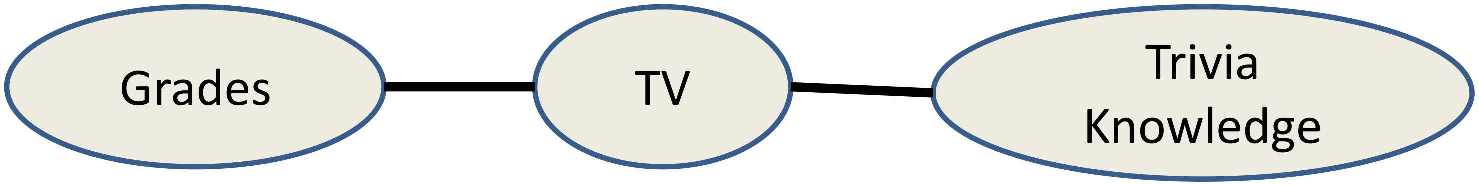
Inference in Markov Networks

Doug Downey
Northwestern EECS 395/495 Fall 2011

Markov Network Inference

- $P(\boldsymbol{x}) = \frac{\Pi_c \phi_c(\boldsymbol{x}_c)}{Z}$ $\quad Z = \Sigma_{\boldsymbol{x}} \Pi_c \phi_c(\boldsymbol{x}_c)$

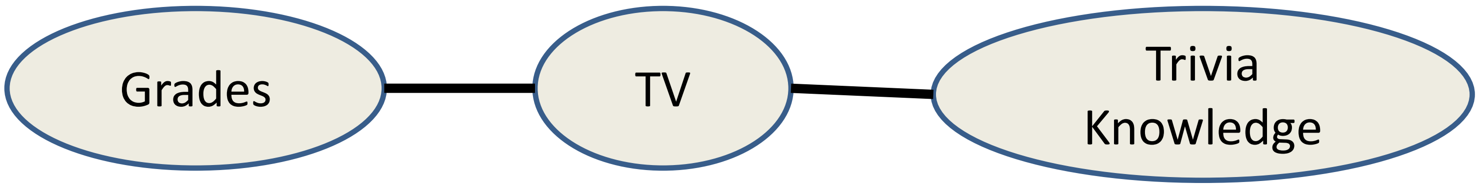


Grades	TV	ϕ(A, B)
Low	Little	2.0
Good	Little	3.0
Low	Lots	3.0
Good	Lots	1.0

TV	Trivia Knowledge	ϕ(A, B)
Little	Little	2.0
Lots	Little	1.0
Little	Lots	1.5
Lots	Lots	3.0

Markov Network Inference

- P(Grades | TV=Little)? Straightforward: enumerate, then re-normalize

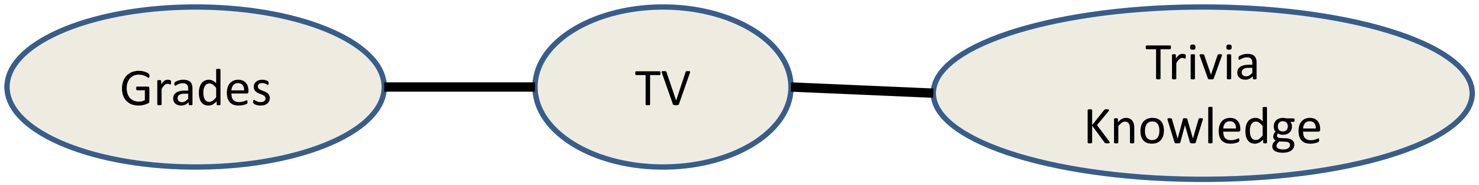


Grades	TV	ϕ(A, B)
Low	Little	2.0
Good	Little	3.0
Low	Lots	3.0
Good	Lots	1.0

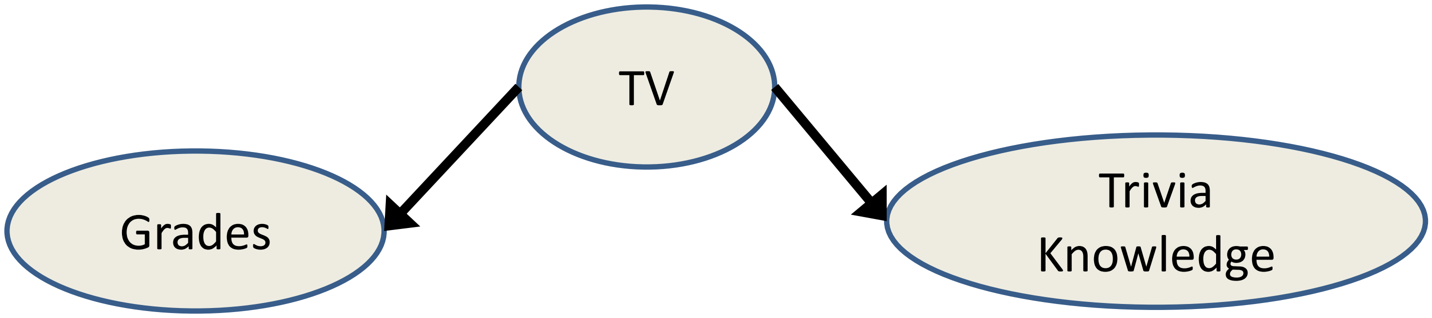
TV	Trivia Knowledge	ϕ(A, B)
Little	Little	2.0
Lots	Little	1.0
Little	Lots	1.5
Lots	Lots	3.0

But…

- P(Grades)? Tougher.



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

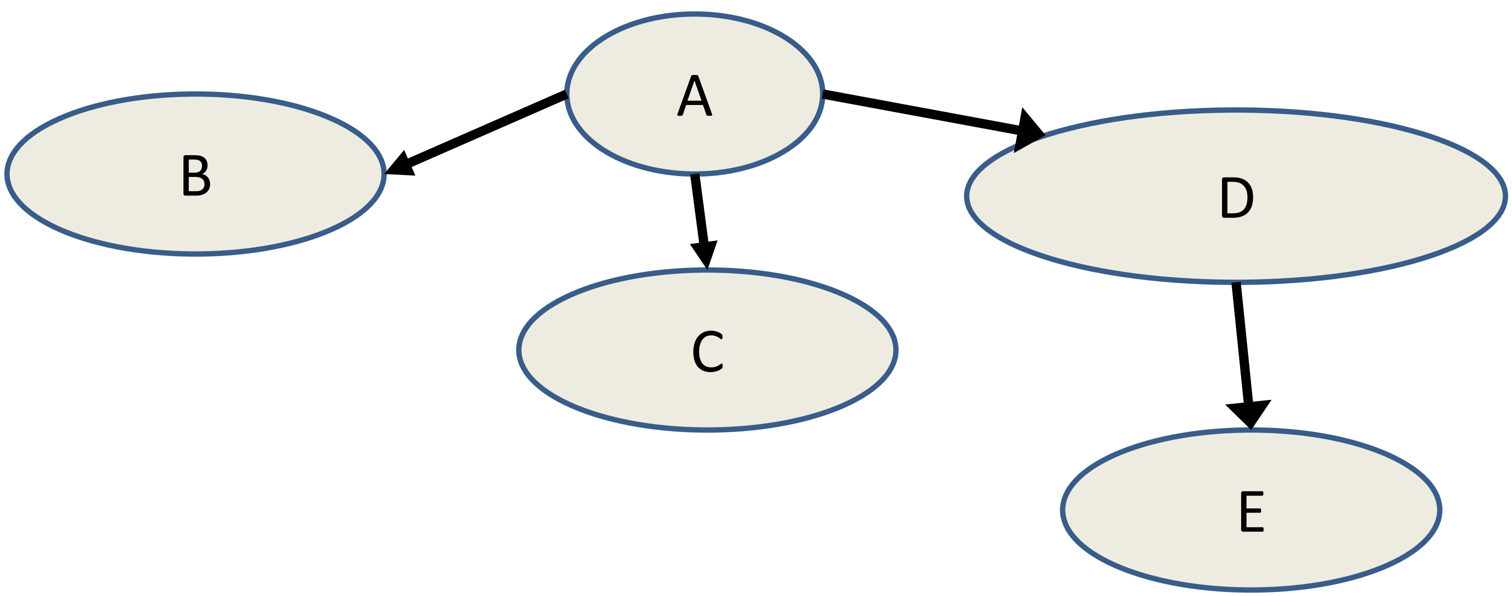


Inference in Markov Networks

- 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*(*Grades*), *P*(*TV*), *P*(*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:

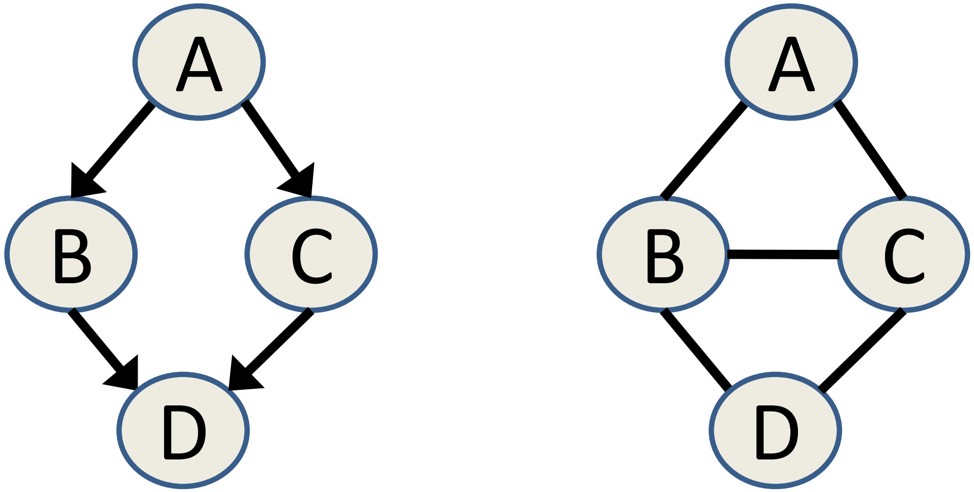


Junction Tree

- 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:



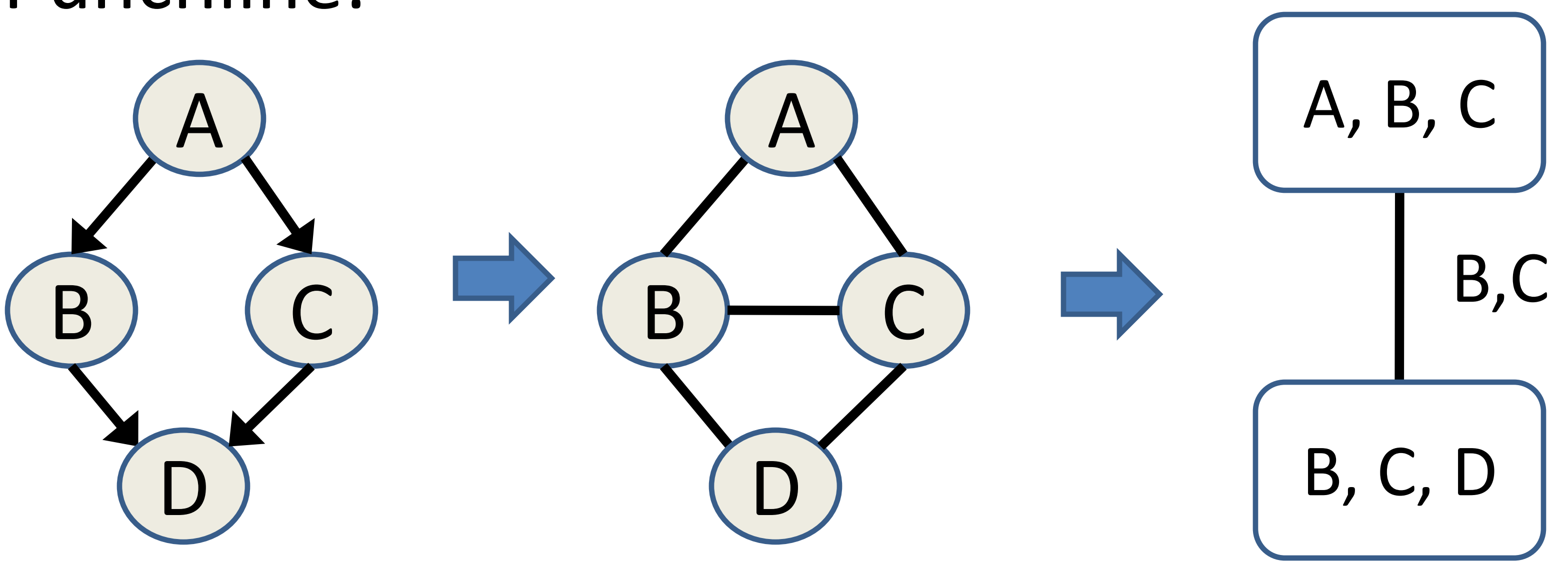
- What are the factors of the Markov Net?

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:



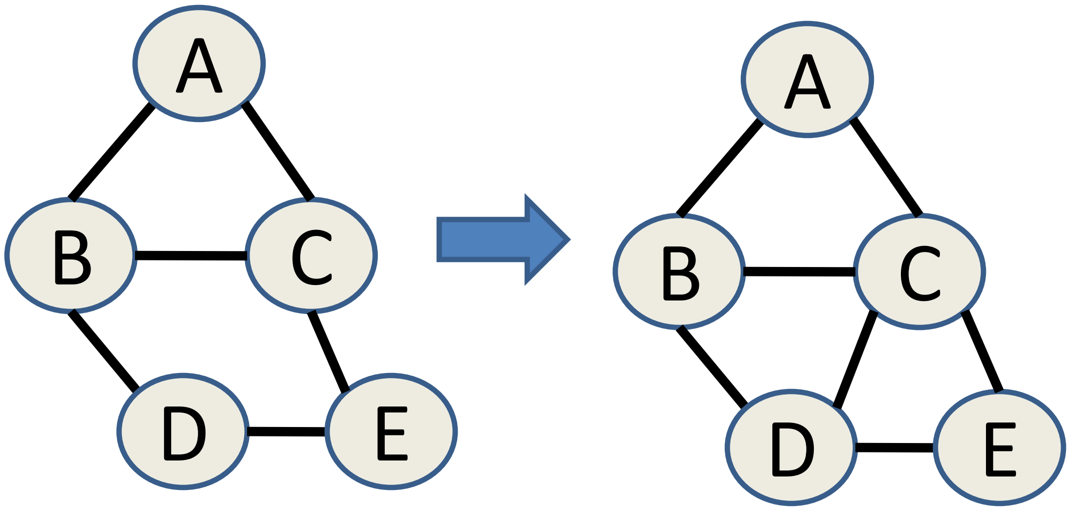
- Details follow

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

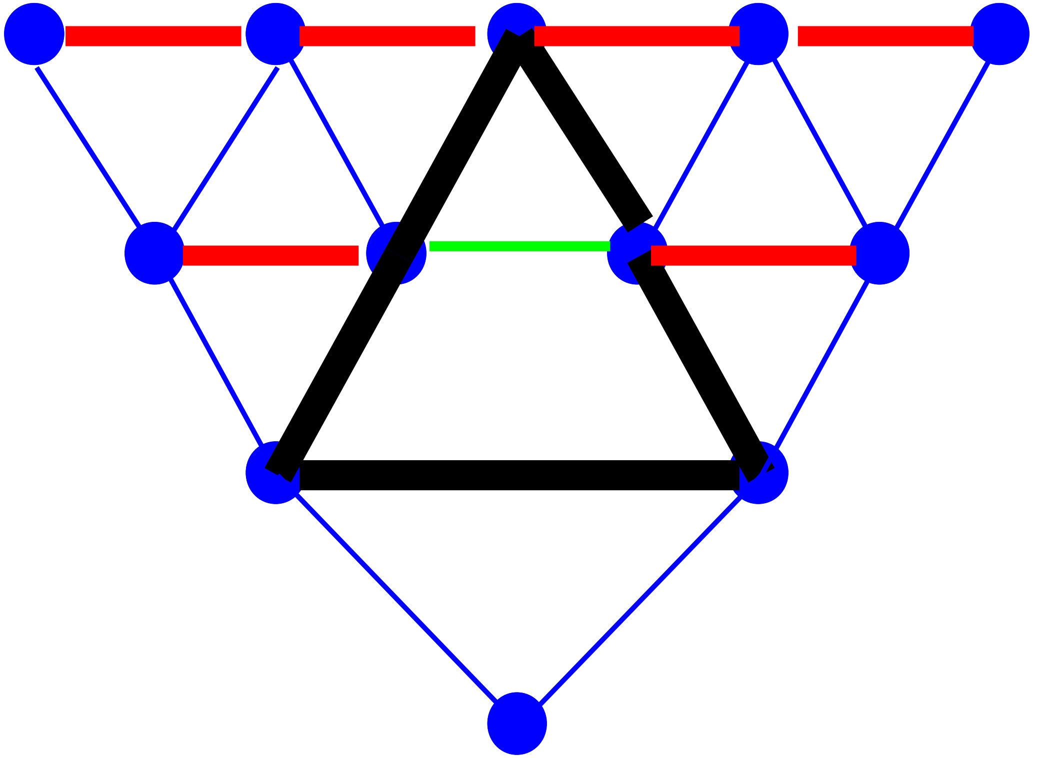
- Goal: Every cycle of length > 3 has a chord



- Why? Stay tuned.

Triangulation Algorithm

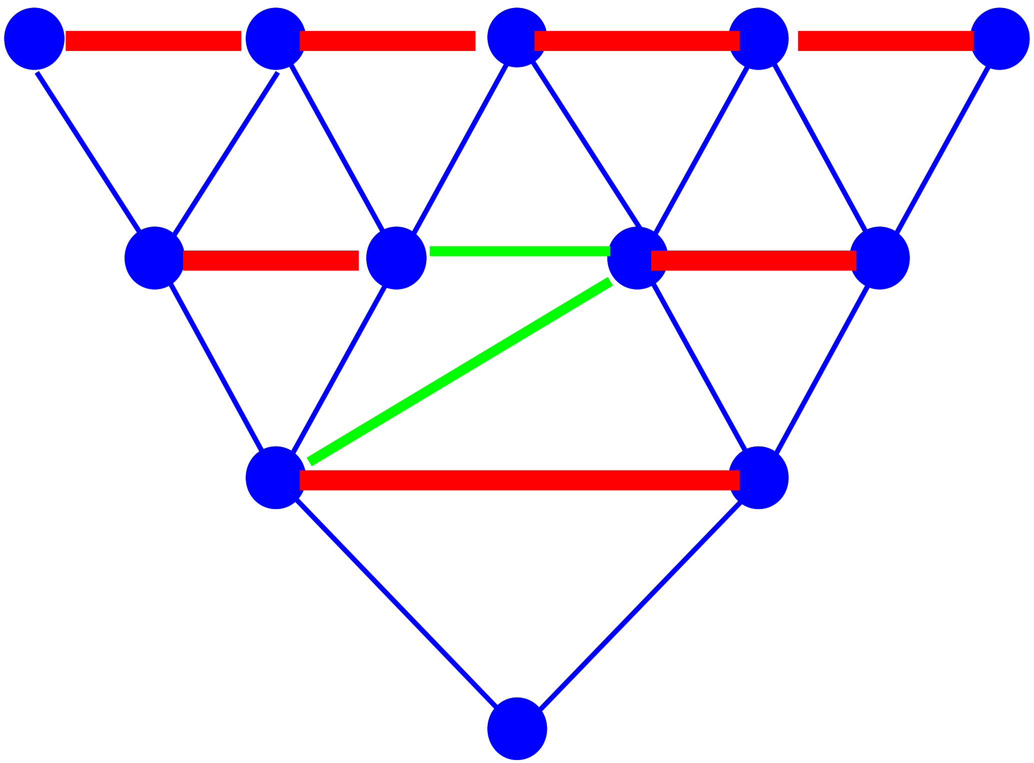
Repeat while there exists a cycle of length > 3 with no chord:
Add a chord (edge between two non-adjacent vertices in such a cycle).



