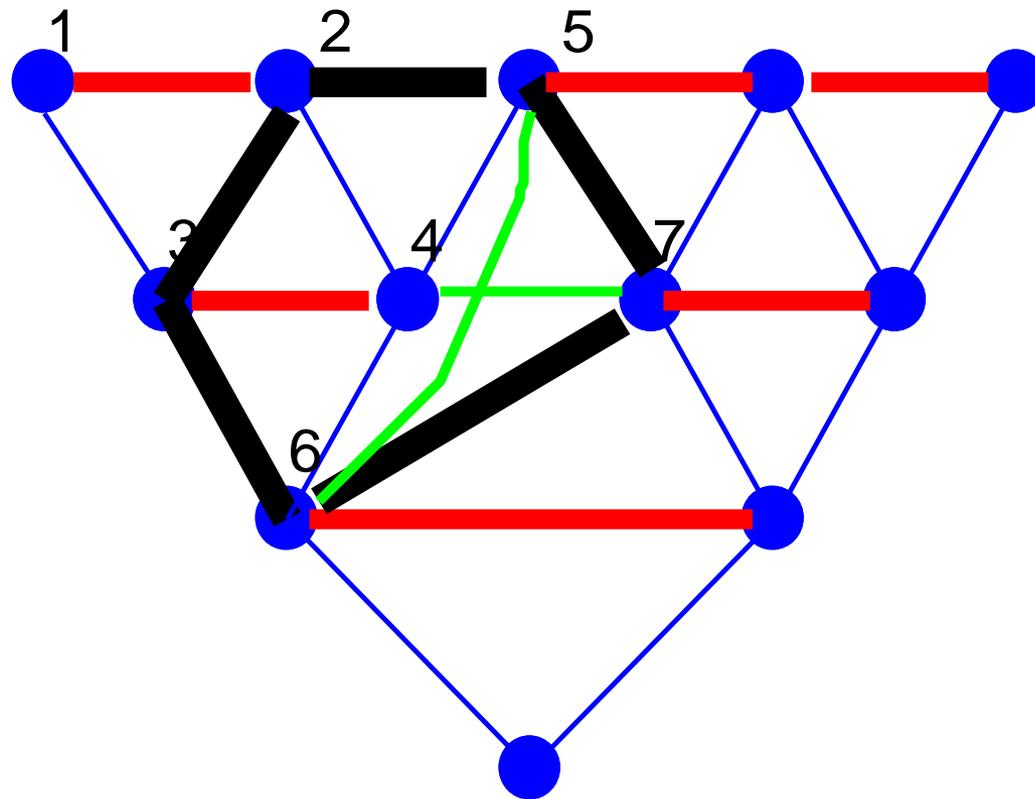
Triangulation Checking (3 of 3)

No edge between nodes 5 and 6, both of which are parents of 7.



- From David Page, UWisc, pages.cs.wisc.edu/~dpage/cs731/lecture5.ppt

Connect the two offending nodes



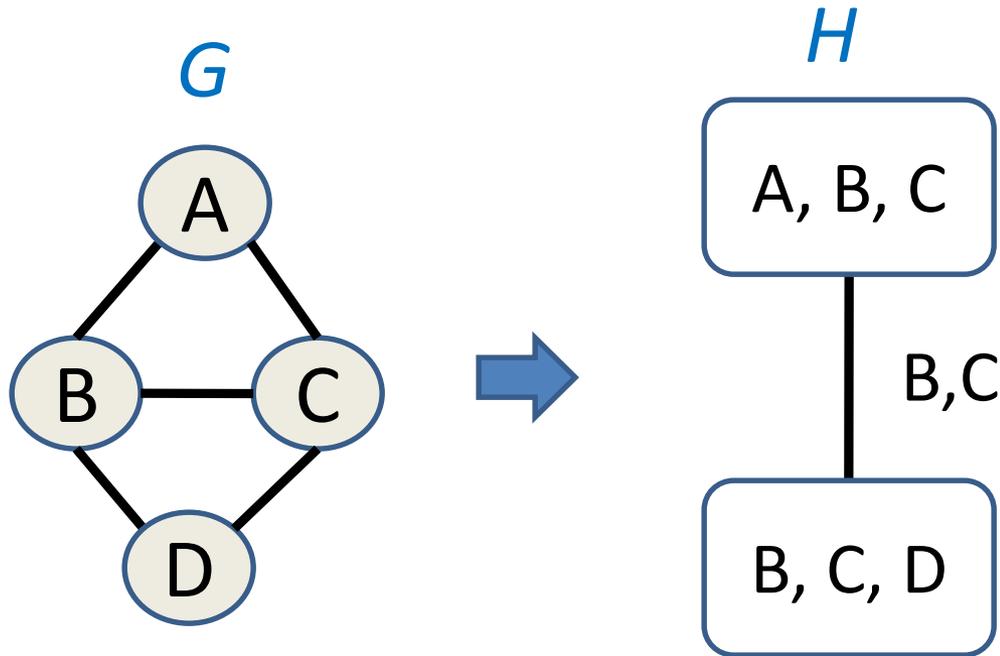
- From David Page, UWisc, pages.cs.wisc.edu/~dpage/cs731/lecture5.ppt

