Triangulated Graphs

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

- triangulated $\equiv$ chordal $\iff$ 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, $C(v_i) = \text{clique}$
• It is easy to see that the set of cliques $C(v_i), i \in \mathbb{N}$, 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.

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

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

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

• $G_1$

• $G_2$

- $G = [1, 7, 2, 6, 3, 5, 4]$ no simplicial vertex

• $G_1$ has 96 different peo.
• Algorithms for computing PEO.

Algorithm LexBFS
begin
1. forall \( v \in V \) do \( \text{label}(v) := () \);
2. for \( i := |V| \) downto 1 do
3. \( \text{choose } v \in V \text{ with lex max } \text{label}(v) \);
4. \( G(i) \gets v \);
5. for all \( u \in V \cap N(v) \) do
6. \( \text{label}(u) \gets \text{label}(u) + i \)
7. \( V := V \setminus \{v\} \);
end

end

\( G = \{a\} \)
\( \text{label}(b) = (5) \)
\( \text{label}(c) = (5) \)

\( G = \{b, a\} \)
\( \text{label}(c) = (4) \)
\( \text{label}(d) = (4) \)
\( \text{label}(e) = (54) \)

\( G = \{e, b, a\} \)
\( \text{label}(d) = (43) \)
\( \text{label}(c) = (4) \)
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.   • $S(i) \leftarrow v$;
5.   • $V := V \setminus \{v\}$;
end

$G = \{\alpha\}$

$G = \{e, \alpha\}$

$G = \{b, e, \alpha\}$

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

$O(n+m)$. 
-24-
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. $v \leftarrow 6(i)$;
4. $X \leftarrow \{ x \in \text{adj}(v) \mid 6^{-1}(v) < 6^{-1}(x) \}$;
5. if $X \neq \emptyset$ then goto 8
6. $u \leftarrow 6(\min \{ 6^{-1}(x) \mid x \in X \})$;
7. $X \leftarrow \{ u \} \parallel A(u)$ ($X - \{ u \} \equiv X \cup u$)
8. if $A(v) - \text{adj}(v) \neq \emptyset$ then ret("false")
end
9. ret("true")
end

$A(v) \subseteq \text{adj}(v) \Rightarrow \text{true}$.

- $v=a$ $X = \{ e, b \}$
  - $u=e$ $A(e) = \{ b \}$ $A(u) - \text{adj}(u) = \emptyset$

- $v=e$ $X = \{ b, d \}$
  - $u=b$ $A(b) = \{ d \}$ $A(e) - \text{adj}(e) = \emptyset$

$c = \{ a, e, b, d, c \}$
Example:

\[ v = 1: \quad X = 3, 4, 8 \quad u = 3 \quad A(3) = 4, 8 \]
\[ v = 2: \quad X = 4, 7 \quad u = 4 \quad A(4) = 7 \quad A(2) \subseteq \text{adj}(2) \]
\[ v = 3: \quad X = 4, 5, 8 \quad u = 4 \quad A(4) = 7, 5, 8 \quad A(3) \subseteq \text{adj}(3) \]
\[ v = 4: \quad X = 5, 7, 8 \quad u = 5 \quad A(5) = 7, 8 \quad A(4) \subseteq \text{adj}(4) \]
\[ v = 5: \quad X = 6, 7, 8 \quad u = 6 \quad A(6) = 7, 8 \quad A(5) \subseteq \text{adj}(5) \]
\[ v = 6: \quad X = 7, 8 \quad u = 7 \quad A(7) = 8 \quad A(6) \subseteq \text{adj}(6) \]
- **Correctness of the Algorithm**

  **Theorem:** The algorithm **PERFECT** returns **TRUE** if and only if \( G \) is a p.e.o.

  **Proof:**

  (\( \Leftarrow \)) Suppose the algorithm returns **FALSE** on iteration number \( \delta^{-1}(u) \).

  This may happen only if in stage \( 8 \):

  \[
  A(u) - \text{adj}(u) \neq \emptyset \quad \Rightarrow \quad w \in A(u) \land w \notin \text{adj}(w)
  \]

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

  Thus, \( \delta^{-1}(v) < \delta^{-1}(u) < \delta^{-1}(w) \) for \( u, w \in \text{adj}(v) \).

  But, since \( uvw \notin E \Rightarrow G \) is not p.e.o.
Suppose $6$ is not a p.e.o and the Algorithm returns TRUE.

Let $v$ be a vertex with max index in $6$, such that $X_v = \{ w \mid w \in \text{adj}(v) \text{ and } 6(v) < 6(w) \}$ does not induce a clique.

