Triangulated Graphs

- G triangulated $\Leftrightarrow$ G has the triangulated graph property.

- triangulated $\equiv$ chordal $\equiv$ perfect elimination

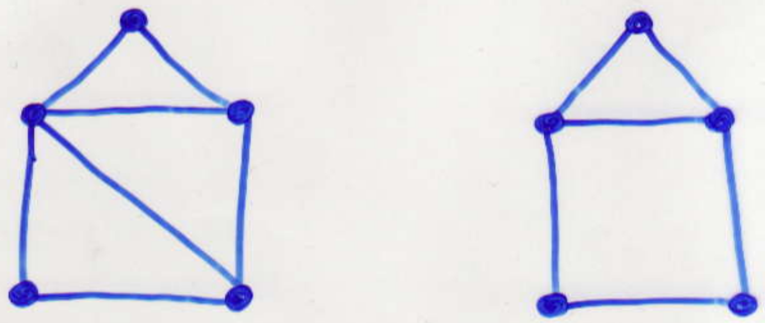
- Based on a theorem of Dirac on chordal graph, Fulkerson and Gross (also Rose) gave useful algorithmic characterization.

- Dirac showed that every chordal graph has a simplicial node, a node all of whose neighbors form a clique.

- It follows easily from the triangulated property that deleting nodes of a chordal graph yields another chordal graph.

- This observation leads to the following Algorithm:
  - find a simplicial node
  - delete it
  - recurse on the resulting graph, until no node remain.

- If G contains C_k, $k \geq 4$, no node in that cycle will ever become simplicial.



- The procedure provides us with all the maximal cliques.

- Let $C(v_i) = \{v_i\} \cup$ v_i's higher neighbors

- Then, $C(v_i)$ = clique

- It is easy to see that the set of cliques $C(v_i)$, $1 \le i \le n$, include all the maximal cliques.

- Note that some cliques $C(v_i)$ are not maximal.

- Theorem. There are at most n maximal cliques in a chordal graph.

max cliques $\underline{\underline{5}}$

$\underline{\underline{n=6}}$

- The node-ordering provided by the procedure has many other uses.

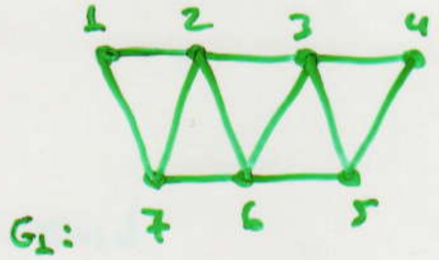
- Rose establishes a connection between chordal graphs and symmetric linear systems.

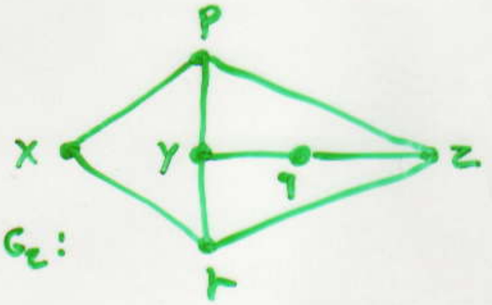
- node-ordering : perfect elimination ordering
  perfect elimination scheme

- Let $\sigma = [v_1, v_2, \ldots, v_n]$ be an ordering of the vertices of a graph $G=(V,E)$.

  - σ = peo if each v_i is a <u>simplicial</u> <u>node</u> to graph $G[\{v_i, v_{i+1}, \ldots, v_n\}]$

  - that is, $X_i = \{ v_j \in adj(v_i) \mid i<j \}$ is complete.

- 



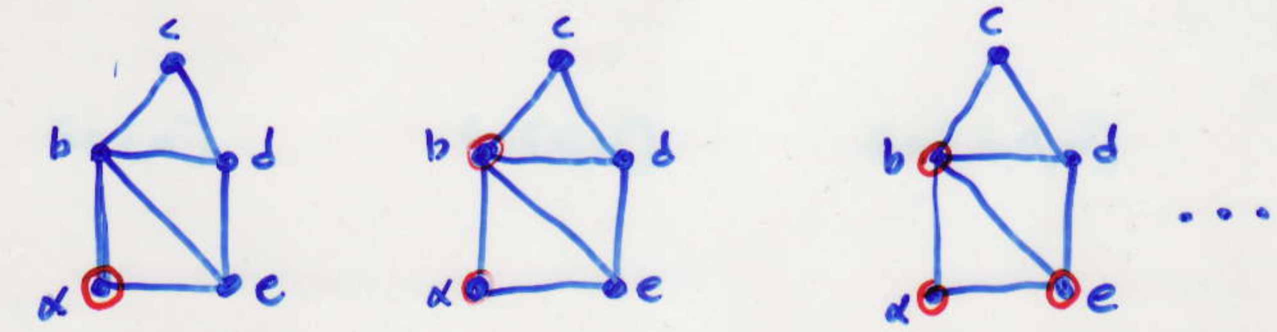
$\sigma = [1, 7, 2, 6, 3, 5, 4]$

no simplicial vertex

- G_1 has 96 different peo.

- Algorithms for computing peo.

```
Algorithm LexBFS
begin
    forall v ∈ V do label(v) := ();
    for i := |V| downto 1 do
        • choose v ∈ V with lex max label(v);
        • σ(i) ← v;
        • forall u ∈ V ∩ N(v) do
              label(u) ← label(u) + i
        • V := V \ {v};
    end
end
```

- 

$\sigma = [a]$
label(b) = (5)
label(e) = (5)

$\sigma = [b, a]$
label(c) = (4)
label(d) = (4)
label(e) = (54)

$\sigma = [e, b, a]$
label(d) = (43)
label(c) = (4)

Algorithm MCS

begin

1. for $i := |V|$ downto 1 do

2. • choose $v \in V$ with a max number of numbered neighbours;

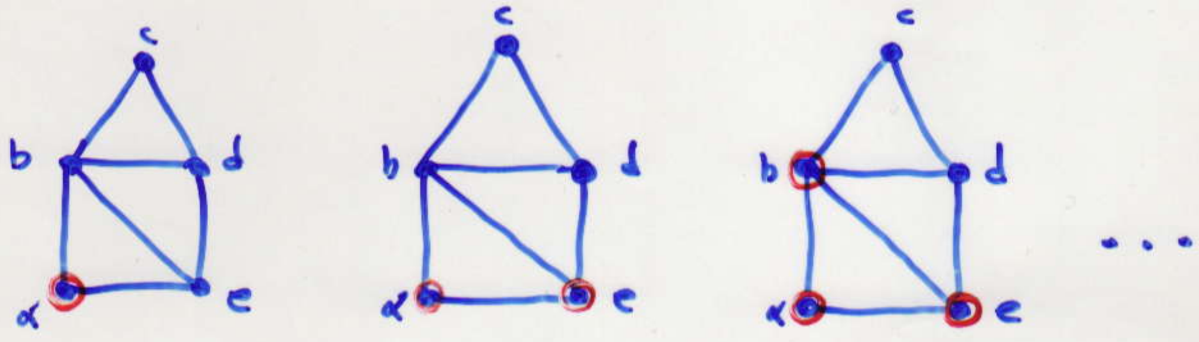
3. • number v by i;

4. • $\sigma(i) \leftarrow v$;

5. • $V := V \setminus \{v\}$;

end

•



$\sigma = [\alpha]$ $\sigma = [e, \alpha]$ $\sigma = [b, e, \alpha]$

Complexity: $O(1 + \text{degree}(v))$

$O(n+m)$.