- 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*

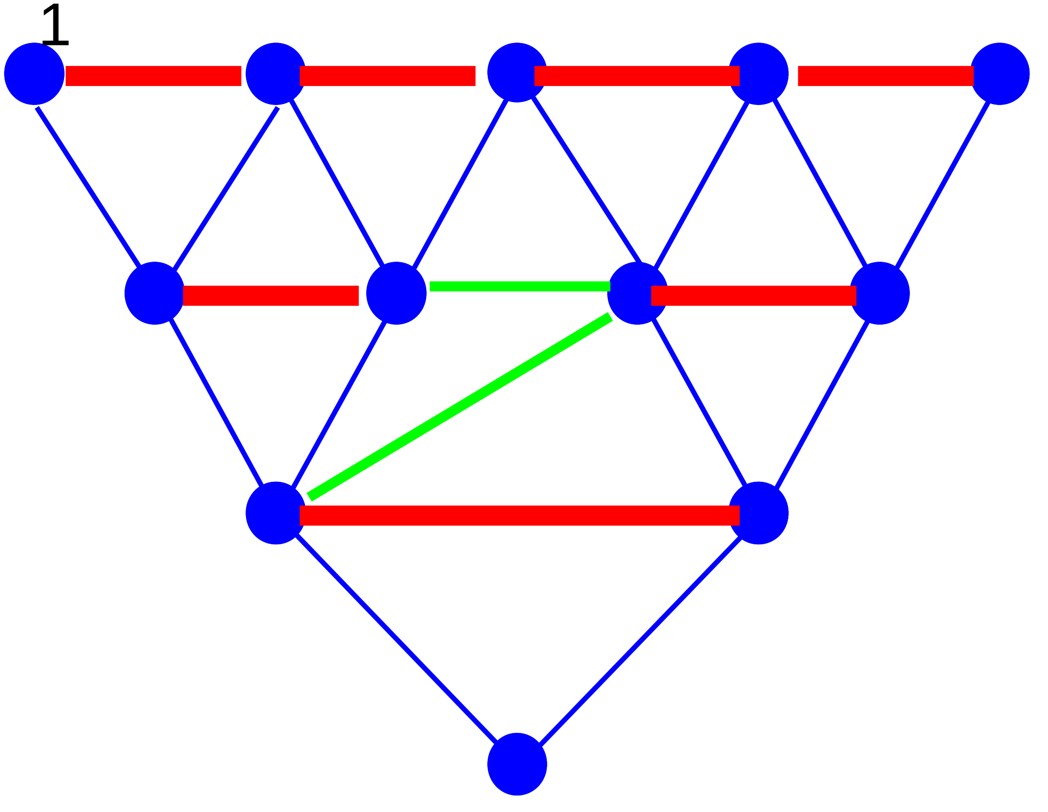
    if *i* has two non-adjacent labeled neighbors

        return false

return true

Triangulation Checking (3 of 3)

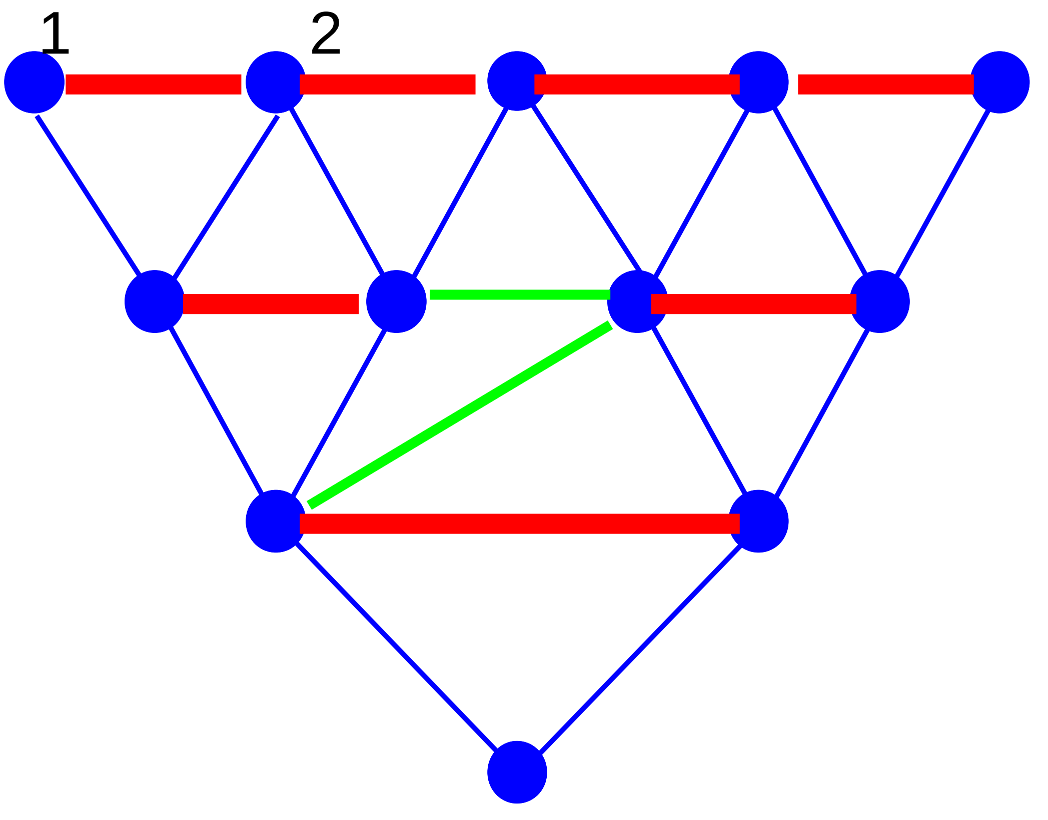
It appears to be triangulated, but how can we be sure?



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

Triangulation Checking (3 of 3)

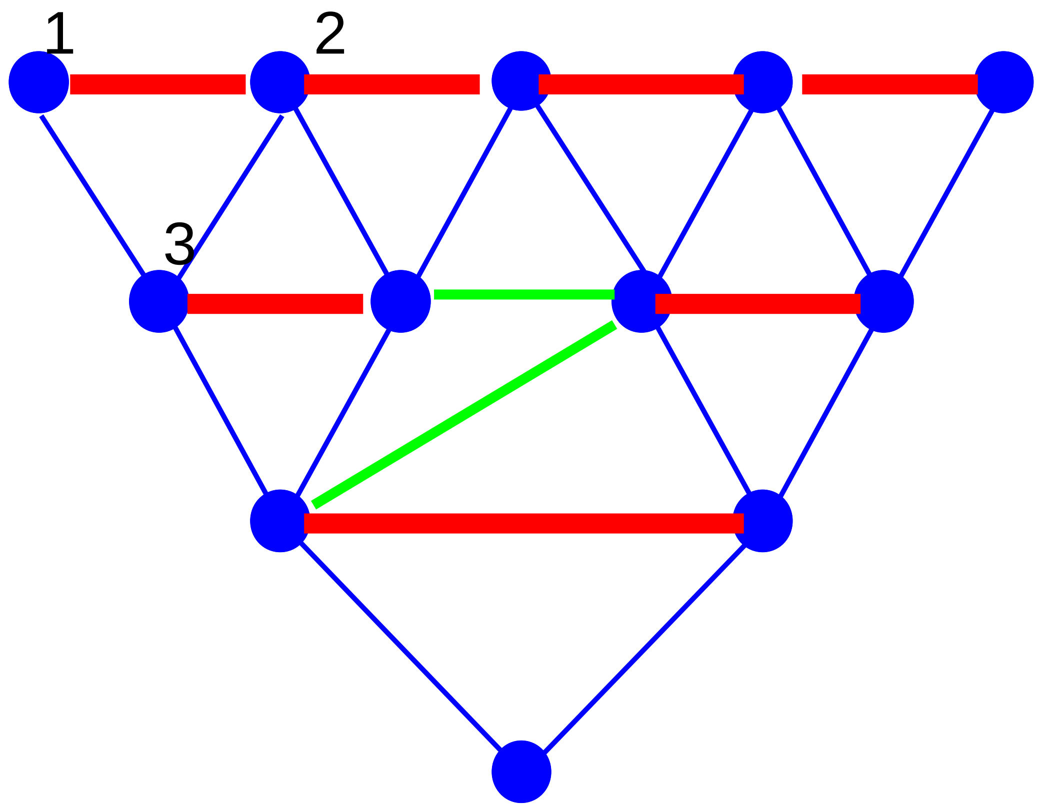
It appears to be triangulated, but how can we be sure?



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

Triangulation Checking (3 of 3)

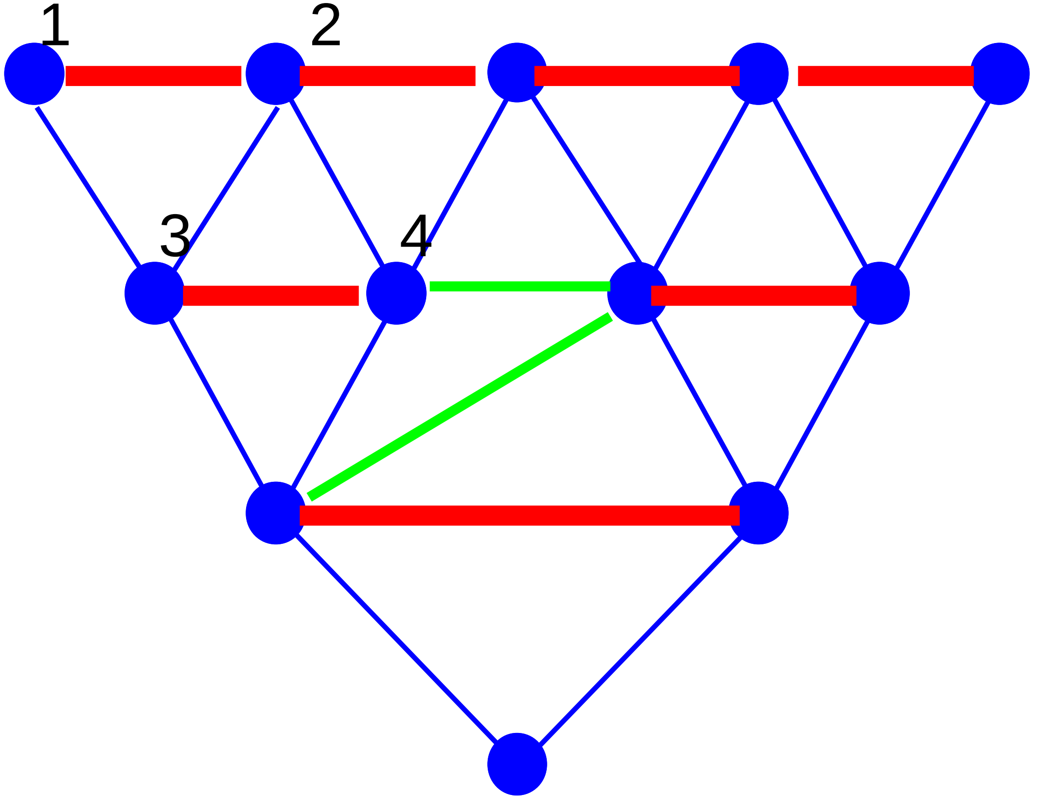
It appears to be triangulated, but how can we be sure?