Junction Tree Outline

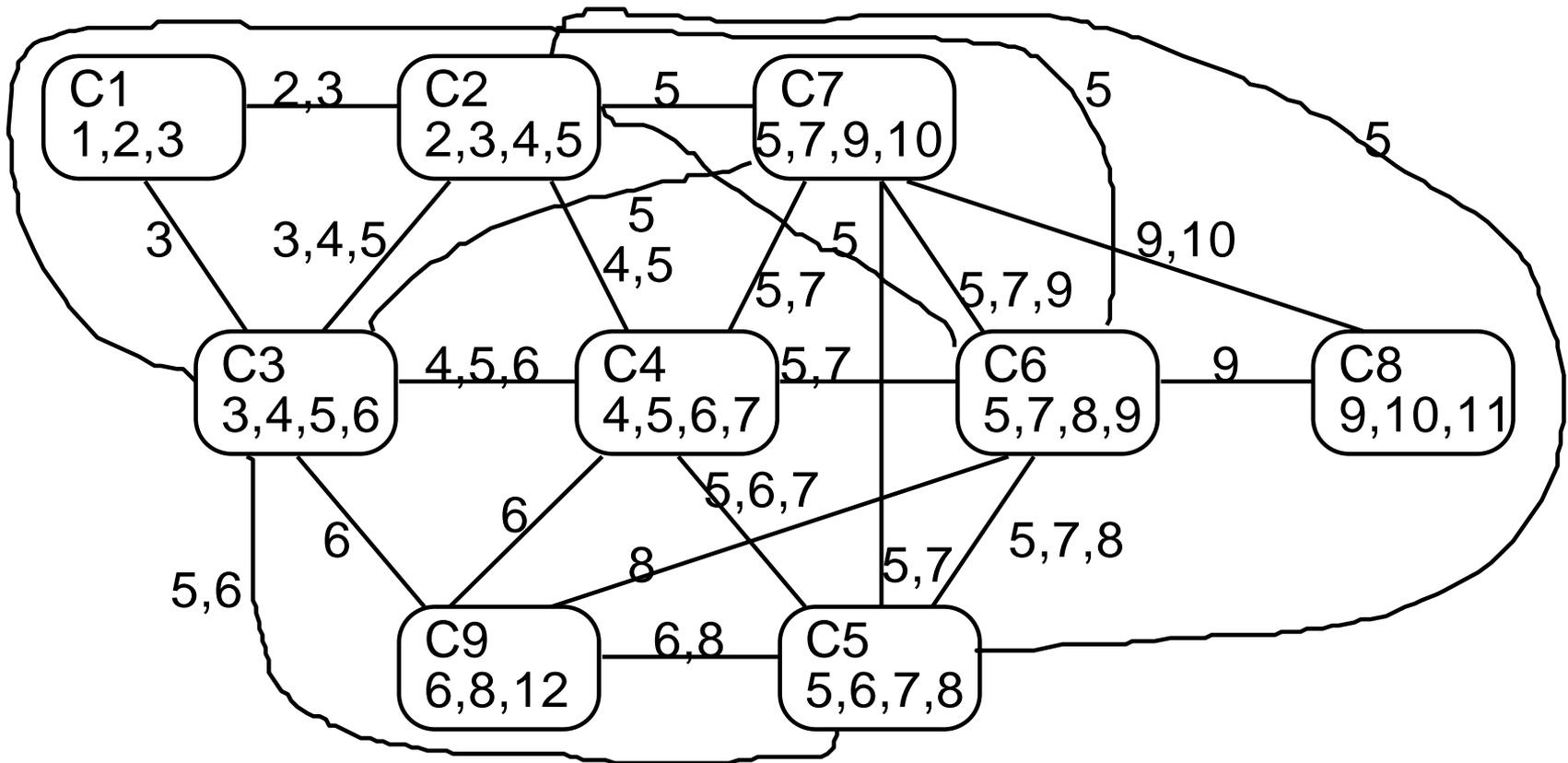
- If Bayes Net, convert to Markov Net
- Convert Markov Net into Junction Tree
 - Triangulate
 - **Build Clique Graph**
 - Build Junction Tree
- Do Inference using Junction Tree

Building Clique Graph H

- Create a **node** in H for each maximal clique in G
- Create **edges** in H between adjacent cliques in G
 - Convenience: Label edges in H with nodes' intersection