- Testing a peo → Naive Algorithm

$O(n+m)$

Algorithm PERFECT

begin

1. for all vertices u do $A(u) \leftarrow \emptyset$;
2. for $i \leftarrow 1$ to $n-1$ do
3. $\quad v \leftarrow \sigma(i)$;
4. $\quad X \leftarrow \{x \in adj(v) \mid \sigma^{-1}(v) < \sigma^{-1}(x)\}$;
5. $\quad$ if $X = \emptyset$ then goto 8
6. $\quad u \leftarrow \sigma(\min\{\sigma^{-1}(x) \mid x \in X\})$;
7. $\quad X - \{u\} \,\|\, A(u) \quad (X - \{u\} \equiv X_u)$
8. $\quad$ if $\underline{A(v) - adj(v) \neq \emptyset}$ then ret("false")

$\quad$ end

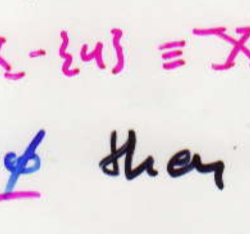
9. ret("true")

end

$A(v) \subseteq adj(v) \Rightarrow$ true.

$A(u) = \{u, x, y\}$

$A(u) - adj(u)$ ff



- $v = a \quad X = \{e, b\}$

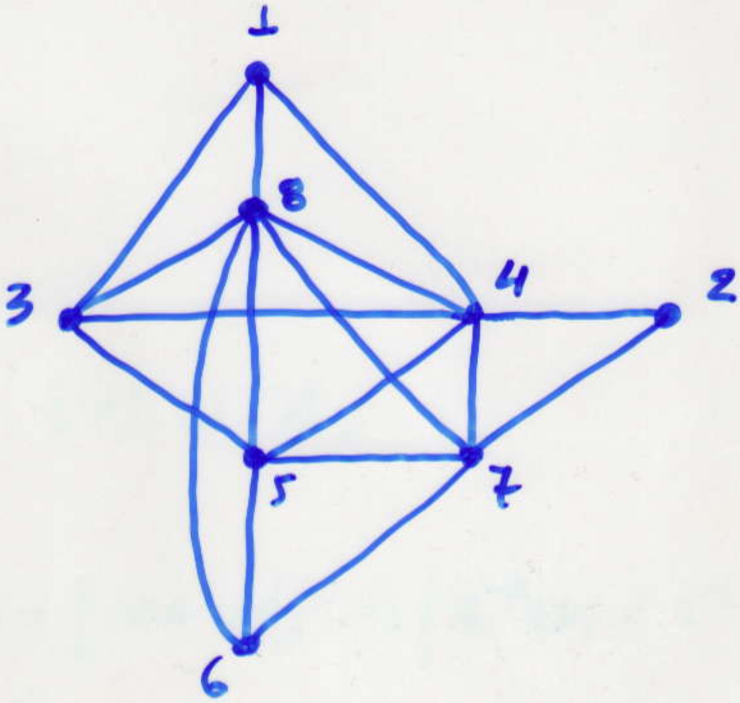
$u = e \quad A(e) = \{b\} \quad A(a) - adj(a) = \emptyset$

$\sigma = [a, e, b, d, c]$

- $v = e \quad X = \{b, d\}$

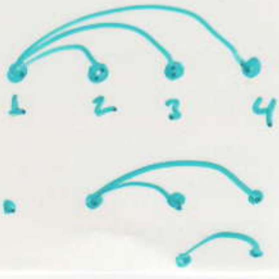
$u = b \quad A(b) = \{d\} \quad A(e) - adj(e) = \emptyset$

Example:



$G = [1, 2, 3, 4, 5, 6, 7, 8]$

$v = 1$:	$X = 3, 4, 8$	$u = 3$	$A(3) = 4, 8$	
$v = 2$:	$X = 4, 7$	$u = 4$	$A(4) = 7$	$A(2) \subseteq adj(2)$
$v = 3$:	$X = 4, 5, 8$	$u = 4$	$A(4) = 7, 5, 8$	$A(3) \subseteq adj(3)$
$v = 4$:	$X = 5, 7, 8$	$u = 5$	$A(5) = 7, 8$	$A(4) \subseteq adj(4)$
$v = 5$:	$X = 6, 7, 8$	$u = 6$	$A(6) = 7, 8$	$A(5) \subseteq adj(5)$
$v = 6$:	$X = 7, 8$	$u = 7$	$A(7) = 8$	$A(6) \subseteq adj(6)$



Ελέγχω $A(1) - adj(1) = \emptyset$

» $A(2) - adj(2) = \emptyset$

» $A(3) - adj(3) = \emptyset$

25-1.

- Correctness of the Algorithm

- **Theorem**: The algorithm PERFECT returns TRUE iff σ is a p.e.o.

Proof:

($\Leftarrow$) Suppose the algorithm returns FALSE on interation number $\sigma^{-1}(u)$.

This may happen only if in stage 8:
$A(u) - adj(u) \neq \emptyset \Rightarrow w \in A(u)$ ק' $w \notin adj(u)$

The vertex w was added to $A(u)$ at stage 7 of a prior interation, number $\sigma^{-1}(v)$.

thus, $\sigma^{-1}(v) < \sigma^{-1}(u) < \sigma^{-1}(w)$ for $u, w \in adj(v)$

But, since $uw \notin E \Rightarrow \sigma$ is not p.e.o.

($\Rightarrow$) Suppose σ is not a p.e.o and the Algorithm returns TRUE.

Let v be a vertex with max index in σ, such that $X_v = \{ w \mid w \in adj(v) \text{ and } \sigma^{-1}(v) < \sigma^{-1}(w) \}$ does not induce a clique.

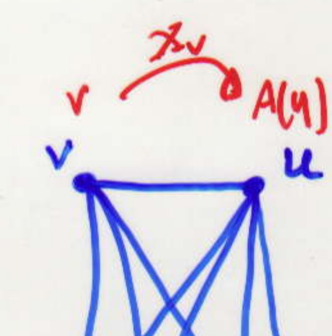
1st neighbors

$$\sigma = [\ldots.\ \ v, \ldots, u, \ldots.]$$

max

Let u be the vertex defined in stage 6 of the iteration $\sigma^{-1}(v)$.

In stage 7 of this iteration, $X_v \setminus \{u\}$ is added to $A(u)$.



Since the Algorithm returns TRUE in stage 8 $\Rightarrow$

$A(u) \subseteq adj(u) \Rightarrow$

$X_v \setminus \{u\} \subseteq adj(u)$

every $x \in X_v \setminus \{u\}$ is a neighbor of u.

- Since $\sigma^{-1}(v)$ is chosen to be the max index in σ for which X_v is not a clique

and v is prior to u $\Rightarrow$ X_u is a clique

Thus, all u's neighbors in $X_v \setminus \{u\}$ are adjacent to each other $\Rightarrow$

X_v is a clique, a contradiction.