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

Triangulation Checking (3 of 3)

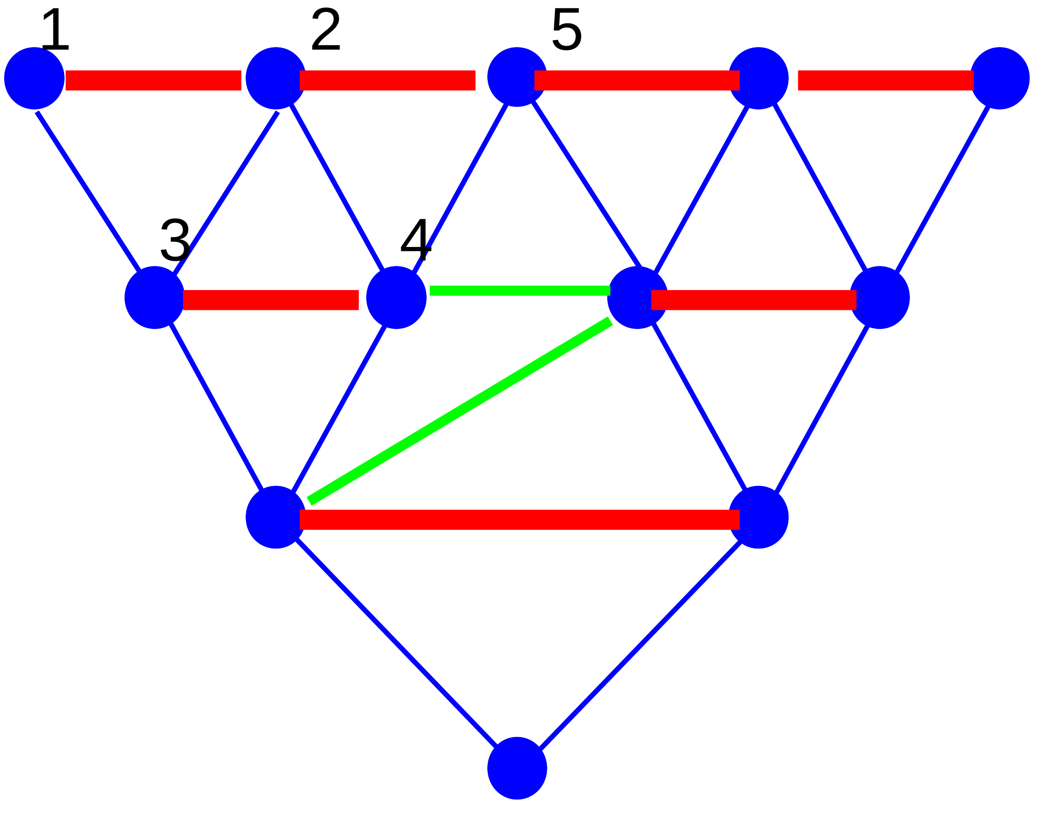
It appears to be triangulated, but how can we be sure?



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

Triangulation Checking (3 of 3)

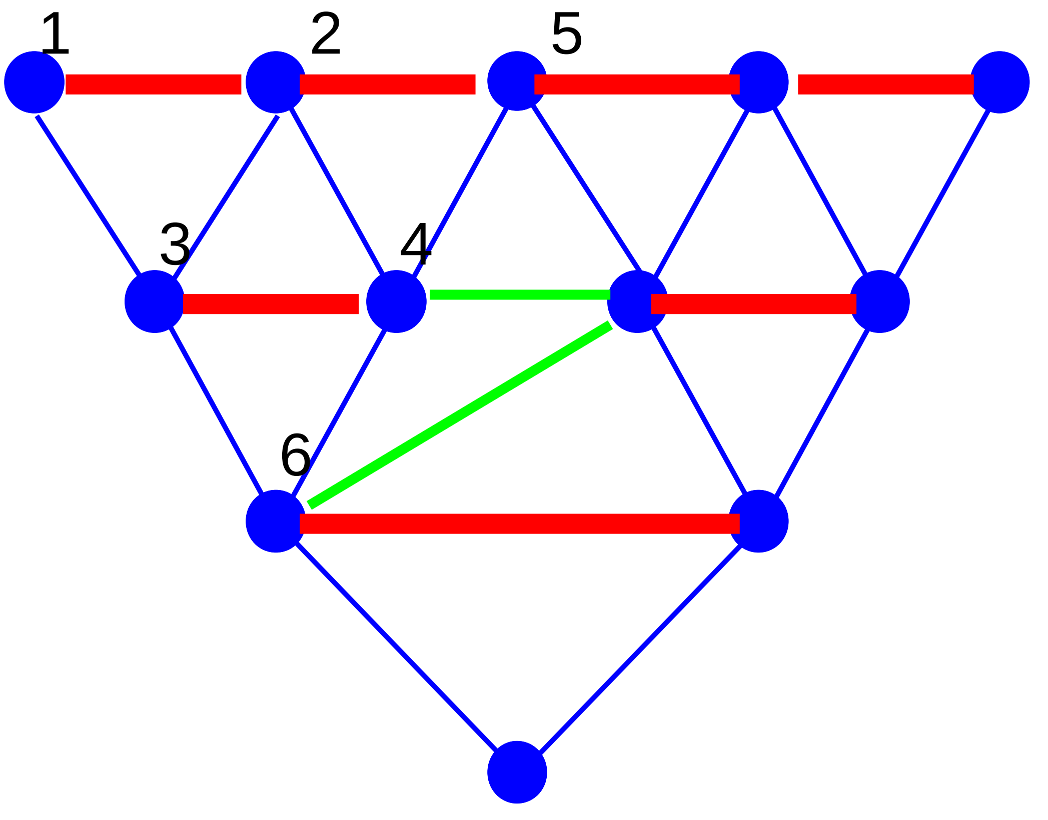
It appears to be triangulated, but how can we be sure?



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

Triangulation Checking (3 of 3)

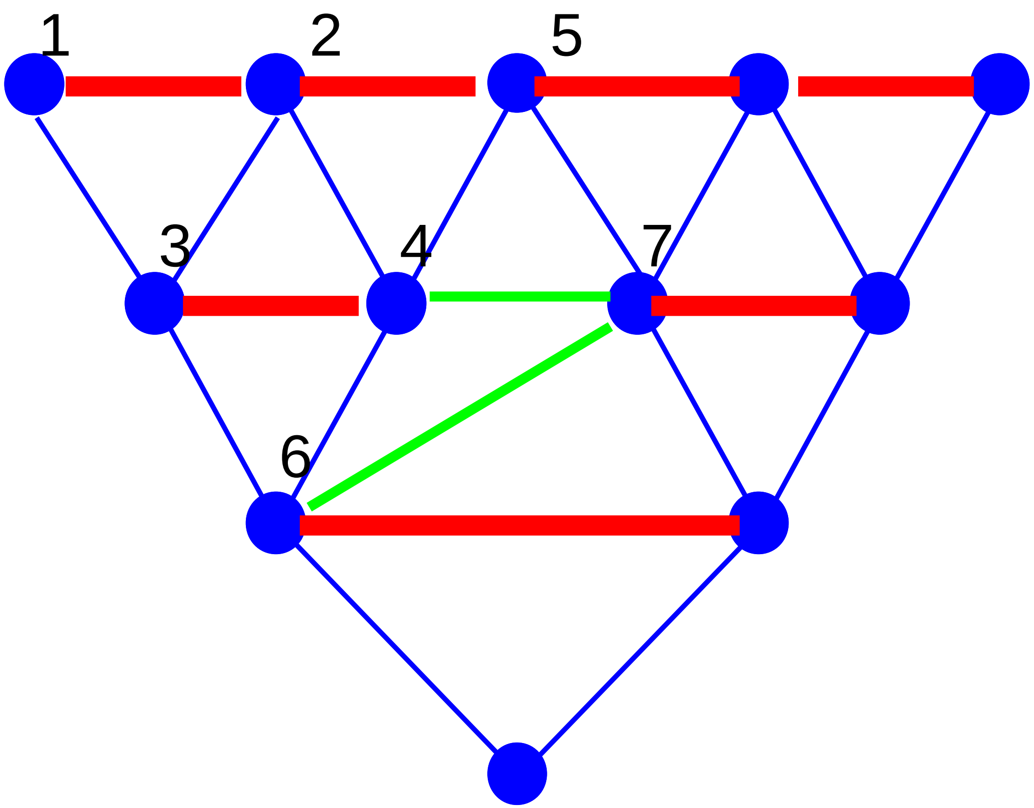
It appears to be triangulated, but how can we be sure?



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

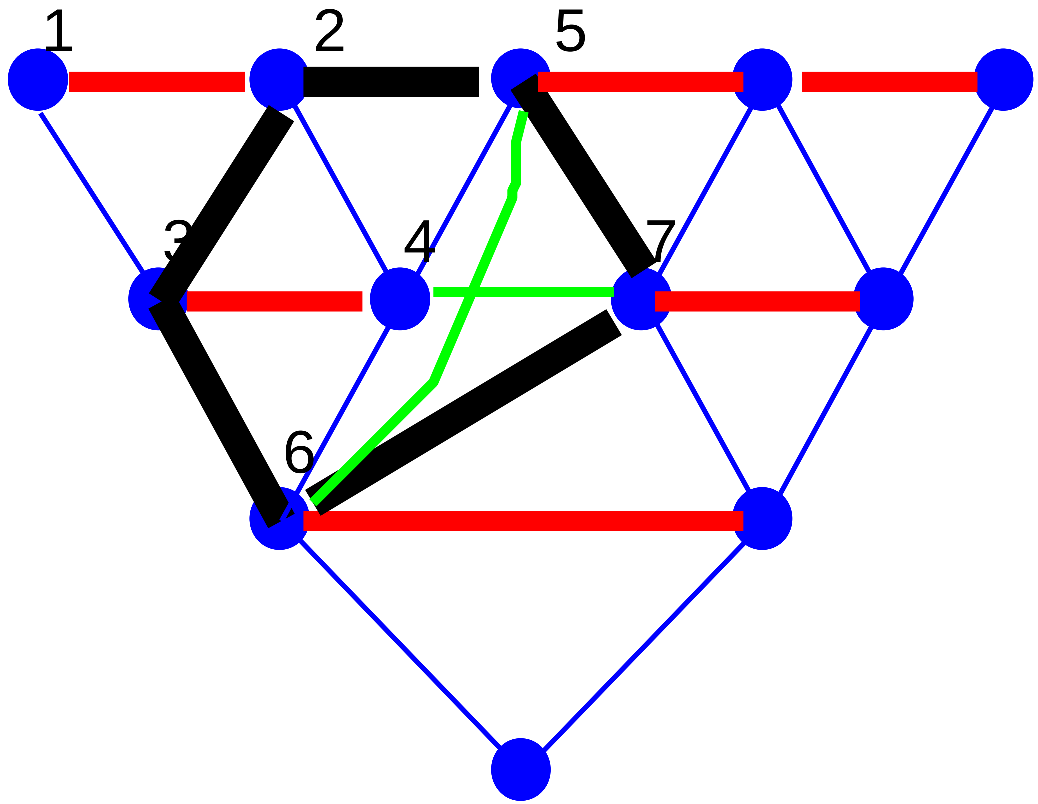
Triangulation Checking (3 of 3)

