Algorithmic Graph Theory (SS2016)
Chapter 5
Perfect Graphs

Walter Unger

Lehrstuhl für Informatik 1

15:46, June 14, 2016
Contents I

1. Introduction
   - Reminder
   - Definition
   - Bipartite Graphs
   - Comparability Graphs
   - Statements
   - Interval Graphs

2. Theorems
   - Statements

3. Chordal Graphs

4. Clique-Separators
   - Chordal Graphs
   - Clique-Separator
   - Fill-In
   - MES
   - Clique-Separable

Definition
- Statements
- Recognition
- Algorithmen
- An alternative Characterisation
- Statements
Reminder 1

- Colouring is hard!
Reminder I

- Colouring is hard!
- Colouring is NP-complete.
Reminder 1

- Colouring is hard!
- Colouring is NP-complete.
- Colouring is not approximable.
Reminder 1

- Colouring is hard!
- Colouring is NP-complete.
- Colouring is not approximable.
- There are no good bounds known.
Reminder 1

- Colouring is hard!
- Colouring is NP-complete.
- Colouring is not approximable.
- There are no good bounds known.
- **Question:** is there a graph class with good bounds?
Reminder II

Definition

Let $G = (V, E)$ be a graph.

$$\alpha(G) = \max\{ |V'| ; \ V' \subset V \land \forall a, b \in V' : (a, b) \notin E \}$$
Reminder II

**Definition**

Let $G = (V, E)$ be a graph.

$$\alpha(G) = \max\{ |V'| ; \ V' \subset V \land \forall a, b \in V' : (a, b) \notin E \}$$

$$\omega(G) = \max\{ |V'| ; \ V' \subset V \land \forall a, b \in V' : (a, b) \in E \}$$
Definition

Let $G = (V, E)$ be a graph.

\[
\alpha(G) = \max\{ |V'| ; V' \subseteq V \land \forall a, b \in V' : (a, b) \notin E \} \\
\omega(G) = \max\{ |V'| ; V' \subseteq V \land \forall a, b \in V' : (a, b) \in E \} \\
\chi(G) = \min\{ k ; \exists V_1, V_2, \ldots, V_k : \bigcup_{i=1}^k V_i = V \land \forall i : 1 \leq i \leq k : \forall a, b \in V_i : (a, b) \notin E \}
\]
Reminder II

**Definition**

Let $G = (V, E)$ be a graph.

\[
\begin{align*}
\alpha(G) &= \max\{ |V'| ; \ V' \subset V \land \forall a, b \in V' : (a, b) \notin E \} \\
\omega(G) &= \max\{ |V'| ; \ V' \subset V \land \forall a, b \in V' : (a, b) \in E \} \\
\chi(G) &= \min\{ k ; \ \exists V_1, V_2, \ldots, V_k : \bigcup_{i=1}^k V_i = V \land \\
&\quad \forall i : 1 \leq i \leq k : \forall a, b \in V_i : (a, b) \notin E \} \\
\overline{\chi}(G) &= \min\{ k ; \ \exists V_1, V_2, \ldots, V_k : \bigcup_{i=1}^k V_i = V \land \\
&\quad \forall i : 1 \leq i \leq k : \forall a, b \in V_i : (a, b) \in E \}
\end{align*}
\]
Reminder II

**Definition**

Let \( G = (V, E) \) be a graph.

\[
\begin{align*}
\alpha(G) &= \max\{ |V'| ; V' \subset V \land \forall a, b \in V' : (a, b) \notin E \} \\
\omega(G) &= \max\{ |V'| ; V' \subset V \land \forall a, b \in V' : (a, b) \in E \} \\
\chi(G) &= \min\{ k ; \exists V_1, V_2, \ldots, V_k : \bigcup_{i=1}^k V_i = V \land \forall i : 1 \leq i \leq k : \forall a, b \in V_i : (a, b) \notin E \} \\
\overline{\chi}(G) &= \min\{ k ; \exists V_1, V_2, \ldots, V_k : \bigcup_{i=1}^k V_i = V \land \forall i : 1 \leq i \leq k : \forall a, b \in V_i : (a, b) \in E \}
\end{align*}
\]