Complexity: $O(n+m)$

• χ and maximal cliques

(a) • Every maximal clique of a chordal graph $G=(V,E)$ is of the form

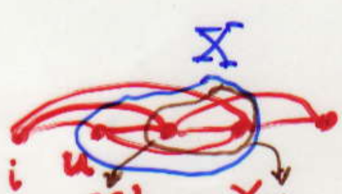
$$\{v\} \cup X_v$$

where

$$X_v = \{ x \in adj(v) \mid \sigma^{-1}(v) < \sigma^{-1}(x) \}$$

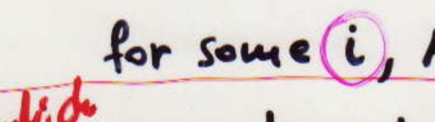
(b) • Proposition (Fulkerson and Gross 1965): A chordal graph on n nodes has at most n maximal cliques, with equality if and only if the graph has no edges.

(c) • Algorithm χ-and-maxclique

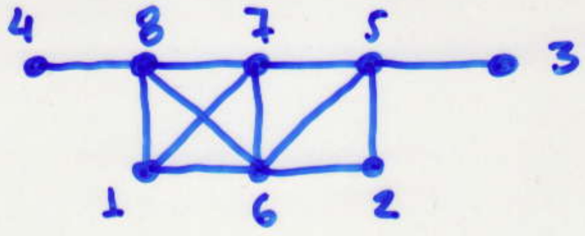


• Some of "$\{u\} \cup X_u$" will not be max clique.

• $\{u\} \cup X_u$ is not maximal clique iff for some i, Algorithm PERFECT, X_u is the set which is concatenated to $A(u)$. $\equiv X_i - \{u\}$



• Example (a):



$\{1\} \cup X_1 = \{1,6,7,8\}$ $\qquad$ $\sigma = [1,2,3,4,5,6,7,8]$

$\{2\} \cup X_2 = \{2,5,6\}$

$\{3\} \cup X_3 = \{3,5\}$ $\qquad$ $\{6\} \cup X_6 = \{6,7,8\}$

$\{4\} \cup X_4 = \{4,8\}$ $\qquad$ $\{7\} \cup X_7 = \{7,8\}$

$\{5\} \cup X_5 = \{5,6,7\}$ $\qquad$ $\{8\} \cup X_8 = \{8\}$

maximal $\qquad$ not maximal

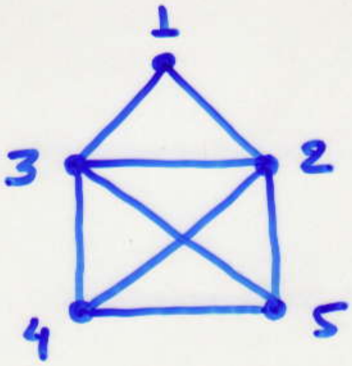
• Example (c):

$\{6\} \cup X_6 = \{6,7,8\}$ $\qquad$ $X_6 = \{7,8\}$

$i=1 \Rightarrow v=1$

$u=6 \Rightarrow X_1 = \{6,7,8\} \Rightarrow X_1 - \{6\} = \{7,8\}$

- Example (c):



- $\{3\} \cup X_3 = \{3,4,5\}$

  $X_3 = \{4,5\}$

$\sigma = [1,2,3,4,5]$

$i=2 \Rightarrow v=2$

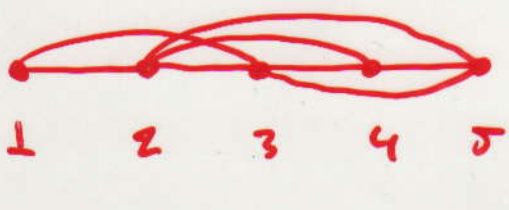
$u=3 \Rightarrow X_2 = \{3,4,5\} \Rightarrow X_2 - \{3\} = \{4,5\}$

- $\{2\} \cup X_2 = \{2,3,4,5\}$

  $X_2 = \{3,4,5\}$

$i=1 \Rightarrow v=1$

$u=2 \Rightarrow X_1 = \{2,3\} \Rightarrow X_1 - \{2\} = \{3\}$

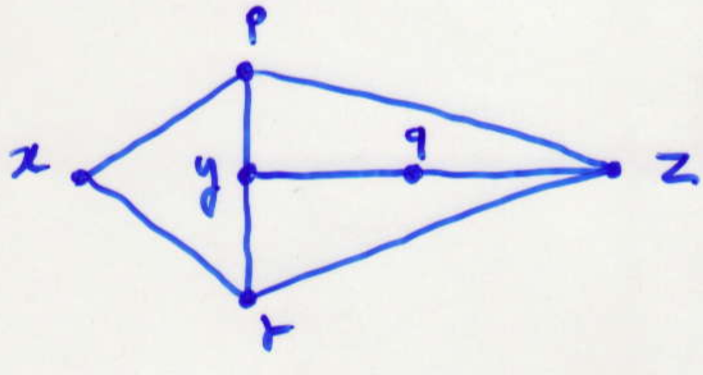


• Characterizing Triangulated Graphs

• **Definition**: A subset $S \subset V$ is called a vertex separator for nonadjacent vertices a, b (or a-b separator), if in G_{V-S} vertices a and b are in different connected components.

If no proper subset of S is an a-b separator, S is called minimal vertex separator.

• Example:



the set $\{y, z\}$ is a minimal vertex separator for p and q.

the set $\{x, y, z\}$ is a minimal p-r separator.

Theorem (Dirac 61, Fulkerson and Gross 65)

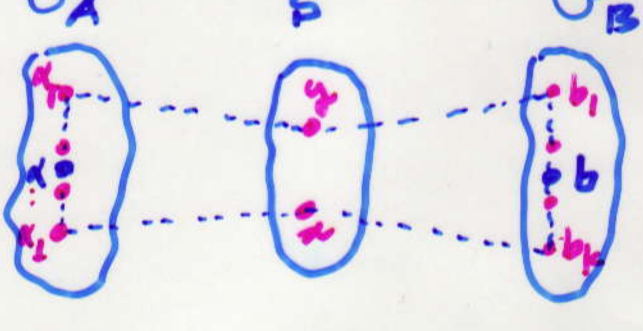
(1) G is triangulated.

(2) G has a p.e.o. Moreover, any simplicial vertex can start a perfect order.

(3) Every minimal vertex separator induces a complete subgraph of G.

Proof:

(1) $\Rightarrow$ (3):



- Let S be an a-b separator.
- We will denote G_A, G_B the connected comp. of G_{V-S} containing a, b respectively.

- Since S is minimal, every vertex $x \in S$ is a neighbor of a vertex in A and a vertex in B.

- For any $x, y \in S$, $\exists$ minimal paths $[x, a_1, \ldots, a_r, y]$ ($a_i \in A$) and $[x, b_k, \ldots, b_1, y]$ ($b_i \in B$)

- Since $[x, a_1, \dots, a_r, y, b_1, \dots, b_k, x]$ is a simple cycle of length $l \geq 4$, $\Rightarrow$ it contains a chord.

- For every i,j $a_i b_i \notin E$ (S is a-b separator) and also $a_i a_j \notin E$, $b_i b_j \notin E$ (by the minimality of the paths)

- Thus, $xy \in E$.

$(3) \Rightarrow (1)$. Suppose every minimal separator S is a clique.

Let $[v_1, v_2, \dots, v_k, v_1]$ be a chordless cycle.

- v_1 and v_3 are nonadjacent.
- $S_{1,3}$ contains v_2 and at least one of $v_4, v_5, \dots, v_k$.

But v_2, v_i $(i = 4, 5, \dots, k)$ are nonadjacent $\Rightarrow$ $S_{1,3}$ does not induce a clique.