Bigger Example – Clique Graph



- From David Page, UWisc, pages.cs.wisc.edu/~dpage/cs731/lecture5.ppt

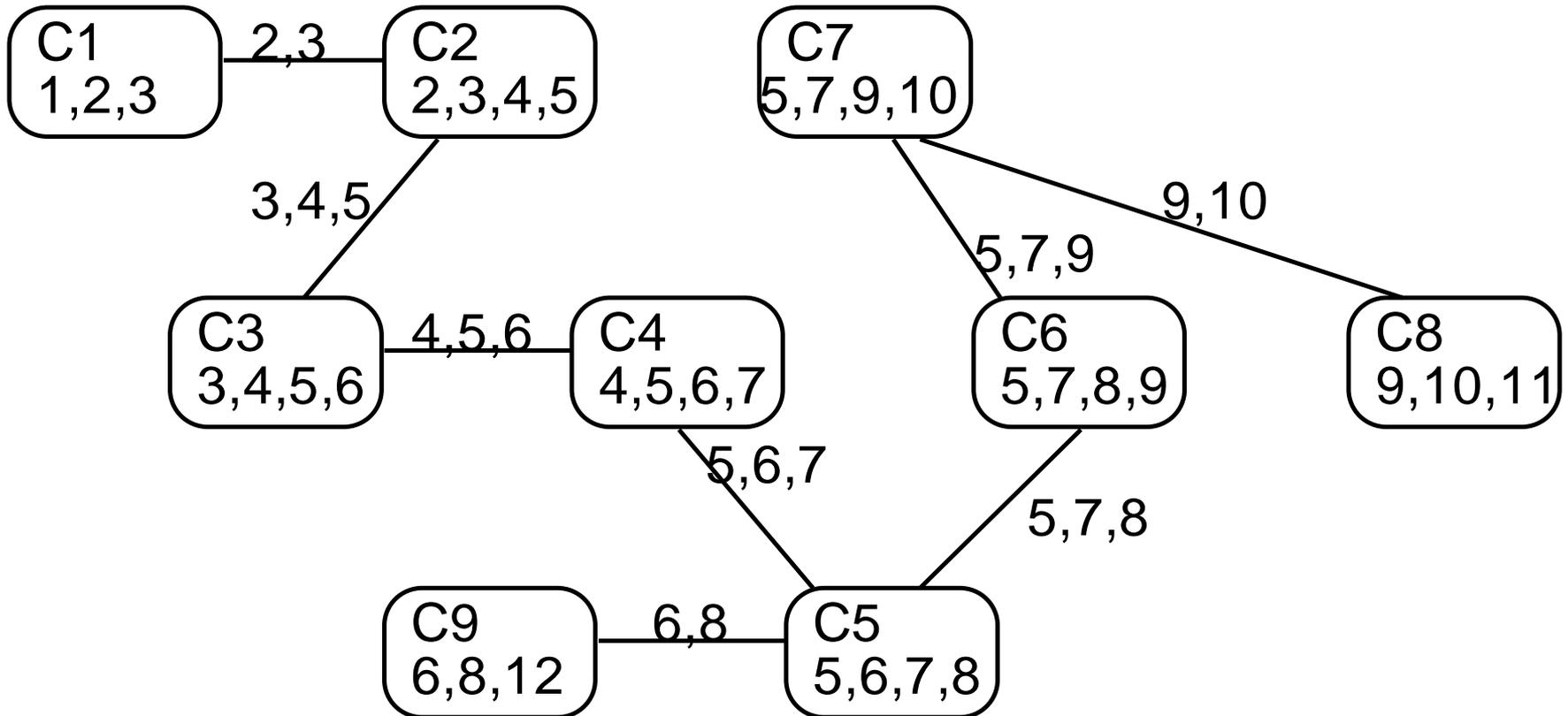
Junction Tree Outline

- If Bayes Net, convert to Markov Net
- Convert Markov Net into Junction Tree
 - Triangulate
 - Build Clique Graph
 - **Build Junction Tree**
- Do Inference using Junction Tree

Build Junction Tree

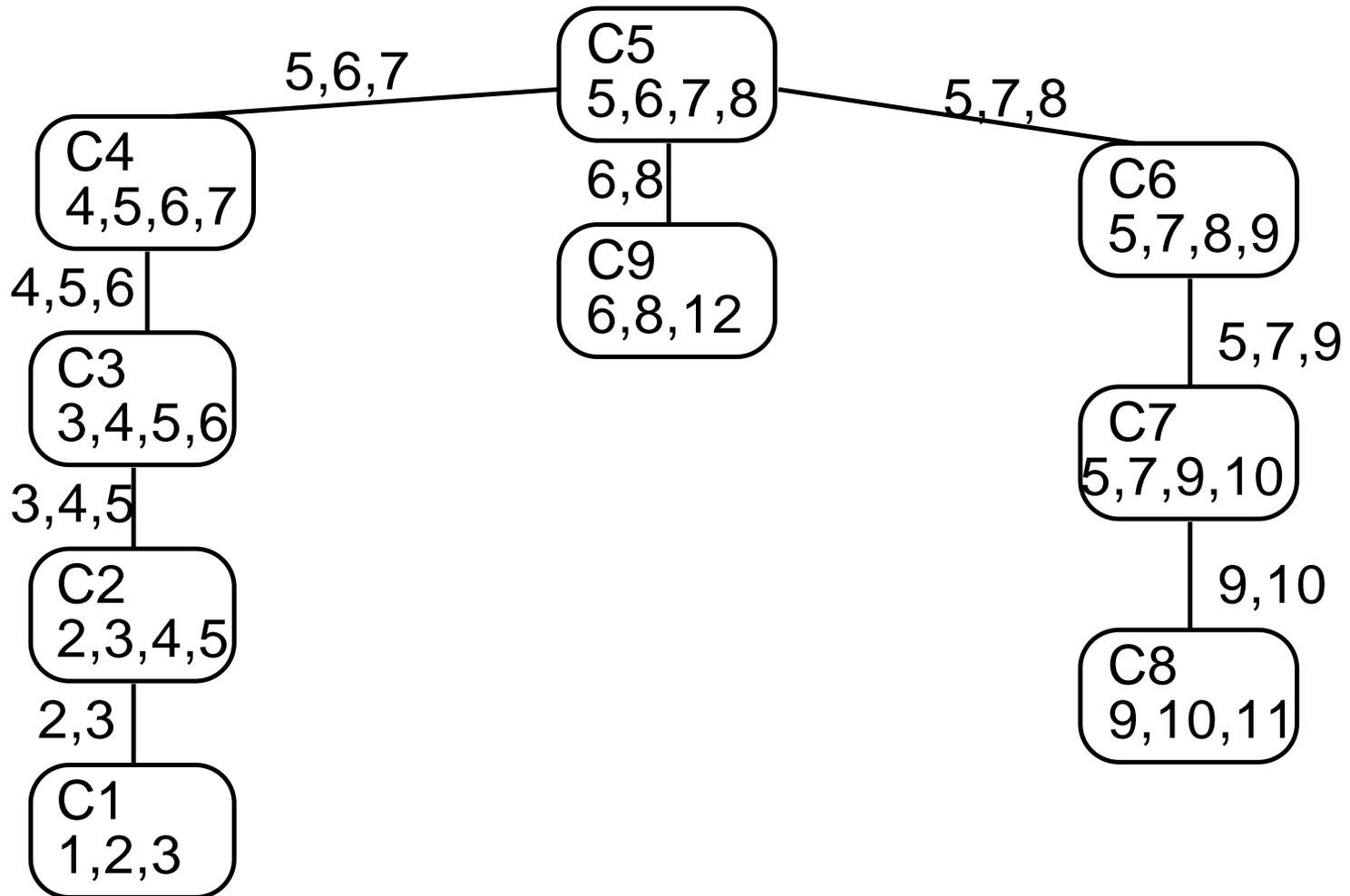
- A **Junction Tree** is a subgraph of the clique graph that
 - Is a tree
 - Contains all the nodes of the clique graph
 - Satisfies the **junction tree property**
 - For each pair of cliques U, V with intersection S , all cliques on path between U and V contain S