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

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

Since the Algorithm returns TRUE in stage 6 $\Rightarrow$ $A(u) \subseteq \text{adj}(u) \Rightarrow X_v \setminus \{u\} \subseteq \text{adj}(u)$

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

25-3.
• Since $C^{-1}(v)$ is chosen to be the max index in $C$ 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: $O(n^2mu)$


- **X and maximal cliques**

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

  $$ \emptyset \cup X_v $$

  where

  $$ X_v = \{ x \in \text{adj}(v) \mid \overline{G}(v) < \overline{G}(x) \} $$

  (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 X-and-max-clique

  - Some of $\emptyset \cup X_v$ will not be max clique.
  - $\emptyset \cup X_v$ is not maximal clique iff for some $i$, Algorithm PERFECT, $X_v$ is the set which is concatenated to $A(u)$. 

-26-
Example (a):

\[ \{1\} \cup X_1 = \{1,6,7,8\} \]
\[ \{2\} \cup X_2 = \{2,5,6\} \]
\[ \{3\} \cup X_3 = \{3,5\} \]
\[ \{4\} \cup X_4 = \{4,8\} \]
\[ \{5\} \cup X_5 = \{5,6,7\} \]

\[ \text{maximal} \]

\[ G = \{1,2,3,4,5,6,7,8\} \]
\[ \{6\} \cup X_6 = \{6,7,8\} \]
\[ \{7\} \cup X_7 = \{7,8\} \]
\[ \{8\} \cup X_8 = \{8\} \]

\[ \text{not maximal} \]

Example (c):

\[ \{6\} \cup X_6 = \{6,7,8\} \quad X_6 = \{7,8\} \]

\[ \begin{align*}
1 = 1 & \implies v = 1 \\
u = 6 & \implies X = \{6,7,8\} \quad X - \{6\} = \{7,8\}
\end{align*} \]
Example (c):

- \( \{3 \} \cup X_3 = \{3, 4, 5\} \)
  \( X_3 = \{4, 5\} \)

\[ 6 = \{1, 2, 3, 4, 5\} \]

\( i = 2 \implies v = 2 \)

\( u = 3 \implies X_2 = \{3, 4, 5\} \implies X_2 - \{3\} = \{4, 5\} \)

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

\( i = 1 \implies v = 1 \)

\( u = 2 \implies X_1 = \{2, 3\} \implies X_1 - \{2\} = \{3\} \)
Characterizing Triangulated Graphs

- **Definition:** A subset $SCV$ is called a vertex separator for nonadjacent vertices $\alpha, \beta$ (or $\alpha$-$\beta$ separator), if in $G_r \cdot S$ vertices $\alpha$ and $\beta$ are in different connected components.

If no proper subset of $S$ is an $\alpha$-$\beta$ separator, $S$ is called minimal vertex separator.

- **Example:**

  ![Diagram of a graph](attachment:image.png)

  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 $[x, x_1, \ldots, x_r, y]$ ($x_1 \in A$) and $[x, b_1, \ldots, b_s, y]$ ($b_1 \in B$).
• Since $[x, a_1, \ldots, a_r, y, b_1, \ldots, b_k, z]$ is a simple cycle of length $l \geq 4$, it contains a chord.

• For every $i, j$ $a_i \sim b_i \not\in E$ ($S$ is $x$-$b$ separator) and also $a_i \sim a_j \not\in E$, $b_i \sim b_j \not\in E$ (by the minimality of the paths)

• Thus, $xy \not\in E$.

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

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

• $v_2$ and $v_3$ are nonadjacent.

• $S_{1,3}$ contains $v_2$ and at least one of $v_4, v_5, \ldots, v_k$.

But $v_2, v_i \ (i=4, 5, \ldots, k)$ are nonadjacent $\Rightarrow$ $S_{1,3}$ do not induce a dique.
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 poset;
- work backwards, assign to each $v_i$ in turn the minimum color not assigned to its higher neighbors.

**Example:**

$G = \{x, c, b, d, e, f\}$

$3 \not\subseteq 2 \subseteq 1$

$\omega(G) = \chi(G)$
Finding the Stability number $\alpha(G)$

- Gavril gives the following solution.

**Method:**

- Let $G$ be a peo of a chordal graph $G$.
- Define inductively a sequence of
  vertices $y_1, y_2, \ldots, y_t$ as follow:
  
  \[ y_1 = G(1); \]
  \[ y_i \text{ is the first vertex in } G \text{ which follows } y_{i-1} \text{ and which is not } \]
  \[ y_i \text{ in } X_{y_1} \cup X_{y_2} \cup \ldots \cup X_{y_{i-1}}; \]
  All vertices following $y_t$ are in $X_{y_1} \cup X_{y_2} \cup \ldots \cup X_{y_t}.$

**Example:**

\[ G = \{a, c, b, d, e, f\} \]

\[ y_1 y_2 \cdots y_3 \cdot \]

\[ X_{y_1} = \{b, f\} \]
\[ X_{y_2} = \{b, d\} \]
\[ X_{y_3} = \{f\} \]
Theorem: The set \( \{Y_1, Y_2, \ldots, Y_t\} \) is a maximum stable set of \( G \), and the collection of sets \( Y_i = \{y_i\} \cup X_{y_i}, 1 \leq i \leq t \), comprises a minimum clique cover of \( G \).