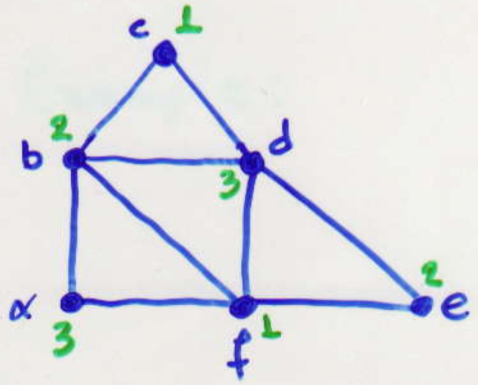
Coloring chordal Graphs

- Gavril gives a coloring algorithm, based on a greedy approach.

- We use positive integers as colors.

- Method:
  - start at the last node v_n of the peo;
  - work backwards, assign to each v_i in turn the minimum color not assigned to its higher neighbors.

- Example:



$$\sigma = [\alpha, c, b, d, e, f]$$

$$3 \ 1 \ 2 \ 3 \ 2 \ 1$$

$$\omega(G) = \chi(G)$$

Finding the stability number $\alpha(G)$

- Gavril gives the following solution.

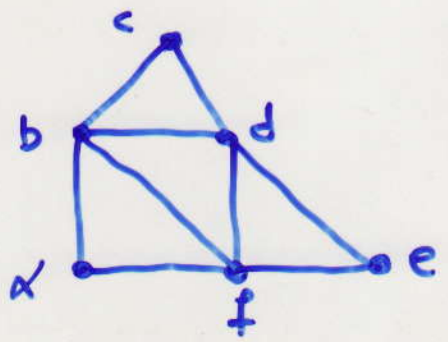
- Method:
  - Let σ be a peo of a chordal graph G.
  - Define inductively a sequence of vertices $y_1, y_2, \dots, y_t$ as follows:

    $y_1 = \sigma(1)$;

    y_i is the first vertex in σ which follows y_{i-1} and which is not in $X_{y_1} \cup X_{y_2} \cup \dots \cup X_{y_{i-1}}$;

    All vertices following y_t are in $X_{y_1} \cup X_{y_2} \cup \dots \cup X_{y_t}$.

- Example:



$\sigma = [a, c, b, d, e, f]$

$y_1 \; y_2 \; \bullet \; \bullet \; y_3 \; \bullet$

$X_{y_1} = \{b, f\}$

$X_{y_2} = \{b, d\}$

$X_{y_3} = \{f\}$

- **Theorem**: The set $\{y_1, y_2, \dots, y_t\}$ is a maximum stable set of G, and the collection of sets $Y_i = \{y_i\} \cup X_{y_i}$, $1 \le i \le t$, comprises a minimum clique cover of G.

Proof. The set $\{y_1, y_2, \dots, y_t\}$ is stable since if $y_j y_i \in E$ for $j < i$, then $y_i \in X_{y_j}$ which cannot be. Thus, $\alpha(G) \geq t$.

On the other hand, each of the sets $Y_i = \{y_i\} \cup X_{y_i}$ is a clique, and so $\{Y_1, \dots, Y_t\}$ is a clique cover of G. Thus, $\alpha(G) = k(G) = t$.

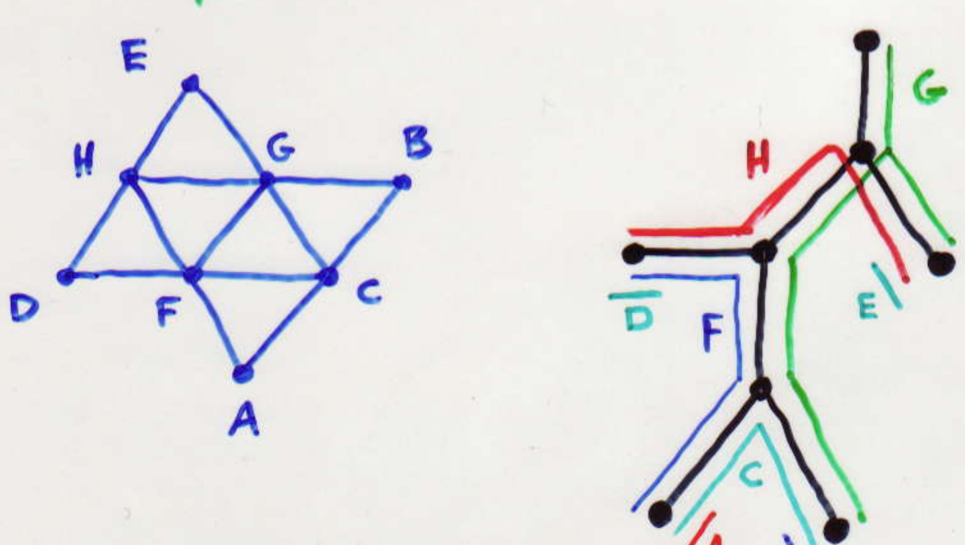
We have produced the desired maximum stable set and minimum clique cover.

A characterization of Chordal Graphs

- The chordal graphs are exactly the intersection graphs of subtrees of trees.

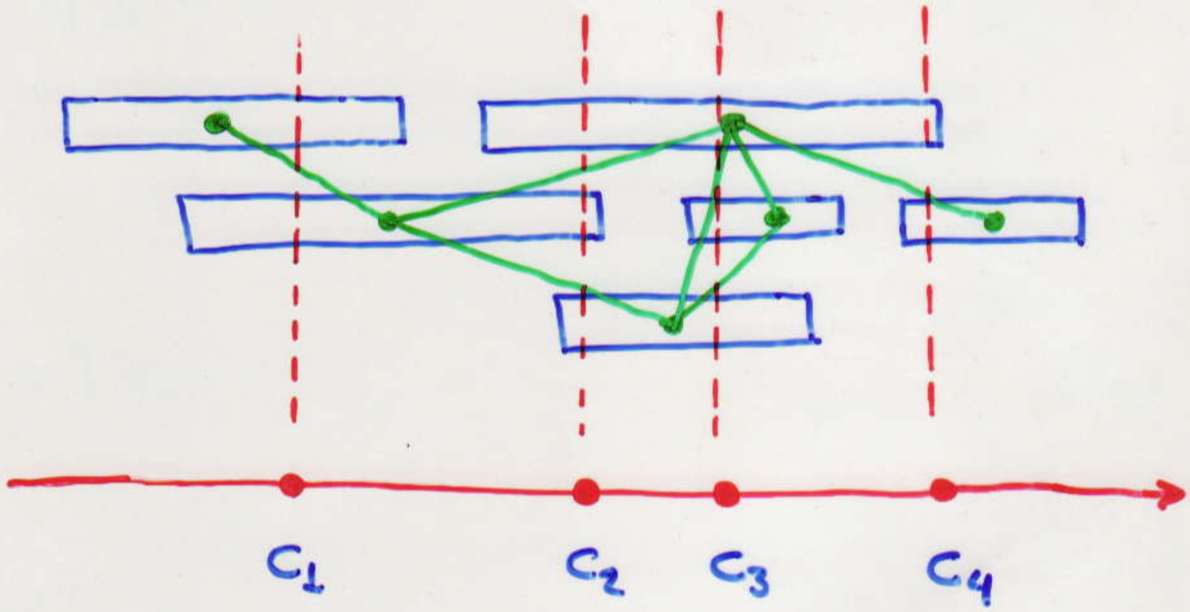
- That is, for a tree T and subtrees $T_1, T_2, \ldots, T_n$ of the tree T there is a graph whose nodes correspond to subtrees T_i, and where two nodes are adjacent if the corresponding subtrees share a node of T.