Junction Tree Example



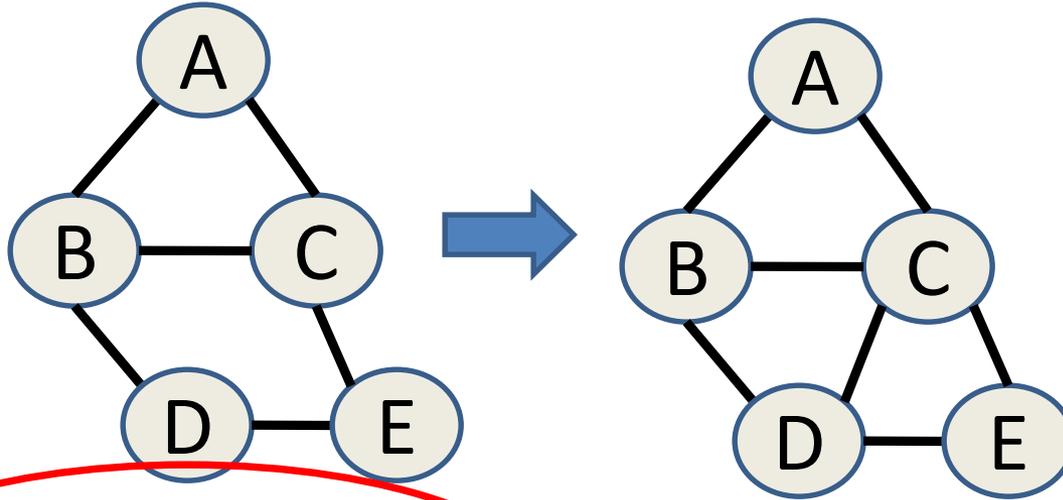
- From David Page, UWisc, pages.cs.wisc.edu/~dpage/cs731/lecture5.ppt

Choose a Root



Remember This?

- Goal: Every cycle of length > 3 has a chord



- Why? Stay tuned.

Can we always find a Junction Tree?

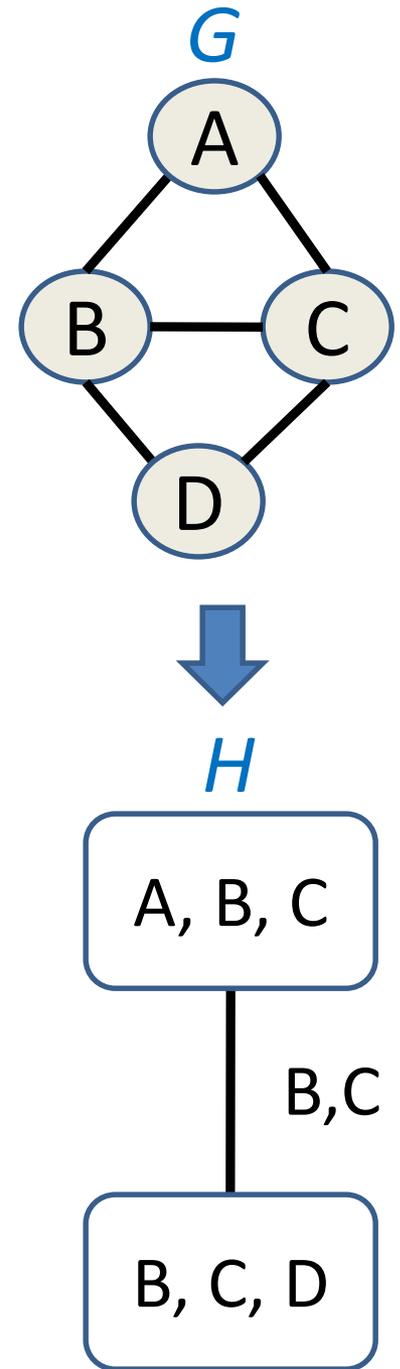
- Yes, for clique graphs of triangulated graphs
 - Define “edge weight” on the clique graph to be the size of the intersection
 - Then a maximum-weight spanning tree is a junction tree
- [Jensen & Jensen, 1994]

Junction Tree

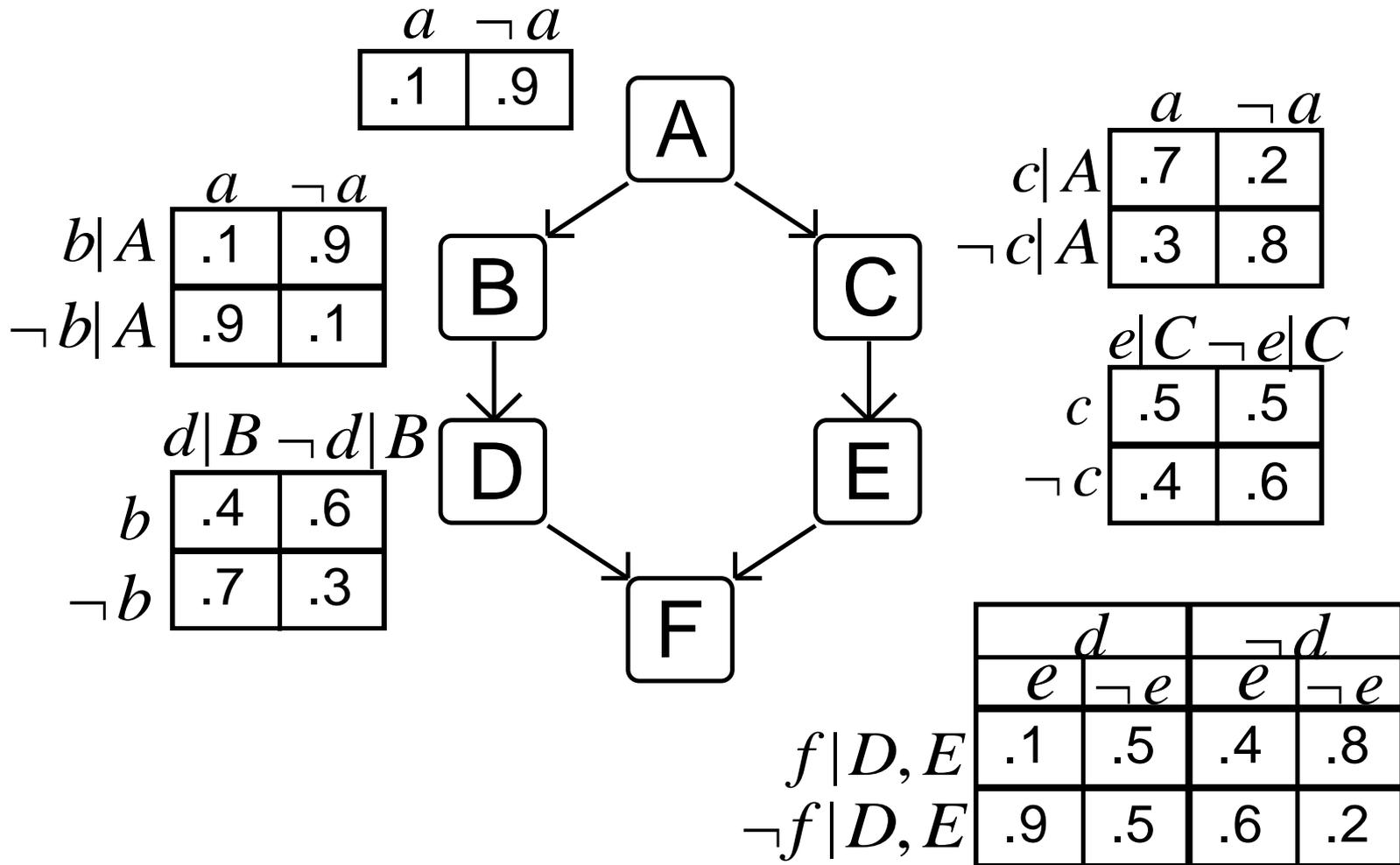
- If Bayes Net, convert to Markov Net
- Convert Markov Net into Tree
 - Triangulate
 - Build Clique Graph
 - Build Junction Tree
- **Do Inference on Tree**