Proof. The set \( \{Y_1, Y_2, \ldots, Y_t\} \) 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 \( \{Y_1, \ldots, Y_t\} \) is a clique cover of \( G \). Thus, \( \alpha(G) = \kappa(G) = t \).

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:

![Diagram of chordal graphs]

- [Diagram of chordal graphs with labeled nodes A, B, C, D, E, F, G, H, showing the relationship between the chordal graph and the tree T.]
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 statements are equivalent.
  1. \( G \) is an interval graph.
  2. \( G \) contains no \( C_4 \) and \( \bar{G} \) is a comparability graph.
  3. 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 \( l > 3 \) 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 \( \pi = (v_1, v_2, ..., v_n) \) such that each \( v_i \) is simplicial in \( G - \{v_1, ..., v_{i-1}\} \) and the neighborhoods of the vertices adjacent to \( v_i \) in \( G - \{v_1, ..., v_{i-1}\} \) 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 \( \text{deg}(x) \);
     - initialize \( \text{label}(x) \leftarrow 1 \);
     - initialize \( \text{position}(x) \leftarrow \text{undefined} \);
  2. for \( k \leftarrow n \) down to 1 do
     - Let \( S \) be the set of unpositioned vertices with largest labels;
     - Let \( x \) be a vertex in \( S \) with largest degree;
     - Let \( c(k) = x \); set \( G^+(x) = k \);
     - for each unpositioned vertex \( y \in \text{adj}(x) \) do
       - Append \( k \) to the label of \( y \);
Example:

\[ G = (13, 14, 15, 16, 17, 18, 11, 12, 8, 9, 10, 7, 5, 3, 6, 2, 4, 1) \]

- Theorem: Given a chordal comparability graph, the CLBFS algorithm constructs a ses.

• 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 \( S \subseteq V \) is a vertex separator for nonadjacent vertices \( \alpha \) and \( \beta \) (\( \alpha \beta \)-separator) if the removal of \( S \) from the graph separates \( \alpha \) and \( \beta \) into distinct connected components.

• If no proper subset of \( S' \) is a \( \alpha \beta \)-separator then \( S' \) is a minimal vertex separator for \( \alpha \) and \( \beta \).
Theorem: A graph $G=(V,E)$ is chordal (triangulated), if and only if every minimal separator of $G$ induces a clique in $G$.

Proof: Let $u, v$ be two non-adjacent 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 $(x, y) \not\in E$. Then, $uxvy = C_4$, contradiction.

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

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

That is, $S_{1,3}$ contains $v_2$ and $v_i$, where $4 \leq i \leq k$.
But, $v_2, v_i$ are non-adjacent $\Rightarrow S_{1,3}$ does not induce a clique.
- Trangulated (chordal) graph recognition.
  - $G_u = G - \text{adj}(u)$
  - $Cuv = \text{component of } G_u \text{ containing } v$
  - $Muv = \{ x \mid x \in \text{adj}(u) \land x \text{ is adj to some vertex in } Cuv \}$

![Graph diagram]

- Theorem: $G$ is trangulated iff $Muv$ is a clique for every $u, v \in V$ such that $(u, v) \in E$.