- Example:



- The graphs arising in this way are exactly the chordal graphs, and interval graphs arise when the tree T happens to be a simple path.

Interval Graphs



- **Theorem**: Let G be a graph. The following statments are equivalent.
  (i) G is an interval graph.
  (ii) G contains no C_4 and $\overline{G}$ is a comparability gr.
  (iii) The maximal cliques of G can be linearly ordered such that, for every vertex x of G the maximal cliques containing x occur consecutively.

Simple Elimination Scheme for Chordal Comparability Graphs

- **chordal**: if every cycle of length $\ell > 3$ has a chord.

- **comparability**: if we can assign directions to edges of G so that the resulting digraph G' is transitive.

- **simple elimination scheme**: a permutation $\sigma = (v_1, v_2, \dots, v_n)$ such that each v_i is simplicial in $G - \{v_1, \dots, v_{i-1}\}$ and the neighborhoods of the vertices adjacent to v_i in $G - \{v_1, \dots, v_{i-1}\}$ form a total order with respect to set containment.

- If G is a chordal comparability graph we will show that the following algorithm (Cardinality LexBFS) constructs a ses.

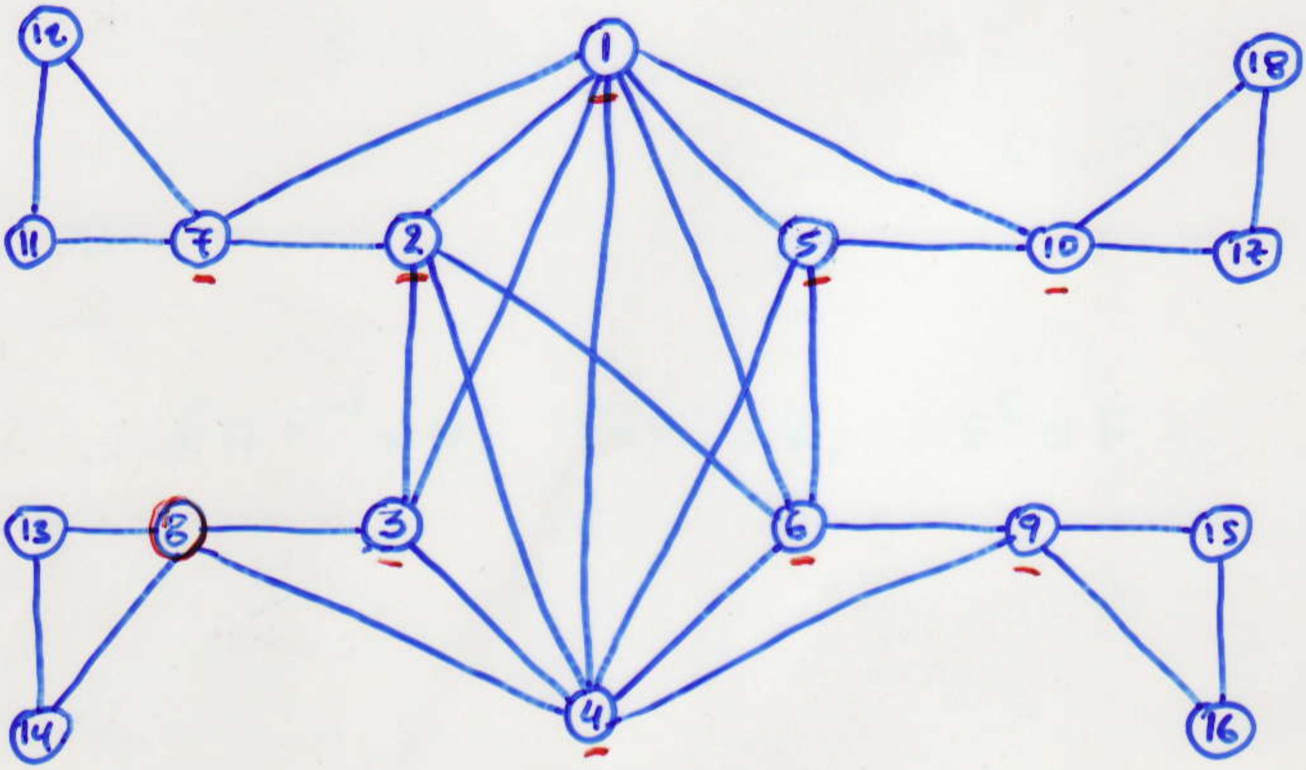
- The Cardinality LexBFS or CLBFS is a modification of LexBFS that always prefers a vertex with largest possible degree.

- We note that the reason we need CLBFS is that it guarantees that if $N(v)$ is a proper subset of $N(w)$, vertex v comes before w in the scheme.

- Algorithm CLBFS;
  1. for each vertex x do
     - Compute the $deg(x)$;
     - Initialize $label(x) \leftarrow ()$;
     - Initialize $position(x) \leftarrow undefined$;
  2. for $k \leftarrow n$ down to 1 do
     - Let S be the set of unpositioned vertices with largest label;
     - Let x be a vertex in S with largest degree;
     - Let $\sigma(k) = x$; set $\sigma^{-1}(x) = k$;
     - For each unpositioned vertex $y \in adj(x)$ do
       - Append k to the label of y;

  end;

• Example:



• ses: $\sigma = (13, 14, 15, 16, 17, 18, 11, 12, 8, 9, 10, 7, 5, 3, 6, 2, 4, 1)$

• Theorem: Given a chordal comparability graph, the CLBFS algrorithm constructs a ses.

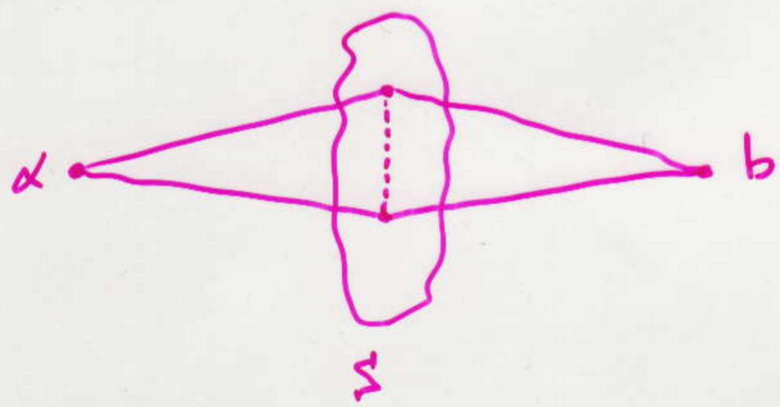
see: Borie, Spinrad, " Construction of a simple elimination scheme for a chordal comparability graph in linear time, DAM 91(1999) 287-292.

NC Algorithms for Triangulated Graphs

- Theorem: A graph $G=(V,E)$ is triangulated iff every minimal separator of G induces a clique in G.