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



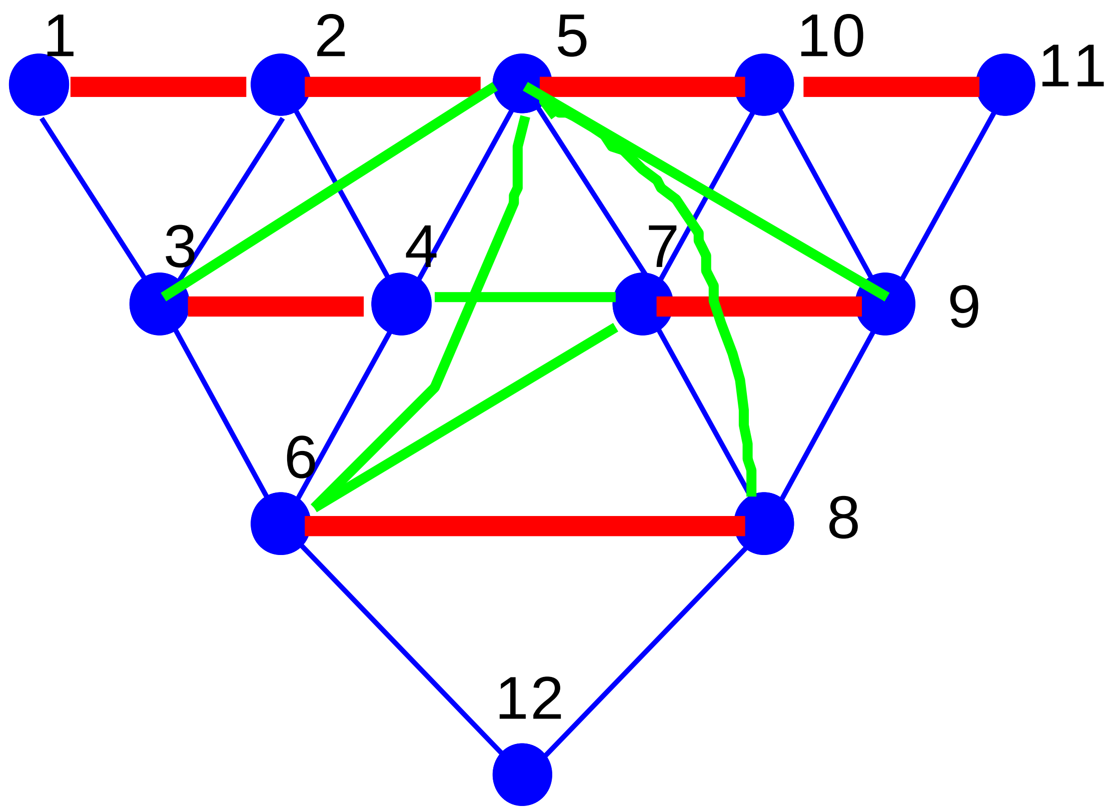
- From David Page, UWisc, pages.cs.**wisc**.edu/~dpage/cs731/lecture5.ppt

Connect the two offending nodes



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

Repeat Until Triangulation Check Succeeds



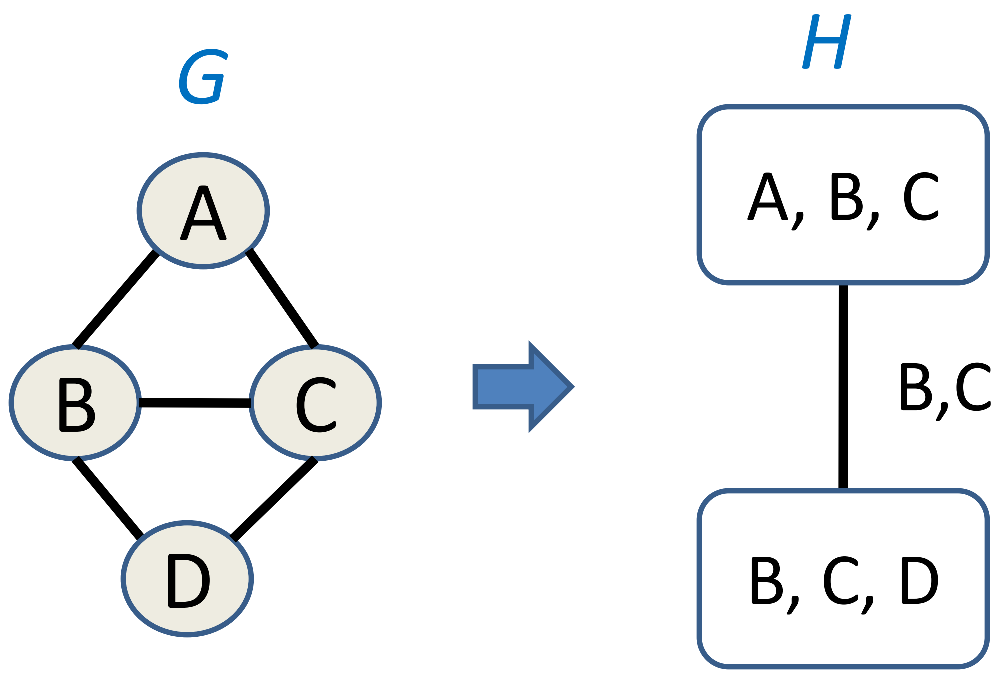
- From David Page, UWisc, pages.cs.**wisc**.edu/~dpage/cs731/lecture5.ppt

Junction Tree Outline

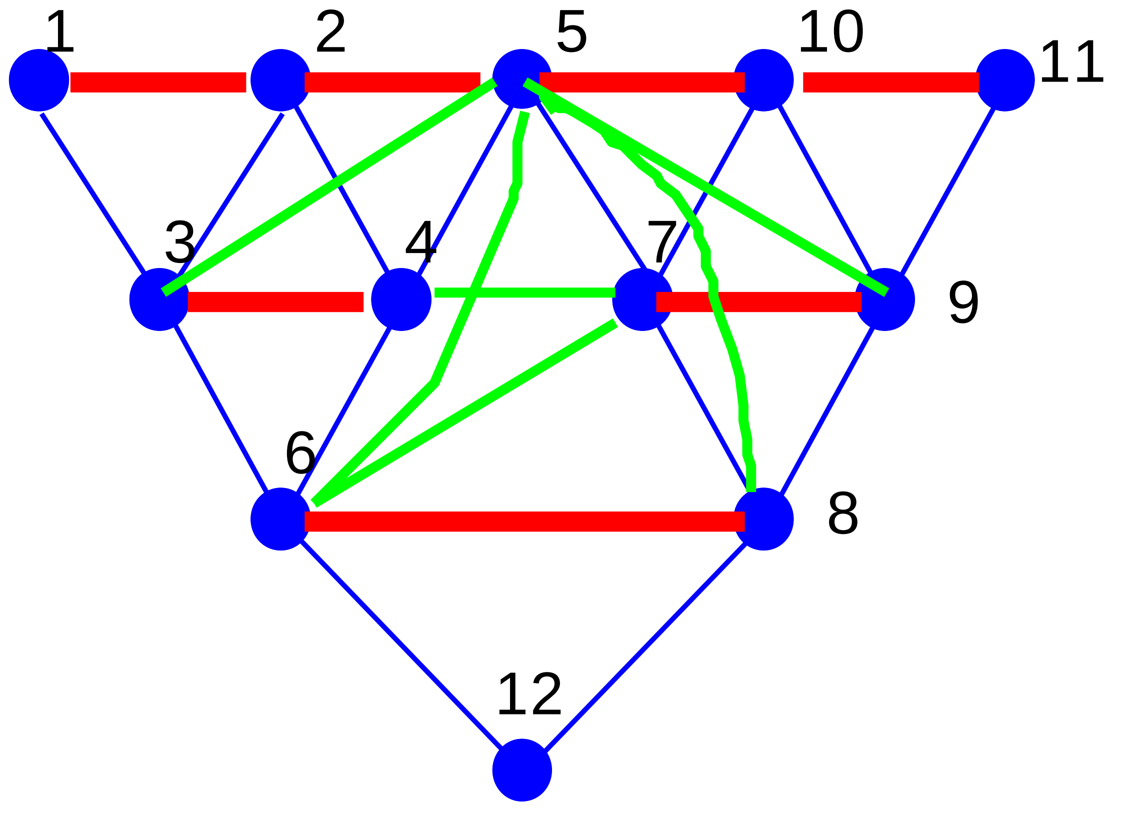
- If Bayes Net, convert to Markov Net
- Convert Markov Net into Junction Tree
  - Triangulate
  - **Build Clique Graph**
  - Build Junction Tree
- Do Inference using Junction Tree