- Chandra & Iyengar: $O(\log n) - O(n^4)$ CREW
- Naor, Naor's 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)       (G - u) - N(u), u \in
G - N[u]
```

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

- **Algorithm**: \( O(bay^2u) - O(m.n^2) \) CREW.
Minimum Spanning Tree


Chong - Han - Lam

$O(\log n) \quad O(n \log n) \quad EREW$

↓

Connected Components

$O(n \log n) \quad O(n \log n) \quad EREW$

↑

Co-connected Components

$G=(V,E) \quad O(n \log n) \Rightarrow \overline{G} \quad O(n^2)$
• Lemma (SDH, LP by Chong)
  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.

  \textbf{Answer.} Since $v$ has minimum degree $\Rightarrow$
  \begin{equation}
  \sum_{x} \text{degree}(x) > n \cdot \text{degree}(v)
  \end{equation}

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

  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
   \( \text{co-comp}[u] = v \), \( \forall u \in V \);

3. Compute the connected components of \( \tilde{G}(N(v)) \);

\[ S \leftarrow \emptyset; \text{ for each } u \in N(v) \text{ do} \]
\[ \text{if } d[u] + d[\tilde{G}(N(u))[u]] < n - 1 \text{ then } \]
\[ u \in S; \]\
Key complexity

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

Computation of degrees in $G(N(v))$

$O(k\gamma m)$ \quad $O\left(\frac{\mu}{k\gamma m}\right)$ \quad EREW

Connected components: $O(k\gamma N)$ \quad $O\left(\frac{N^2}{k\gamma N}\right)$ \quad EREW

\[ O(k\gamma m) \quad O\left(\frac{n}{k\gamma m}\right) \]

\[ \vdots \]

\[ O(k\gamma m) \quad O\left(\frac{n+m}{k\gamma m}\right) \quad \text{EREW} \]

Optimal
Comparability Graphs

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

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

where $F^2 = \{x \in V | x \in F, \exists y \in V, y \in F \}$ for some vertex $b \in V$.

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

- Example:

\begin{align*}
A &= \{ab, bc, cd, cf, ef, bf, ba, bc, dc, fc, fe, fb\} \\
A_1 &= \{ab, bc\} \\
A_2 &= \{cd\} \\
A_3 &= \{ac, ad, ae\} \\
A_4 &= \{bc, bd, ce\}
\end{align*}

$A = \hat{A} = AU A^{-1}$
Algorithm Comparability Graph Recognition
Input: The edges of G;
Output: True if G is a comparability graph;

1. Compute the direct forcing relation v:
   for \(1 \leq e \leq m\) do in parallel
   \(ij \leftarrow\) the \(e\)th edge
   for \(1 \leq s \leq d(i)\) do in parallel
   \(k \leftarrow\) the \(s\)th adjacent vertex of \(i\)
   Test (in time \(O(\log s)\)) if \(k\) is in the sorted list of succ. of \(j\)
   If it is not, put \(ij \in k\) and \(j \in k\).

2. Compute the implication classes as the
Connected components of the graph \((E, v)\).
3. Check if $G$ is a comparability graph:
   
   For $1 \leq e \leq m$ do in parallel
   
   $ij \leftarrow$ the $e$th edge
   
   Find (in time $O(\log n)$) the number of $ij$ using the sorted list of $\pi_{\text{src} \cdot \text{trg}}$
   
   and read its implication class number;

   If $ij$ and $ji$ have same implication class
   
   then $A(e) \leftarrow \text{false}$
   
   else $A(e) \leftarrow \text{true}$

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

Theorem: The algorithm determines whether a graph $G$ is a comparability graph, in $O(\log n)$ time using $\frac{n}{\log n}$ 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 $E = \hat{B}_1 + \hat{B}_2 + \ldots + \hat{B}_k$. The following statements are equivalent:

(i) $G = (V, E)$ is a comparability graph;
(ii) $AN A^{-1} = \emptyset$ for all implication classes $A$ of $E$;
(iii) $B_i \cap B_i^{-1} = \emptyset$ for $i = 1, 2, \ldots, k$;
(iv) every "circuit" of edges $v_1v_2, v_2v_3, \ldots, v_qv_1 \in E$ such that $v_{q-1}v_1, v_qv_2, v_{i-1}v_{i+1} \notin E$ (for $i = 2, 3, \ldots, q-1$) has even length.

Furthermore, when these conditions hold, $B_1 + B_2 + \ldots + B_k$ is a transitive orientation of $E$. 
Example:

\[
\begin{align*}
\text{circuit: } & ab, bc, cd, dc, cf, fe, ef, \\
& fb, ba \\
\text{has odd length.}
\end{align*}
\]

Algorithm TRO

1. Initialize: \( i=1 \); \( E_i = E \); \( F = \emptyset \);
2. Arbitrarily pick an edge \( x_iy_i \in E_i \).
3. Enumerate the implication class \( B_i \) of \( E_i \)
   
   \[
   \begin{cases}
   \text{if } B_i \cap B_i^{-1} = \emptyset \text{ then add } B_i \text{ to } F; \\
   \text{else } "G \text{ is not a comparability graph}"; \text{ stop;}
   \end{cases}
   \]
4. Define: \( E_{i+1} = E_i - B_i \).
5. if \( E_{i+1} = \emptyset \) then \( k \leftarrow 1; \) 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:

$xz \notin E \Rightarrow (xz) \leftrightarrow (yz)$
for every $yz \in E$. 
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 $\iff$ $G$ 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:
  \[
  h(v) = \begin{cases} 
  1 + \max \{ h(w) \mid vw \in F \} & \text{if } v \text{ is a source;} \\
  0 & \text{if } v \text{ is a sink;}
  \end{cases}
  \]

- 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 not 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 comparability 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.