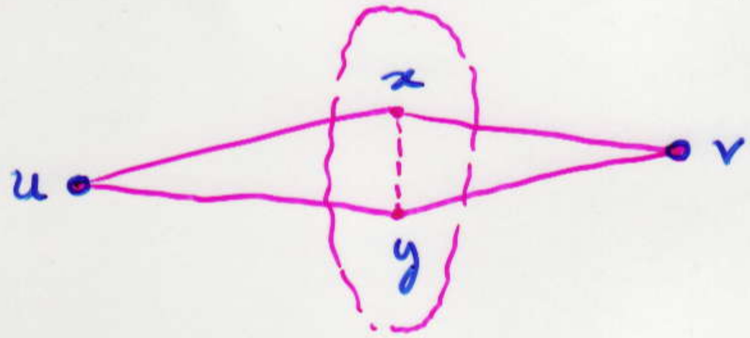
- A subset $S \subset V$ is a vertex separator for nonadjacent vertices a and b (ab-separator) if the removal of S from the graph separates a and b into distinct connected components.

- If no proper subset of S is an ab-separator then S is a minimal vertex separator for a and b.



• **Theorem:** A graph $G=(V,E)$ is chordal (triangulated), iff every minimal separator of G induces a clique in G.

Proof: Let u,v be two nonadjacent vertices of G, and S be the minimal separator of u,v.



If $|S|=1$, then S induces a clique.
Let $x,y \in S$. If x,y are adjacent for every pair of vertices of S, then S induces a clique.
Suppose $(x,y) \notin E$. Then, $uxvy = C_4$, contradiction

($\Leftarrow$) Suppose every minimal separator S is a clique.
Let $v_1, v_2, \dots, v_k, v_1$ be a chordless cycle.

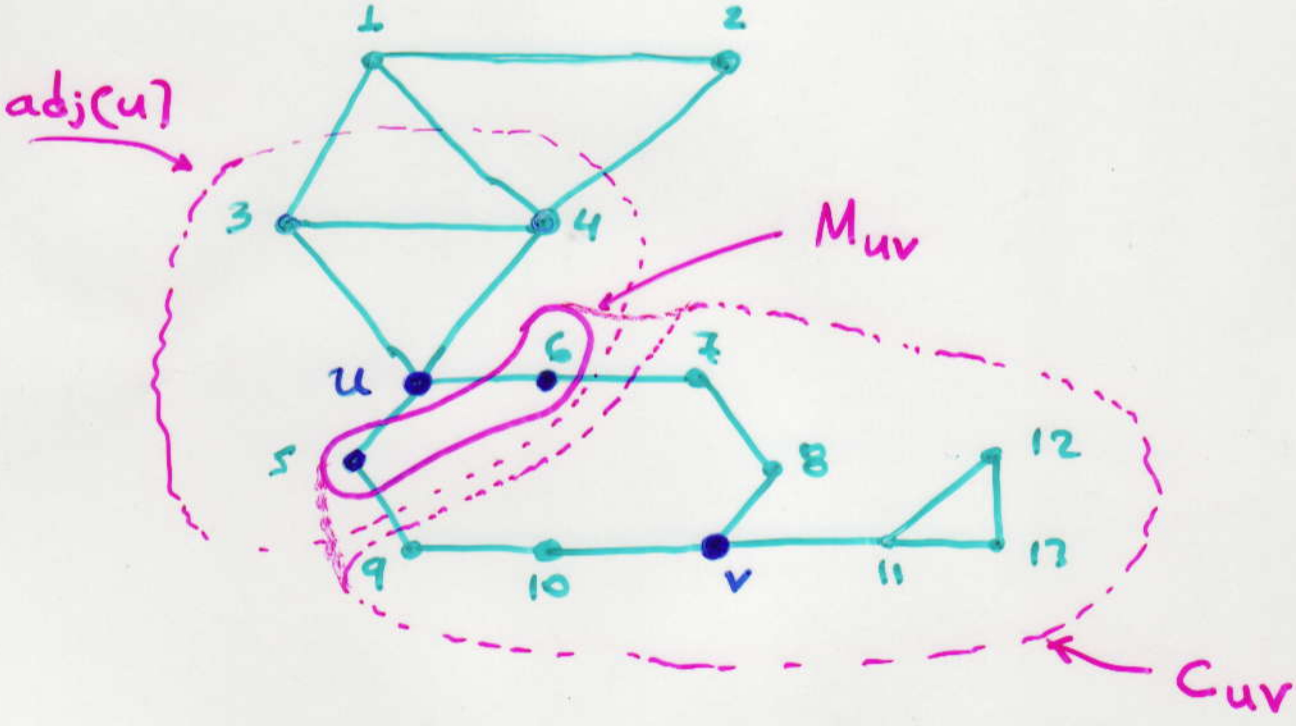
- v_1 and v_3 are nonadjacent.
- S_{13} contains v_2 κ' at least one of $v_4, v_5, \dots, v_k$.

That is, $S_{1,3}$ contains v_2 and v_i, where $4 \le i \le k$.
But, v_2, v_i are nonadjacent $\Rightarrow$ $S_{1,3}$ does not induce a clique

2.

- Triangulated (chordal) graph recognition.

- $G_u = G - adj(u)$
- C_{uv} = component of G_u containing v
- $M_{uv} = \{ x \mid x \in adj(u)$ & x is adj to some vertex in $C_{uv} \}$

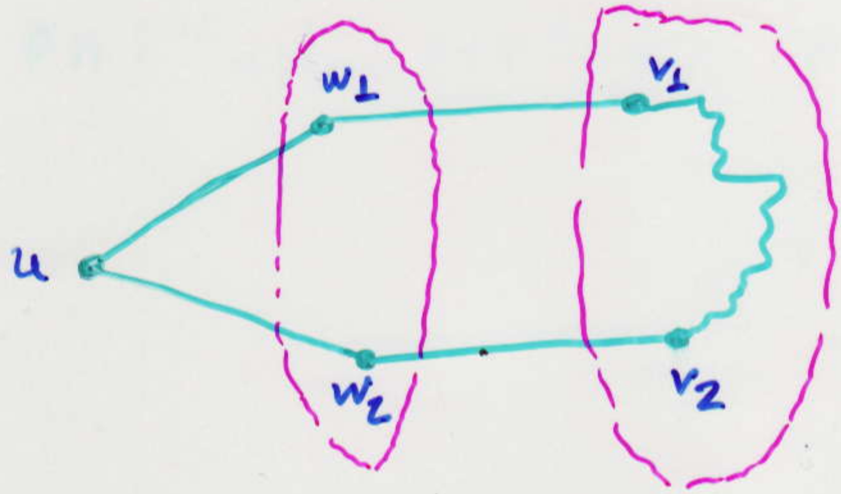


- Theorem: G is triangulated iff M_{uv} is a clique for every $u, v \in V$ such that $(u,v) \notin E$.

- Chandra & Iyengar: $O(\log n) - O(n^4)$ CREW

- Naor, Naor κ.' Schaffer

- $N(u)$ = set of vertices adj to u.
- $G-u$ = graph induced by $V-\{u\}$.
- If $W \subseteq V$, $G-W$ = graph induced by $V-W$



$N(u)$ $\quad$ $(G-u)-N(u)$, ή
$G-N[u]$

- **Theorem:** G is not triangulated iff it contains a vertex $u \in V$: a connected components of $(G-u)-N(u)$ is adjacent two vertices $w_1, w_2 \in N(u)$ which are not adjacent to each other.

- Algorithm : $O(\log^2 n) - O(m \cdot n^2)$ CREW.