Inference

- **Initialize** clique nodes
 - Clique node in H is a table assigning values to its variable combinations
 - Put each potential function (or CPT) in G into exactly one node in H
 - Combine by multiplying “pointwise” (as in variable elimination)

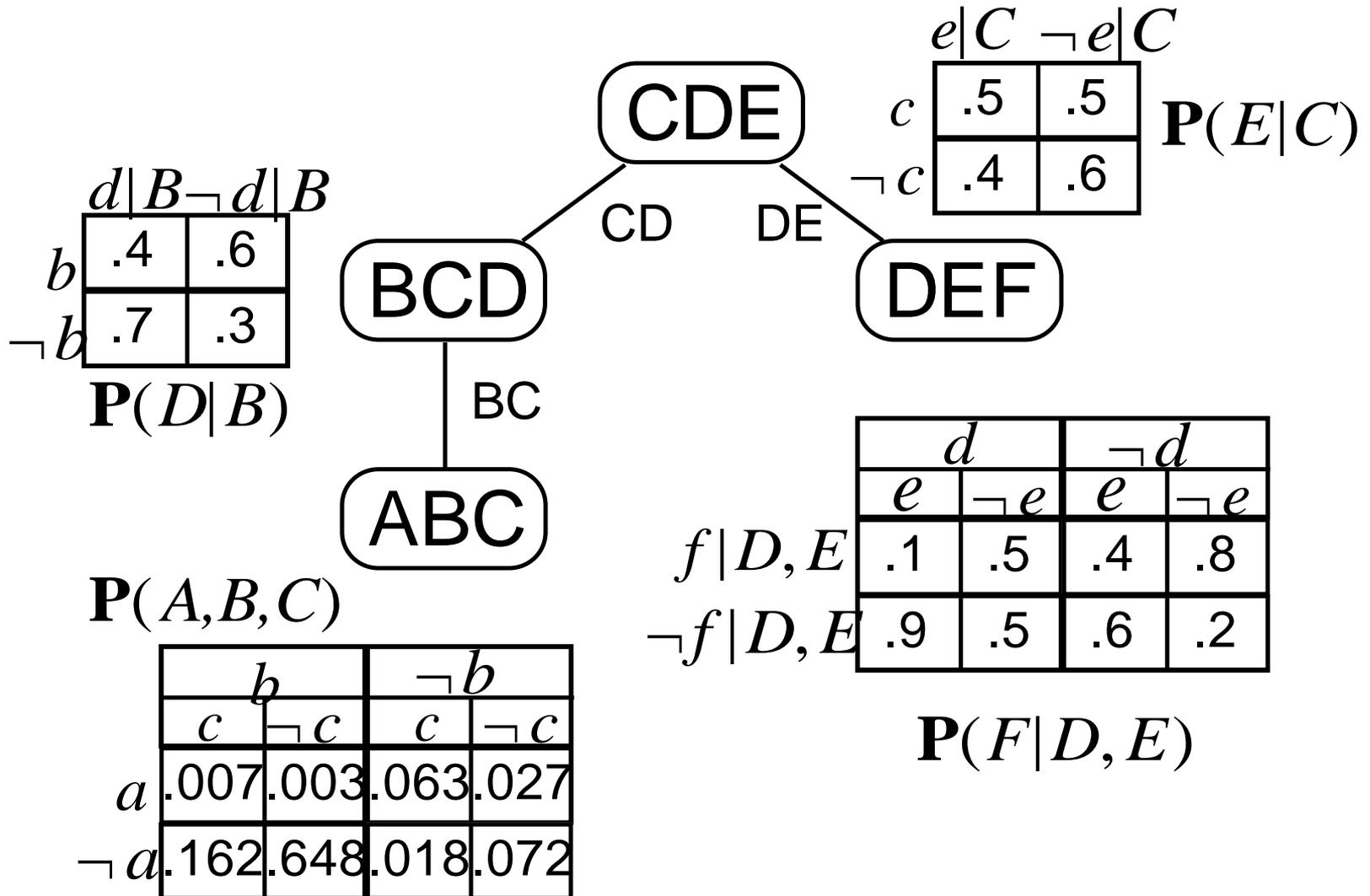


Example



- From David Page, UWisc, pages.cs.wisc.edu/~dpage/cs731/lecture5.ppt

Junction Tree with CPTs



- From David Page, UWisc, pages.cs.wisc.edu/~dpage/cs731/lecture5.ppt