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

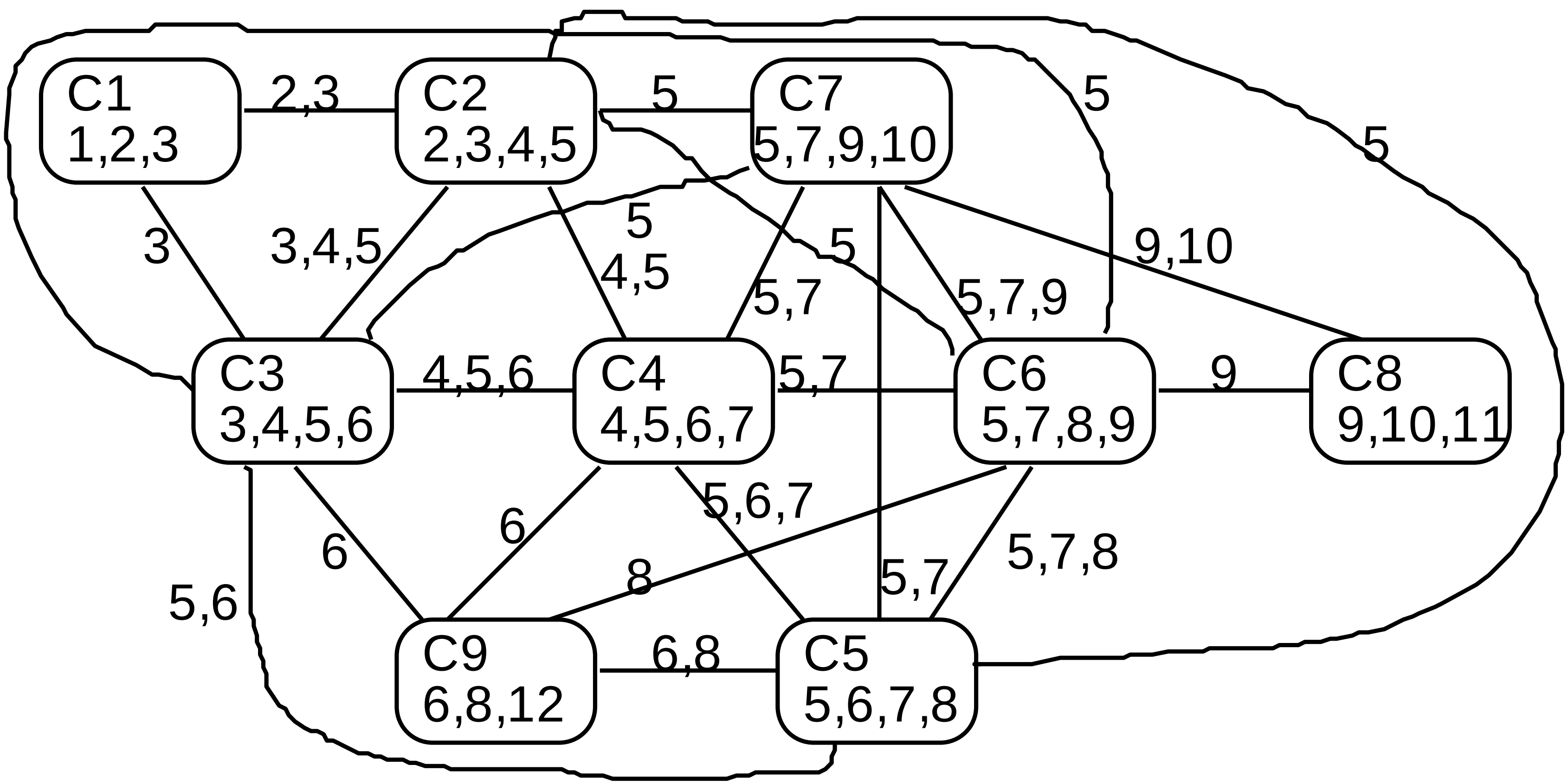


Bigger Example



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

Bigger Example – Clique Graph



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

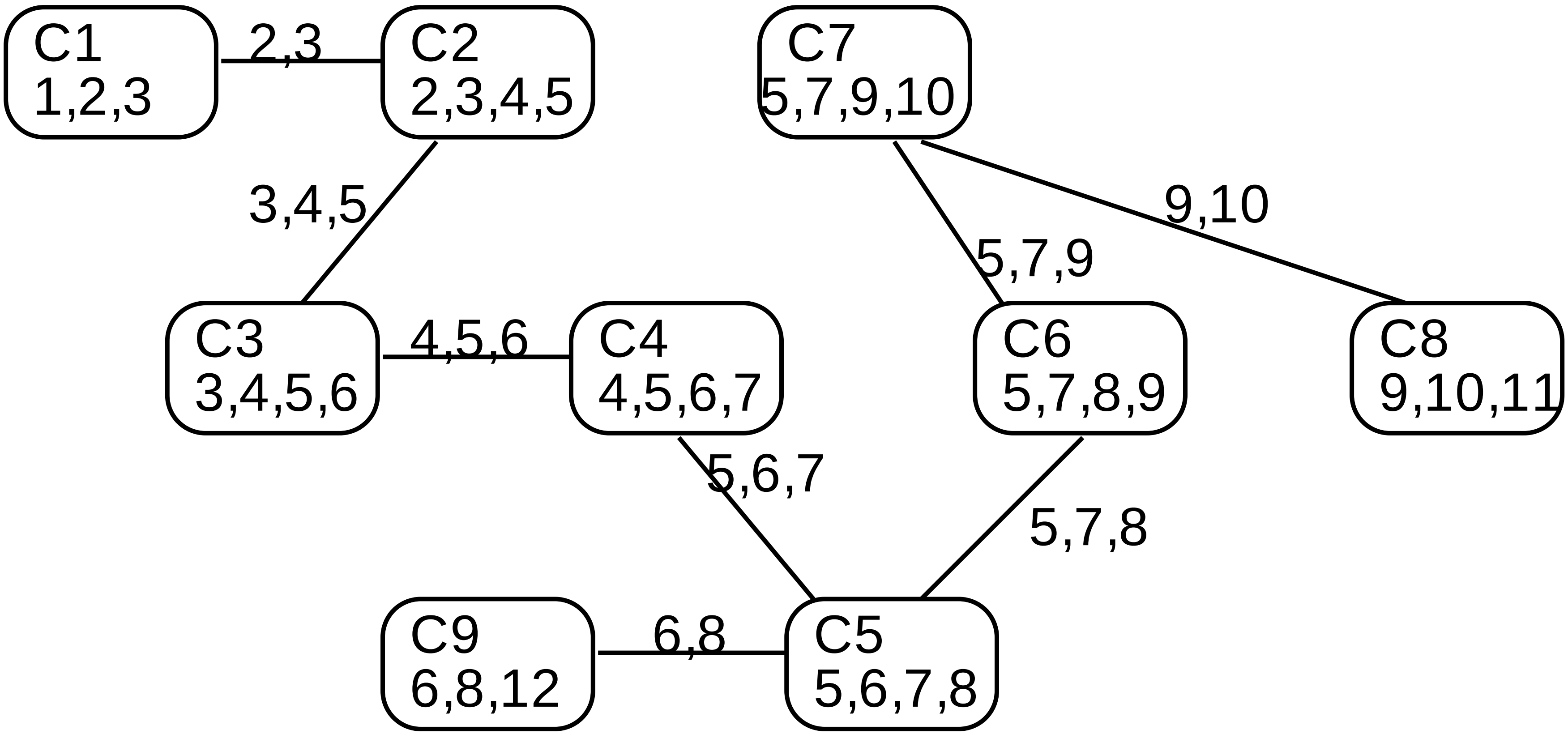
Junction Tree Outline

- 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

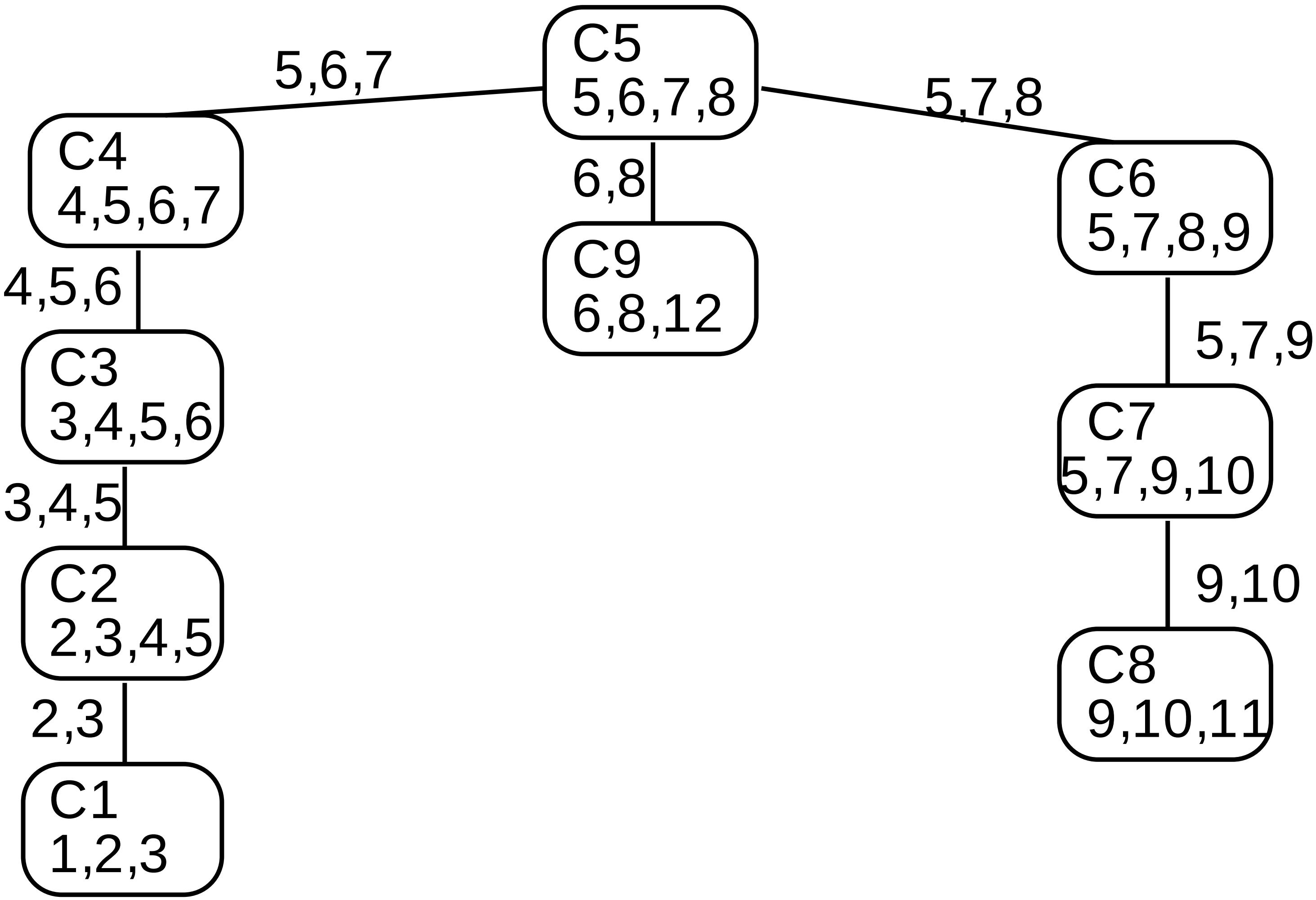
- A **Junction Tree** is a subgraph of the clique graph that
  - Is a tree
  - Contains all the nodes of the clique graph
  - Satisfies the **junction tree property**
    - For each pair of cliques *U*, *V* with intersection *S*, all cliques on path between *U* and *V* contain *S*