- Minimum Spanning Tree

1999 (SODA), JACM, Vol. 48, No 2, 297-323 (2001)

Chong - Han - Lam

$O(\log n)$ $O(n+m)$ EREW

$\Downarrow$

Connected components

$O(\log n)$ $O(n+m)$ EREW

$\Downarrow$

Co-connected components

$G=(V,E)$ $O(n+m)$ $\Rightarrow$ $\overline{G}$ $\underline{\underline{O(n^2)}}$

- Lemma (SDH ϗ LP ϗ Chang)

Let G be an undirected graph on n vertices and m edges.
If v is G's vertex of minimum degree, then the subgraph $G(N(v))$, has fewer than $\sqrt{2m}$ vertices.

<u>Απόδειξη</u>. Since v has minimum degree $\Rightarrow$

$$\sum_x \text{degree}(x) \geq n \cdot \text{degree}(v)$$

$$\Rightarrow \text{degree}(v) \leq \frac{\sum_x \text{degree}(x)}{n} = \frac{2m}{n}$$

Additionally, since $m \leq \frac{n(n-1)}{2} < \frac{n^2}{2}$

we have that: $n > \sqrt{2m}$

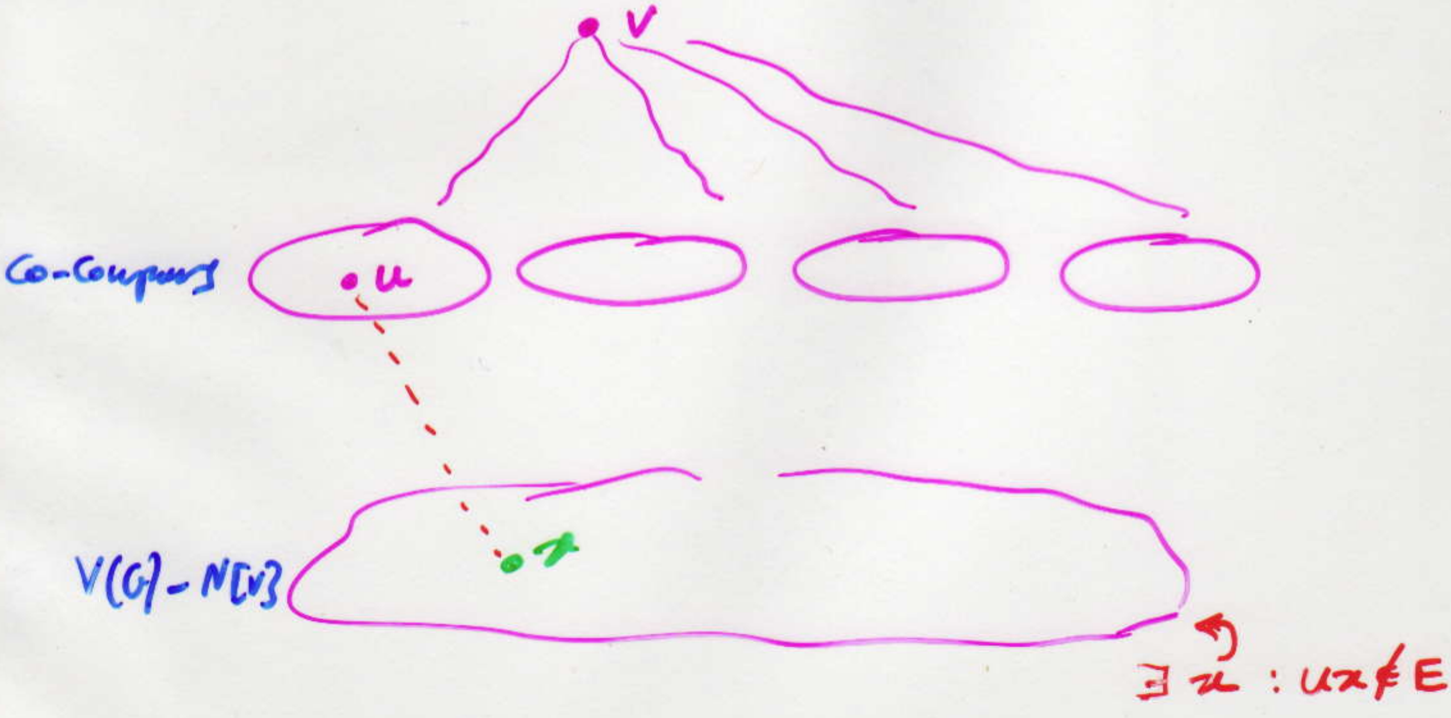
Thus, $\text{degree}(v) < \frac{2m}{\sqrt{2m}} = \sqrt{2m}$ ∎

Algorithm Par-Co-components.

1. Compute the degree(u), $\forall u \in V$;
   Locate the max; let v;

2. If $m < n-1$ or $d[v] = 0$ then
   $\text{co-comp}[u] \leftarrow v$, $\forall u \in V$;

3. Compute the connected compons of $\bar{G}(N(v))$;



$S \leftarrow \emptyset$; For each $u \in N(v)$ do

if $d_G[u] + d_{\bar{G}(N(v))}[u] < n-1$ then

$S \leftarrow S \cup \{u\}$;

Key complexity

$$|N(v)| = O(\sqrt{m})$$

Computation of degrees in $\overline{G}(N(v))$

$O(\log m)$ $\quad O\left(\frac{m}{\log m}\right)$ EREW

connected components : $O(\log N)$ $\quad O\left(\frac{N^2}{\log N}\right)$ EREW

$O(\log m)$ $\quad O\left(\frac{m}{\log m}\right)$

$\vdots$

$O(\log n)$ $\quad O\left(\frac{n+m}{\log n}\right)$ EREW

optimal

Comparability Graphs

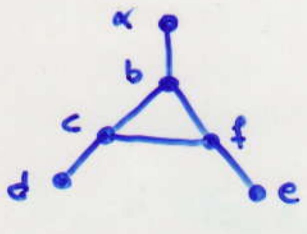
- A graph $G=(V,E)$ is a comparability graph if there exists an orientation (V,F) of G satisfying

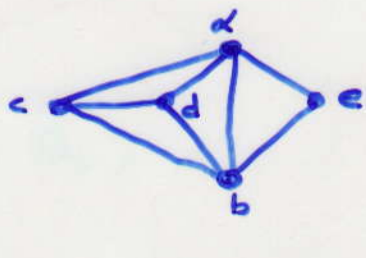
$$F \cap F^{-1} = \emptyset, \quad F + F^{-1} = E, \quad F^2 \subseteq F$$

where $F^2 = \{ ac \mid ab, bc \in F \text{ for some vertex } b \}$.

- Recall: Edges ab, cd are in the same implication class iff there exists a Γ-chain from ab to cd; $ab\,\Gamma^* cd$.

- Example:





$A = \{ab, cb, cd, cf, ef, bf, ba, bc,$
$dc, fc, fe, fb\}$

$A_1 = \{ab\}$
$A_2 = \{cd\}$
$A_3 = \{ac, ad, ae\}$
$A_4 = \{bc, bd, be\}$

- $A = \hat{A} = A \cup A^{-1}$

- $A \cap A^{-1} = \emptyset$

Algorithm Comparability Graph Recognition
Input: The edges of G;
Output: True if G is a comparability graph;

1. Compute the direct forcing relation $\sim$:
   for $1 \le e \le m$ do in parallel
      $ij \leftarrow$ the e^{th} edge
      For $1 \le s \le d(i)$ do in parallel
         $k \leftarrow$ the s^{th} adjacent vertex of i
         Test (in time $O(\log \delta)$) if k
         is in the sorted list of suc. of j
         If it is not, put $ij \sim ik$ and $ji \sim ki$.