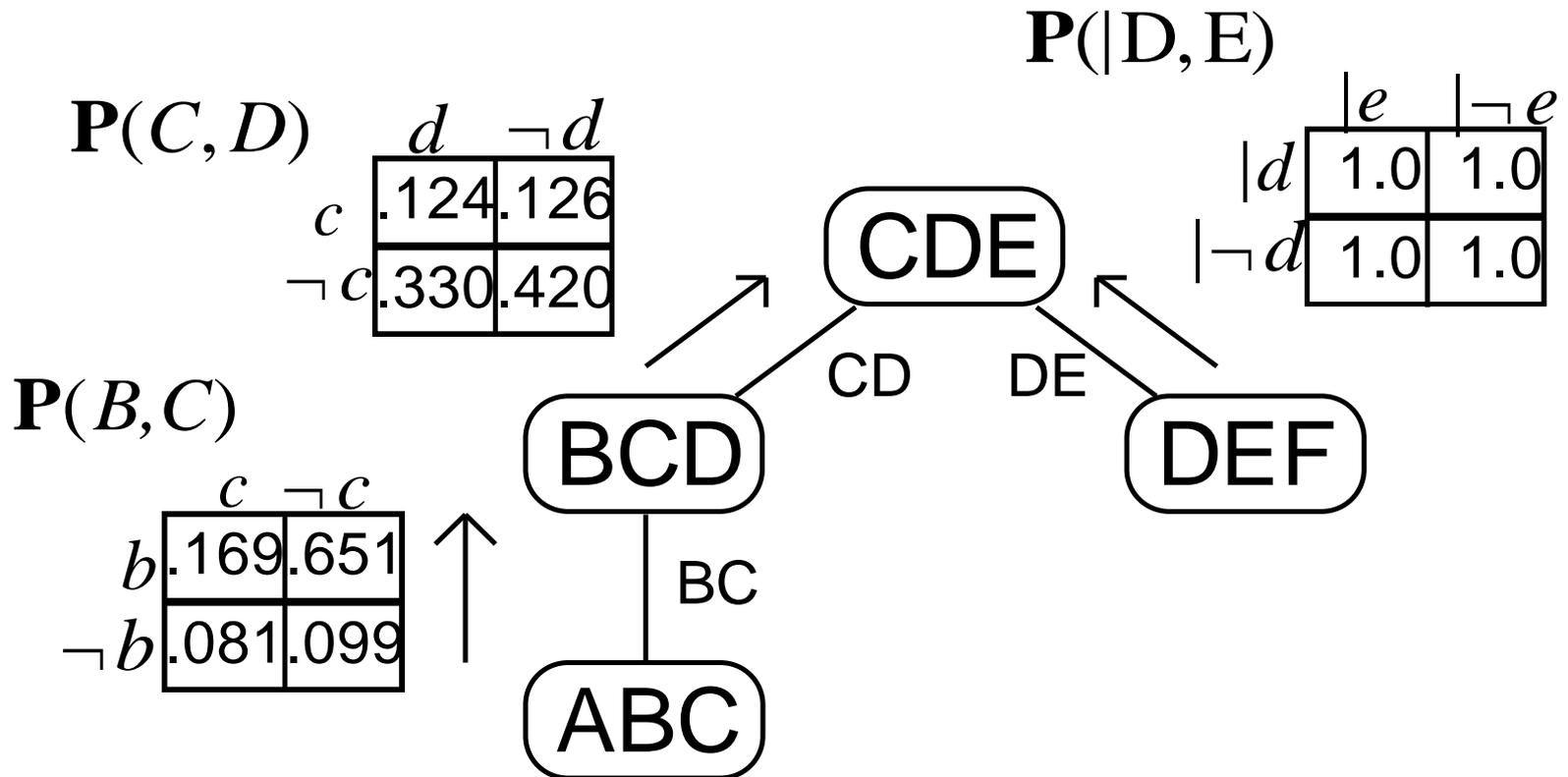
Junction Tree Algorithm

- Incorporate Evidence -- For each $E = e$
 - Find one junction tree node containing E
 - Zero out all cells with $E \neq e$
- Upward Pass (from leaves to root)
 - Each leaf sends message to parent
 - Message = leaf's table after summing out variables not in parent
 - Parent propagates message
 - Multiplies in the child's message, then repeats process