Junction Tree Example



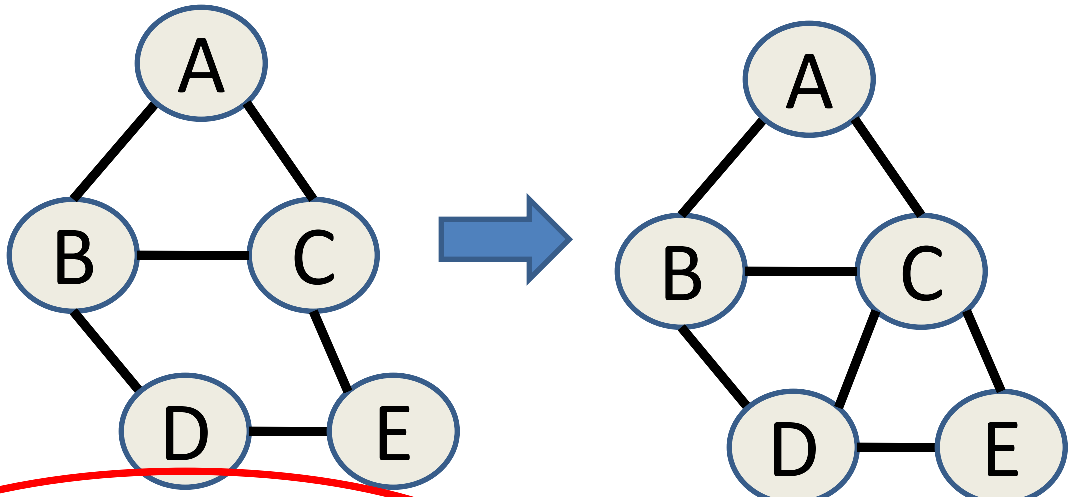
- From David Page, UWisc, pages.cs.**wisc**.edu/~dpage/cs731/lecture5.ppt

Choose a Root



Remember This?

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



- Why? Stay tuned.

Can we always find a Junction Tree?

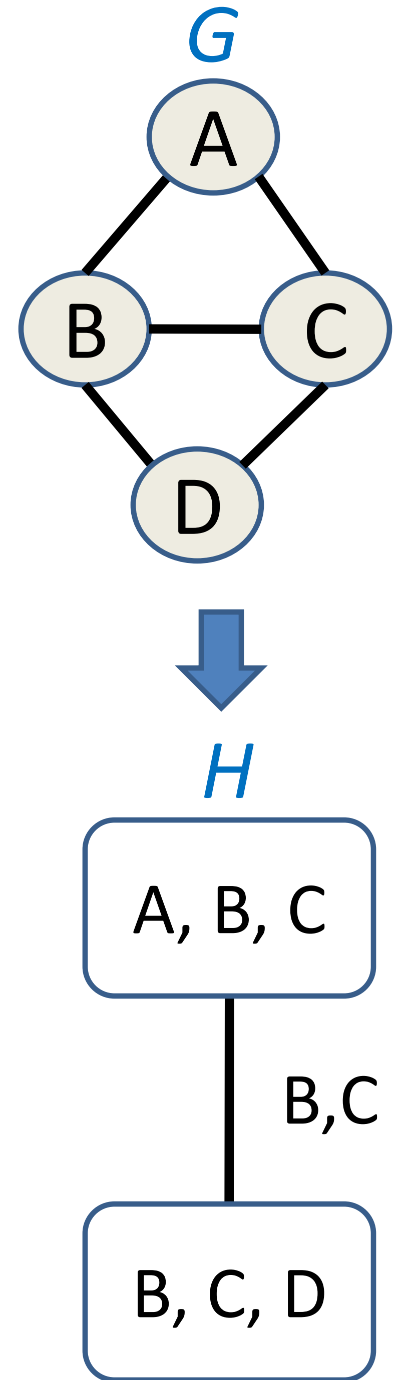
- Yes, for clique graphs of triangulated graphs
- Define “edge weight” on the clique graph to be the size of the intersection
  - Then a maximum-weight spanning tree is a junction tree
    [Jensen & Jensen, 1994]

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)

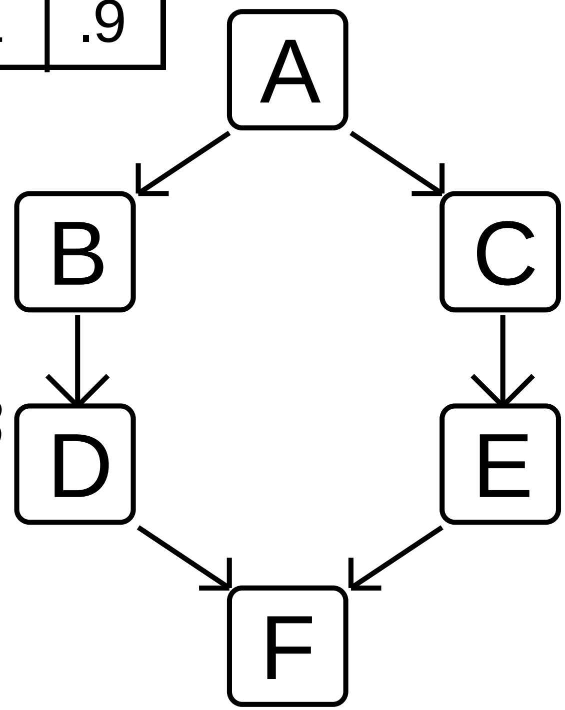


Example

	a	$\neg a$
	.1	.9

	a	$\neg a$
$b \mid A$	.1	.9
$\neg b \mid A$	.9	.1

	$d \mid B$	$\neg d \mid B$
b	.4	.6
$\neg b$	.7	.3



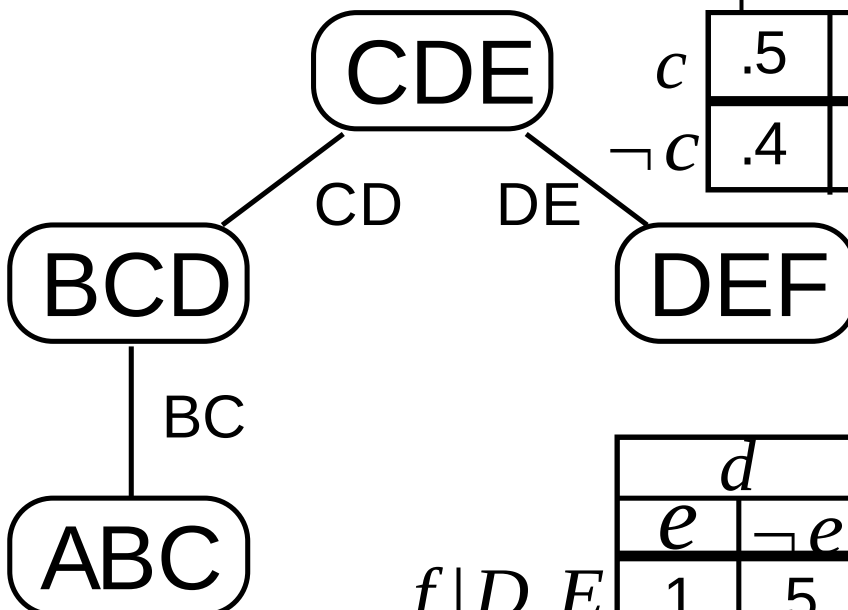
	a	$\neg a$
$c \mid A$	.7	.2
$\neg c \mid A$	.3	.8

	$e \mid C$	$\neg e \mid C$
c	.5	.5
$\neg c$	.4	.6

	d		$\neg d$	
	e	$\neg e$	e	$\neg e$
$f \mid D, E$	.1	.5	.4	.8
$\neg f \mid D, E$	.9	.5	.6	.2

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

Junction Tree with CPTs



$\mathbf{P}(E|C)$

	$e|C$	$\neg e|C$
c	.5	.5
$\neg c$	.4	.6

$\mathbf{P}(D|B)$

	$d|B$	$\neg d|B$
b	.4	.6
$\neg b$	.7	.3

$\mathbf{P}(A,B,C)$

	b		$\neg b$	
	c	$\neg c$	c	$\neg c$
a	.007	.003	.063	.027
$\neg a$	.162	.648	.018	.072

$\mathbf{P}(F|D,E)$

	d		$\neg d$	
	e	$\neg e$	e	$\neg e$
$f|D,E$	.1	.5	.4	.8
$\neg f|D,E$	.9	.5	.6	.2

- 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)