2. Compute the implication classes as the connected components of the graph $(E, \sim)$.

3. Check if G is a comparability graph:

For $1 \le e \le m$ do in parallel

$ij \leftarrow$ the e^{th} edge

Find (in time $O(\log n)$) the number of ij using the sorted list of Succ. of j and read its implication class number;

If ij and ji have same implication class #
then $A[e] \leftarrow$ false
else $A(e) \leftarrow$ true

Return $\bigwedge_{1 \le e \le m} A(e)$

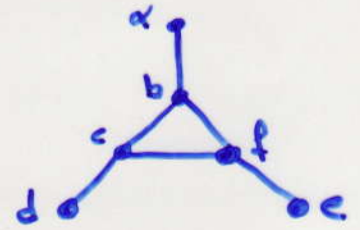
Theorem: The algorithm determines whether a graph G is a comparability graph, in $O(\log n)$ time using δn processors on the CRCW PRAM.

- The next theorem is of major importance since it legitimizes the use of G-decomposition as a constructive tool for deciding whether an undirected graph is a comparability graph.

- Theorem (TRO): Let $G=(V,E)$ be a graph with G-decomposition $E=\hat{B}_1+\hat{B}_2+\dots+\hat{B}_k$. The following statments are equivalent:
  - (i) $G=(V,E)$ is a comparability graph;
  - (ii) $A\cap A^{-1}=\phi$ for all implication classes A of E;
  - (iii) $B_i\cap B_i^{-1}=\phi$ for $i=1,2,\dots,k$;
  - (iv) every "circuit" of edges $v_1v_2, v_2v_3, \dots, v_qv_1 \in E$ such that $v_{q-1}v_1, v_qv_2, v_{i-1}v_{i+1} \notin E$ (for $i=2,3,\dots,q-1$) has even length.

- Furthemore, when these conditions hold, $B_1+B_2+\dots+B_k$ is a transitive orientation of E.

- Example:



circuit: ab, bc, cd, dc, cf, fe, ef, fb, ba has odd length.

- Algorithm TRO

1. Initialize: $i \leftarrow 1$; $E_i \leftarrow E$; $F \leftarrow \emptyset$;
2. Arbitrarily pick an edge $x_i y_i \in E_i$;
3. Enumerate the implication class B_i of E_i
   if $B_i \cap B_i^{-1} = \emptyset$ then add B_i to F;
   else "G is not a comparability graph"; STOP;
4. Define: $E_{i+1} \leftarrow E_i - \hat{B}_i$;
5. if $E_{i+1} = \emptyset$ then $k \leftarrow i$; output F; STOP;
   else $i \leftarrow i+1$; goto 2;

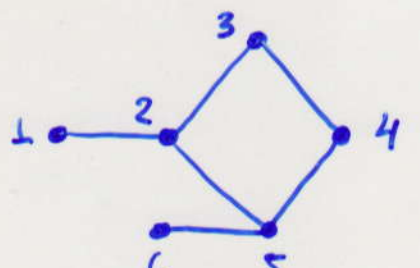
end.

Definition: Given an undirected graph $G=(V,E)$ we define a graph $G'=(V',E')$ in the following manner:

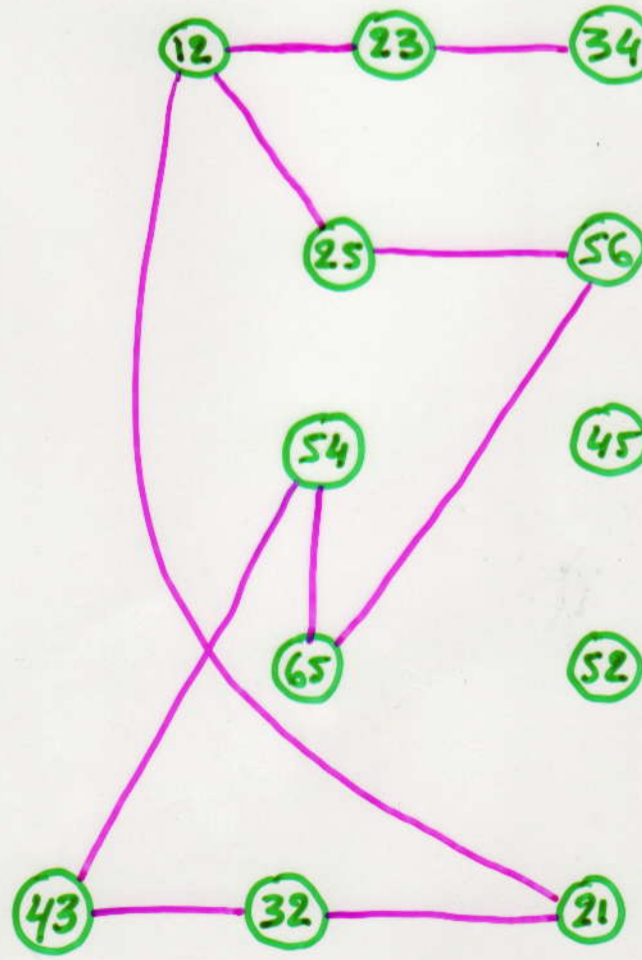
$$V' = \{ (i,j), (j,i) \mid ij \in E \}$$

$$E' = \{ (x,y) \leftrightarrow (y,z) \mid xz \notin E \}$$

Example:



$xx \notin E \Rightarrow$
$(x,y) \leftrightarrow (y,x)$
for every $xy \in E$.



Theorem: (Ghouilla-Houri, 1962) The following claims are equivalent:

1. The graph G is a comparability graph
2. the graph G' is bipartite

Proof: We will use GH characterization (Any odd cycle contains a short-chord).

The vertices of a cycle in G' correspond to adjacent edges in G (with common vertex), which induce a cycle in G with no short chords.

Thus, G' is bipartite $\Leftrightarrow$ G is a comparability graph.

Coloring Comparability Graphs

- Let F be an acyclic orientation (not necessarily transitive) of an undirected graph $G=(V,E)$.

- A height function h can be placed on V as follows:

$$h(v) = \begin{cases} 1 + \max\{h(w) \mid vw \in F\}; \\ 0 \text{ if } v \text{ is a sink}; \end{cases}$$

- The height function can be assigned in linear time using DFS.

- The function h is always a proper vertex coloring of G, but it is <u>not</u> necessarily a minimum coloring.

- #colors = # vertices in the longest path of F
$= 1 + \max\{h(v) \mid v \in V\}$

- The function h is a minimum coloring if F happens to be transitive.

- Suppose that G is a comparability graph, and let F be a transitive orientation of G.

- In such a case, every path in F corresponds to a clique of G because of transitivity.

- Thus, the height function will yield a coloring of G which uses exactly $\omega(G)$ colors, which is the best possible.

- Moreover, since being a comparability graph is a hereditary property, we find that $\omega(G_A) = \chi(G_A)$ for all induced subgraphs G_A of G.

- This proves: Every comparability graph is a perfect graph.

Maximum Weighted clique

- In general the maximum weighted clique problem is NP-complete, but when restricted to comparability graphs it becomes tractable.

- Algorithm MWClique

Input: A transitive orientation F of a comparab. graph $G=(V,E)$ and a weight function w defined on V.

Output: A clique K of G whose weight is max.

Method: We use a modification of the height calculation technique (DFS).

To each vertex v we associate its cumulative weight $W(v)$ = the weight of the heaviest path from v to some sink.

A pointer is assigned to v designating its successor on that heaviest path.