Junction Tree Algorithm

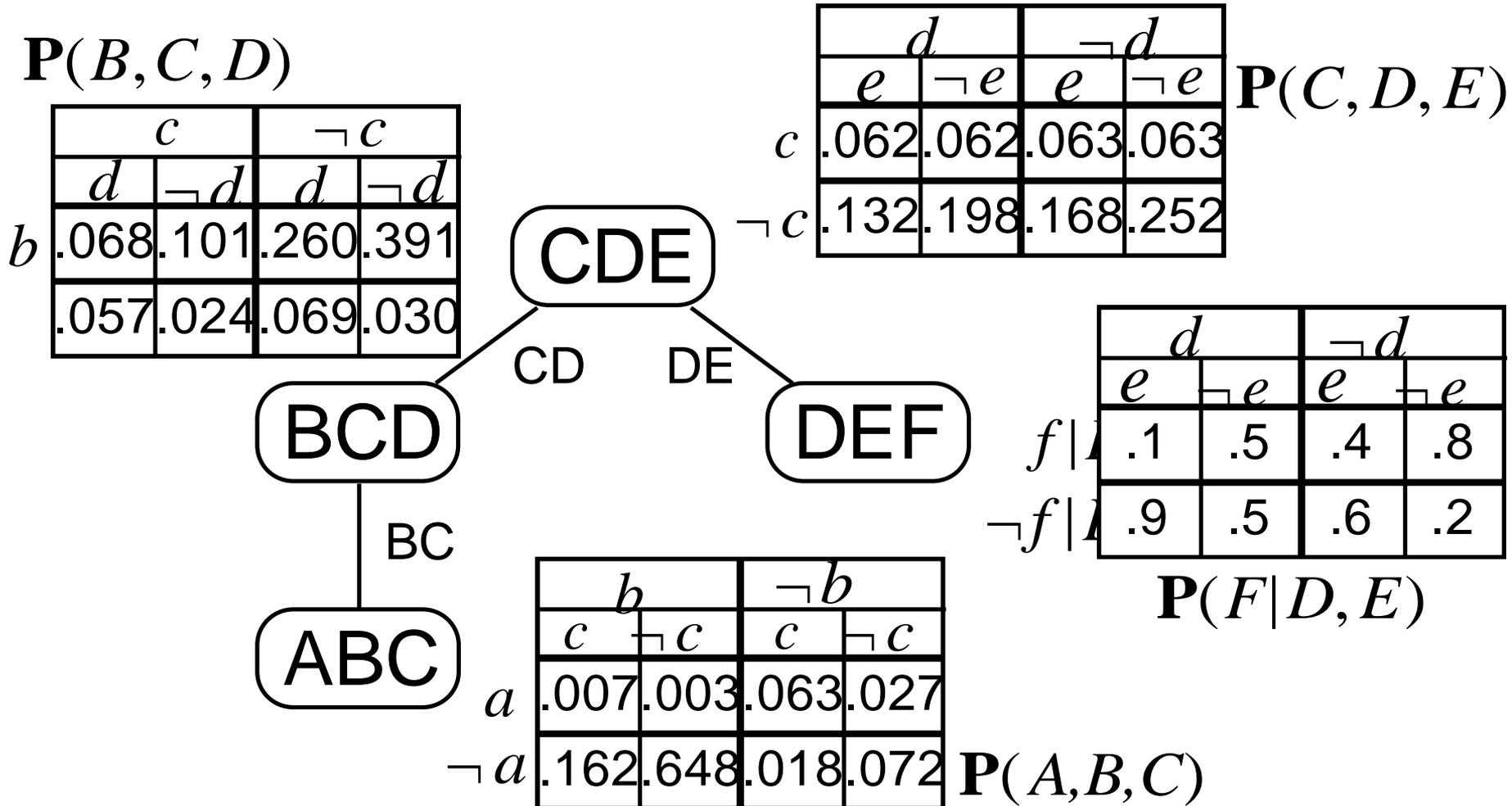
- Downward Pass
 - Root sends child a message
 - Divides its table by child's message from upward pass
 - Sums out variables not in child, and sends
 - Child propagates the message
 - After multiplying in parent's message, child's table is the joint distribution over its variables
 - Child continues the process (acts as root)

Upward Pass – assume no evidence



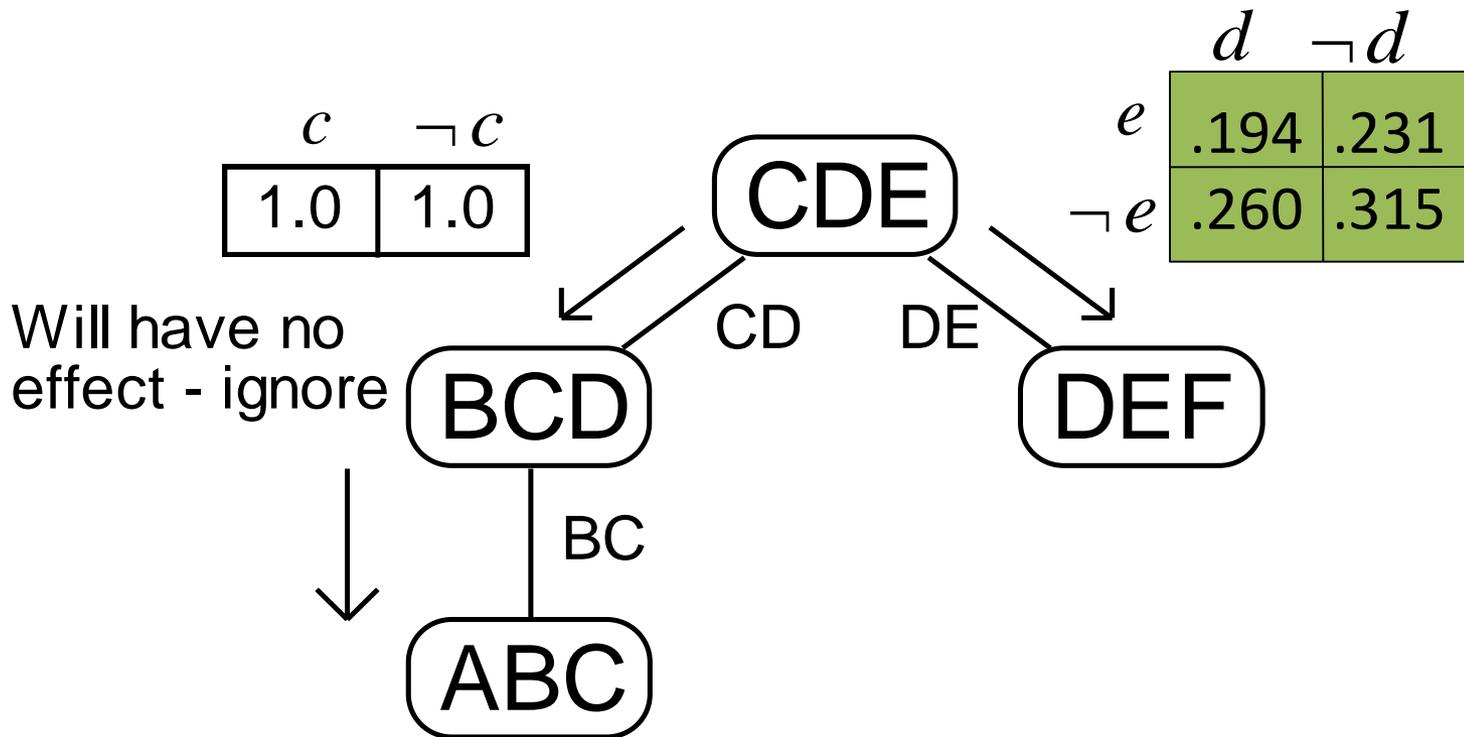
- From David Page, UWisc, pages.cs.wisc.edu/~dpage/cs731/lecture5.ppt