**Further notations:**

\[
\omega(G) = \overline{\alpha}(G),
\chi(G) = \overline{\omega}(G) = \beta_0(G),
\kappa(G) = \overline{\chi}(G)
\]
Statements I

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

**Theorem**

Let \( G = (V, E) \) be a graph. Then we have:

\[
\alpha(G) = \overline{\alpha}(G) \quad \text{and} \quad \chi(G) = \overline{\chi}(G)
\]

**Proof:**

\[
\begin{align*}
\alpha(G) &= \max \{ |V'|; \ V' \subset V \land \forall a, b \in V': (a, b) \not\in E \}\ \\
\omega(G) &= \max \{ |V'|; \ V' \subset V \land \forall a, b \in V': (a, b) \in E \}\ \\
\chi(G) &= \min \{ k; \ \exists V_1, V_2, \ldots, V_k : \bigcup_{i=1}^k V_i = V \land \\
&\quad \forall i: 1 \leq i \leq k : \forall a, b \in V_i : (a, b) \not\in E \}\ \\
\overline{\chi}(G) &= \min \{ k; \ \exists V_1, V_2, \ldots, V_k : \bigcup_{i=1}^k V_i = V \land \\
&\quad \forall i: 1 \leq i \leq k : \forall a, b \in V_i : (a, b) \in E \}
\end{align*}
\]
Let $G = (V, E)$ be a graph with $n = |V|$. Then we have:

$$\frac{n}{\alpha(G)} \leq \chi(G) \leq n - \alpha(G) + 1.$$
Statements III

\[ \omega(G) = \overline{\alpha}(G), \quad \alpha(G) = \omega(G) = \beta_0(G), \quad \kappa(G) = \overline{\chi}(G) \]

**Theorem**

Let \( G = (V, E) \) be a graph with \( n = |V| \). Then we have:

\[ 2\sqrt{n} \leq \chi(G) + \overline{\chi}(G) \leq n + 1 \]
\[ n \leq \chi(G) \cdot \overline{\chi}(G) \leq \left( \frac{n+1}{2} \right)^2. \]

Idea of proof:

\[ \chi(G) = \min\{ k ; \exists V_1, V_2, \ldots, V_k : \bigcup_{i=1}^{k} V_i = V \wedge \forall i : 1 \leq i \leq k : \forall a, b \in V_i : (a, b) \notin E \} \]
\[ \overline{\chi}(G) = \min\{ k ; \exists V_1, V_2, \ldots, V_k : \bigcup_{i=1}^{k} V_i = V \wedge \forall i : 1 \leq i \leq k : \forall a, b \in V_i : (a, b) \in E \} \]
Theorem

Let $G = (V, E)$ be a graph with $n = |V|$. Then we have:

\[
2\sqrt{n} \leq \chi(G) + \overline{\chi}(G) \leq n + 1
\]
\[
n \leq \chi(G) \cdot \overline{\chi}(G) \leq \left(\frac{n+1}{2}\right)^2.
\]

Idea of proof:

\[
\chi(G) = \min \{ k ; \exists V_1, V_2, \ldots, V_k : \bigcup_{i=1}^k V_i = V \land \\
\forall i : 1 \leq i \leq k : \forall a, b \in V_i : (a, b) \notin E \}
\]
\[
\overline{\chi}(G) = \min \{ k ; \exists V_1, V_2, \ldots, V_k : \bigcup_{i=1}^k V_i = V \land \\
\forall i : 1 \leq i \leq k : \forall a, b \in V_i : (a, b) \in E \}
\]

Consider the two Coverings as a grid.
Statements III

$$\omega(G) = \bar{\alpha}(G), \alpha(G) = \bar{\omega}(G) = \beta_0(G), \kappa(G) = \bar{\chi}(G)$$

$$2\sqrt{n} \leq \chi(G) + \bar{\chi}(G) \leq n + 1$$

$$n \leq \chi(G) \cdot \bar{\chi}(G) \leq \left(\frac{n+1}{2}\right)^2.$$
A graph $G = (V, E)$ is called:

1. $\chi$-perfect, iff for all node-induced subgraphs $H$ of $G$ holds: $\chi(H) = \omega(H)$.
2. $\alpha$-perfect, iff for all node-induced subgraphs $H$ of $G$ holds: $\kappa(H) = \alpha(H)$.
3. perfect, if it is $\chi$-perfect [and $\alpha$-perfect].

\[
\begin{align*}
\alpha(G) &= \max\{ |V'| ; \ V' \subset V \land \forall a,b \in V' : (a,b) \notin E \} \\
\omega(G) &= \max\{ |V'| ; \ V' \subset V \land \forall a,b \in V' : (a,b) \in E \} \\
\chi(G) &= \min\{ k ; \ \exists V_1, V_2, \ldots, V_k : \bigcup_{i=1}^k V_i = V \land \\
&\quad \forall i : 1 \leq i \leq k : \forall a,b \in V_i : (a,b) \notin E \} \\
\overline{\chi}(G) &= \min\{ k ; \ \exists V_1, V_2, \ldots, V_k : \bigcup_{i=1}^k V_i = V \land \\
&\quad \forall i : 1 \leq i \leq k : \forall a,b \in V_i : (a,b) \in E \}
\end{align*}
\]
Definitions

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \chi(G) \]

**Definition**

A graph \( G = (V, E) \) is called:

1. \( \chi \)-perfect, iff for all node-induced subgraphs \( H \) of \( G \) holds: \( \chi(H) = \omega(H) \).
2. \( \alpha \)-perfect, iff for all node-induced subgraphs \( H \) of \( G \) holds: \( \kappa(H) = \alpha(H) \).
3. perfect, if it is \( \chi \)-perfect [and \( \alpha \)-perfect].

**Definition**

A property \( E \) of a graph \( G = (V, E) \) is called **hereditary**, iff the property holds for each node-induced subgraph of \( G \).
Examples ($\chi$-perfect)

- Planar graphs:

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\kappa}(G) \]
Examples ($\chi$-perfect)

- Planar graphs: no
- Interval graphs:

$$\omega(G) = \overline{\alpha(G)}, \alpha(G) = \overline{\omega(G)} = \beta_0(G), \kappa(G) = \overline{\chi(G)}$$
Examples ($\chi$-perfect)

- Planar graphs: no
- Intervall-graphs: yes
- Arc-graphs:

$$\omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G)$$
Examples ($\chi$-perfect)

- Planar graphs: no
- Intervall-graphs: yes
- Arc-graphs: no
- Permutation-graphs:
Examples ($\chi$-perfect)

- Planar graphs: no
- Intervall-graphs: yes
- Arc-graphs: no
- Permutation-graphs: yes
- Outerplanar graphs:

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]
Examples ($\chi$-perfect)

- Planar graphs: no
- Intervall-graphs: yes
- Arc-graphs: no
- Permutation-graphs: yes
- Outerplanar graphs: no
- Maximal outerplanar graphs:

$$\omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G)$$
Examples ($\chi$-perfect)

- Planar graphs: no
- Intervall-graphs: yes
- Arc-graphs: no
- Permutation-graphs: yes
- Outerplanar graphs: no
- Maximal outerplanar graphs: yes
- Maximal planar graphs:

$$\omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G)$$
Examples ($\chi$-perfect)

- Planar graphs: no
- Intervall-graphs: yes
- Arc-graphs: no
- Permutation-graphs: yes
- Outerplanar graphs: no
- Maximal outerplanar graphs: yes
- Maximal planar graphs: no (following slide)
- Bipartite graphs:
Examples ($\chi$-perfect)

- Planar graphs: no
- Intervall-graphs: yes
- Arc-graphs: no
- Permutation-graphs: yes
- Outerplanar graphs: no
- Maximal outerplanar graphs: yes
- Maximal planar graphs: no (following slide)
- Bipartite graphs: yes
- K-Trees:
Examples ($\chi$-perfect)

- Planar graphs: no
- Intervall-graphs: yes
- Arc-graphs: no
- Permutation-graphs: yes
- Outerplanar graphs: no
- Maximal outerplanar graphs: yes
- Maximal planar graphs: no (following slide)
- Bipartite graphs: yes
- K-Trees: yes
- Complement of a bipartite graph:
Examples ($\chi$-perfect)