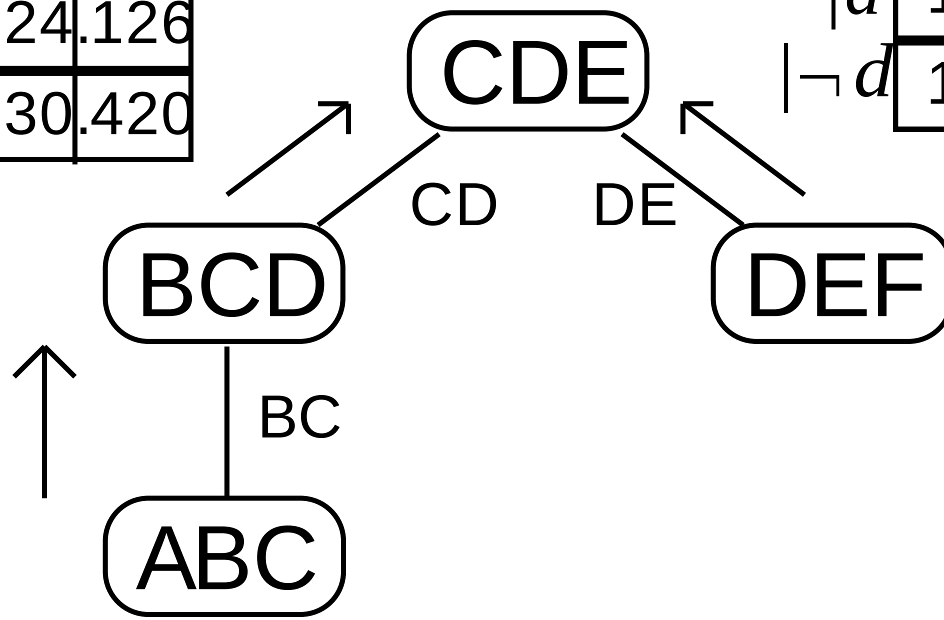
Upward Pass – assume no evidence

$\mathbf{P}(C, D)$

	d	$\neg d$
c	.124	.126
$\neg c$	.330	.420

$\mathbf{P}(B, C)$

	c	$\neg c$
b	.169	.651
$\neg b$	.081	.099



$\mathbf{P}(|\mathrm{D}, \mathrm{E})$

	$\|e$	$\|\neg e$
$\|d$	1.0	1.0
$\|\neg d$	1.0	1.0

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

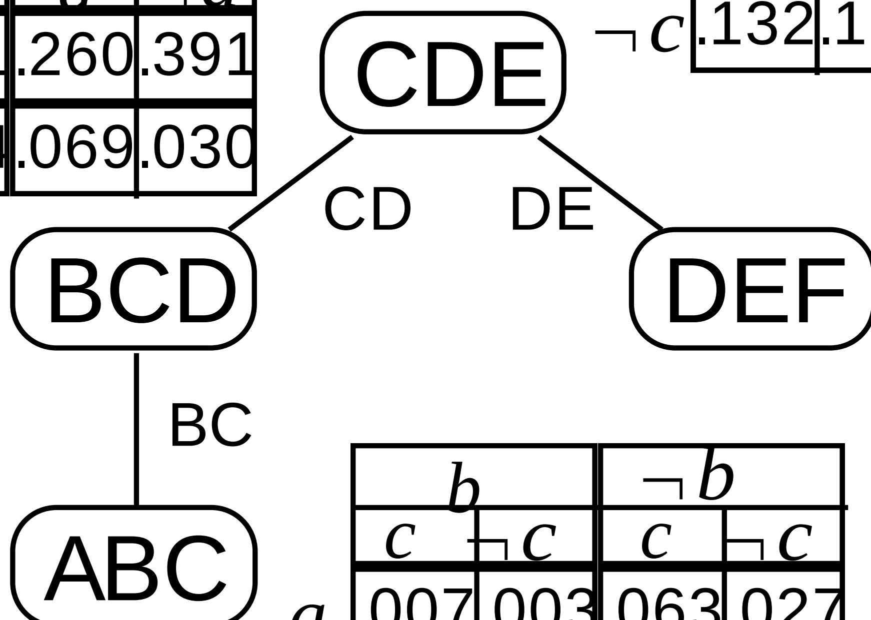
Status After Upward Pass

$\mathbf{P}(B,C,D)$

	c		$\neg c$	
	d	$\neg d$	d	$\neg d$
b	.068	.101	.260	.391
	.057	.024	.069	.030

	d		$\neg d$	
	e	$\neg e$	e	$\neg e$
c	.062	.062	.063	.063
$\neg c$	.132	.198	.168	.252

$\mathbf{P}(C,D,E)$



	d		$\neg d$	
	e	$\neg e$	e	$\neg e$
$f \mid$	.1	.5	.4	.8
$\neg f \mid$	.9	.5	.6	.2

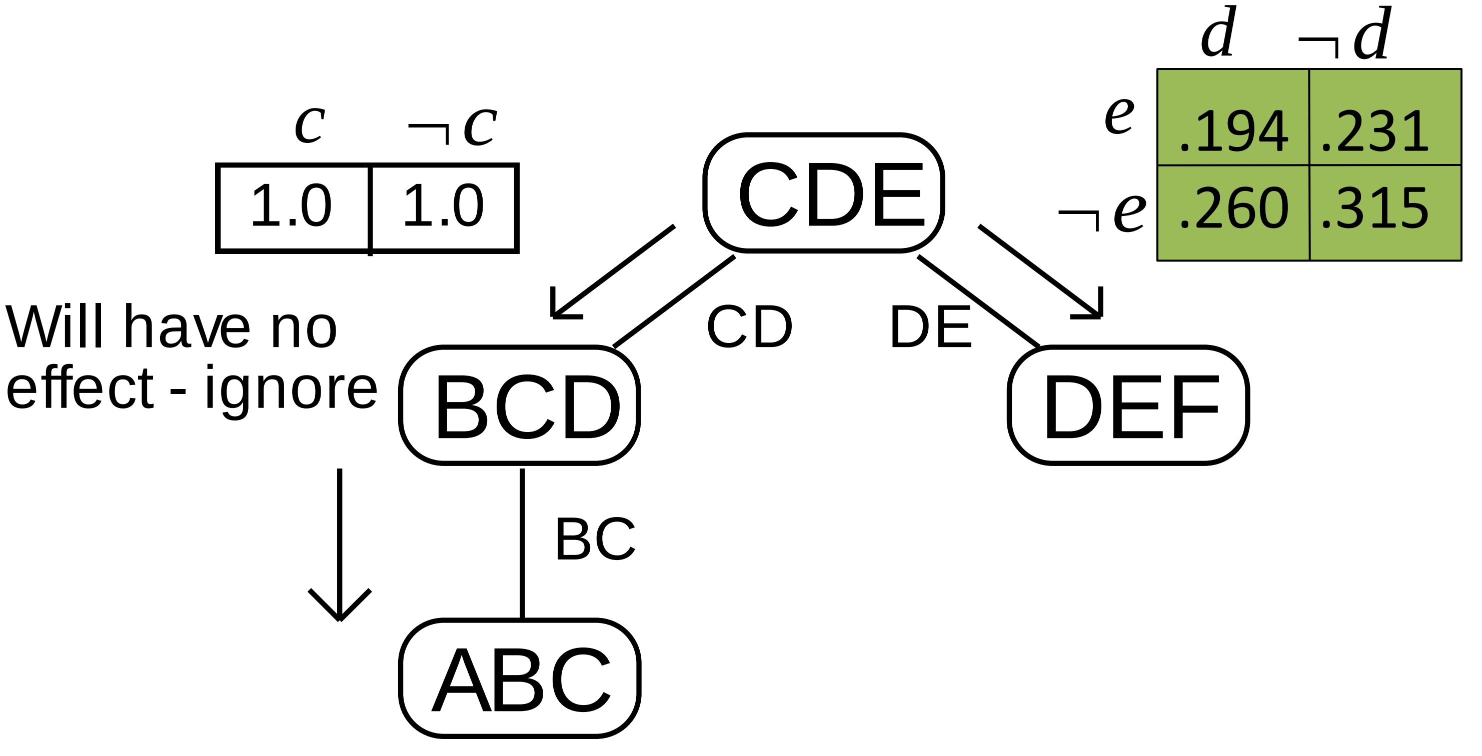
$\mathbf{P}(F|D,E)$

	b		$\neg b$	
	c	$\neg c$	c	$\neg c$
a	.007	.003	.063	.027
$\neg a$	.162	.648	.018	.072

$\mathbf{P}(A,B,C)$

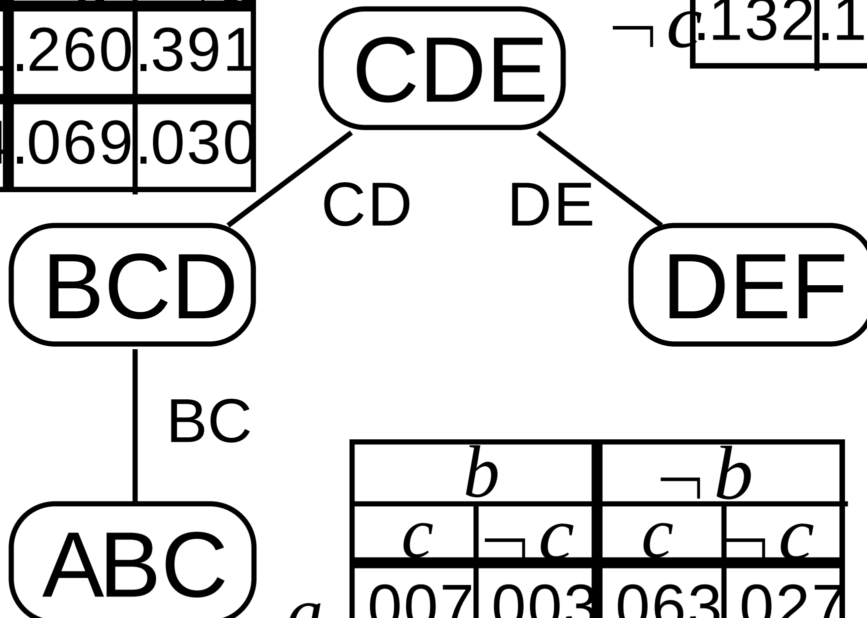
- From David Page, UWisc, pages.cs.**wisc**.edu/~dpage/cs731/lecture5.ppt

Downward Pass



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

Status After Downward Pass



$\mathbf{P}(B,C,D)$

	c		$\neg c$	
	d	$\neg d$	d	$\neg d$
b	.068	.101	.260	.391
$\neg b$	.057	.024	.069	.030

$\mathbf{P}(C,D,E)$

	d		$\neg d$	
	e	$\neg e$	e	$\neg e$
c	.062	.062	.063	.063
$\neg c$	.132	.198	.168	.252

$\mathbf{P}(D,E,F)$

	d		$\neg d$	
	e	$\neg e$	e	$\neg e$
f	.019	.130	.092	.252
$\neg f$	.175	.130	.139	.063

$\mathbf{P}(A,B,C)$

	b		$\neg b$	
	c	$\neg c$	c	$\neg c$
a	.007	.003	.063	.027
$\neg a$	.162	.648	.018	.072

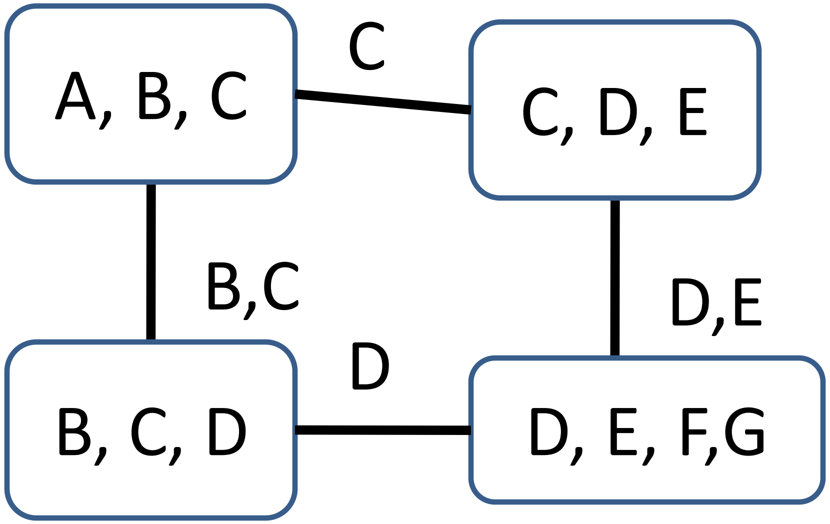
- From David Page, UWisc, pages.cs.**wisc**.edu/~dpage/cs731/lecture5.ppt

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)