Status After Upward Pass



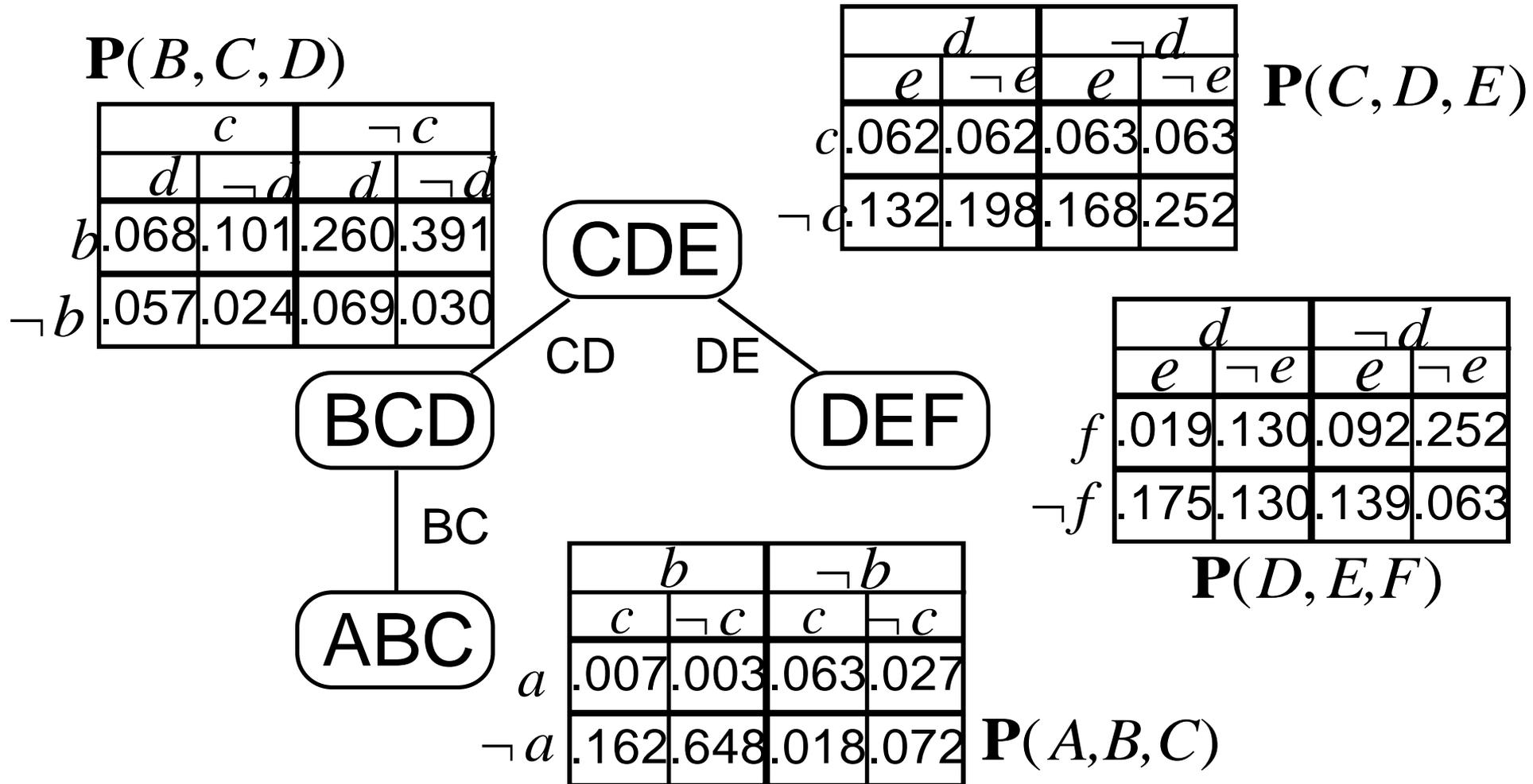
- From David Page, UWisc, pages.cs.wisc.edu/~dpage/cs731/lecture5.ppt

Downward Pass



- From David Page, UWisc, pages.cs.wisc.edu/~dpage/cs731/lecture5.ppt

Status After Downward Pass



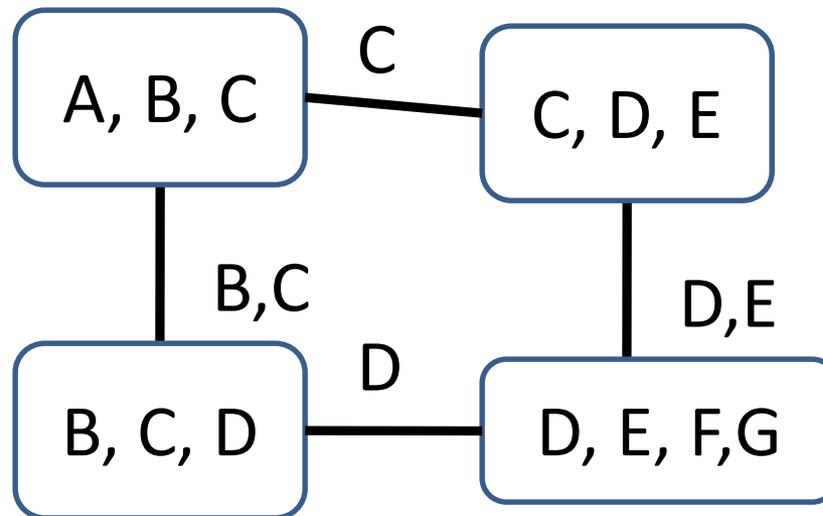
- From David Page, UWisc, pages.cs.wisc.edu/~dpage/cs731/lecture5.ppt

Remember Junction Tree Property

- A **Junction Tree** is a subgraph of the clique graph that
 - Is a tree
 - Contains all the nodes of the clique graph
 - Satisfies the junction tree property
 - For each pair of cliques U, V with intersection S , all cliques on path between U and V contain S

Why a Tree?

- Consider the alternative – cycles:



- Previous algorithm not applicable -- can't define upward, downward pass

Finishing touches

- We have joint distributions
 - $P(A, B, C)$, $P(C, D, E)$, etc.
- Compute marginals by summing out
 - Key: These sums are over small #s of variables
- If evidence changes, we repeat forward-backward pass
 - BUT we don't have to re-compute the junction tree (= savings)