- Planar graphs: no
- Interval-graphs: yes
- Arc-graphs: no
- Permutation-graphs: yes
- Outerplanar graphs: no
- Maximal outerplanar graphs: yes
- Maximal planar graphs: no (following slide)
- Bipartite graphs: yes
- K-Trees: yes
- Complement of a bipartite graph: yes (following slides)
- Cycles of odd length $\geq 5$:

$$\omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G)$$
Examples ($\chi$-perfect)

- Planar graphs: no
- Intervall-graphs: yes
- Arc-graphs: no
- Permutation-graphs: yes
- Outerplanar graphs: no
- Maximal outerplanar graphs: yes
- Maximal planar graphs: no (following slide)
- Bipartite graphs: yes
- K-Trees: yes
- Complement of a bipartite graph: yes (following slides)
- Cycles of odd length $\geq 5$: no
- Linegraphs of bipartite graphs:
Examples ($\chi$-perfect)

- Planar graphs: no
- Intervall-graphs: yes
- Arc-graphs: no
- Permutation-graphs: yes
- Outerplanar graphs: no
- Maximal outerplanar graphs: yes
- Maximal planar graphs: no (following slide)
- Bipartite graphs: yes
- K-Trees: yes
- Complement of a bipartite graph: yes (following slides)
- Cycles of odd length $\geq 5$: no
- Linegraphs of bipartite graphs: yes (following slides)
Example Planar

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]
Example Planar

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]
Example Planar

\[ \omega(G) = \bar{\alpha}(G), \, \alpha(G) = \bar{\omega}(G) = \beta_0(G), \, \kappa(G) = \bar{\chi}(G) \]
Complement of a bipartite Graph

$\omega(G) = \overline{\alpha}(G)$, $\alpha(G) = \overline{\omega}(G) = \beta_0(G)$, $\kappa(G) = \overline{\chi}(G)$

**Lemma**

*The complement of a bipartite graph is $\chi$-perfect.*

**Proof:**
Lemma

The complement of a bipartite graph is \( \chi \)-perfect.

Proof:

- Note, that the class is hereditary.
Complement of a bipartite Graph

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

Lemma

The complement of a bipartite graph is \( \chi \)-perfect.

Proof:

- Note, that the class is hereditary.
- Show \( \chi'(G) = \omega(G) \).
Complement of a bipartite Graph

Lemma

The complement of a bipartite graph is $\chi$-perfect.

Proof:

- Note, that the class is hereditary.
- Show $\chi(\overline{G}) = \omega(\overline{G})$.
- So we have to prove: $\kappa(G) = \alpha(G)$. 

$\omega(G) = \overline{\alpha}(G)$, $\alpha(G) = \overline{\omega}(G) = \beta_0(G)$, $\kappa(G) = \overline{\chi}(G)$
Complement of a bipartite Graph

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

**Lemma**

*The complement of a bipartite graph is \( \chi \)-perfect.*

**Proof:**

- Note, that the class is hereditary.
- Show \( \chi(G) = \omega(G) \).
- So we have to prove: \( \kappa(G) = \alpha(G) \).
- By the theorem of König we get:
Complement of a bipartite Graph

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

**Lemma**

*The complement of a bipartite graph is \( \chi \)-perfect.*

Proof:

- Note, that the class is hereditary.
- Show \( \chi(\overline{G}) = \omega(\overline{G}) \).
- So we have to prove: \( \kappa(G) = \alpha(G) \).
- By the theorem of König we get:
  - Take a maximum matching \( M \) with \( |M| = a \).
Complement of a bipartite Graph

Lemma

The complement of a bipartite graph is \( \chi \)-perfect.

Proof:

- Note, that the class is hereditary.
- Show \( \chi(\overline{G}) = \omega(\overline{G}) \).
- So we have to prove: \( \kappa(G) = \alpha(G) \).
- By the theorem of König we get:
  - Take a maximum matching \( M \) with \(|M| = a\).
  - Assume that \( b \) nodes are not covered by \( M \).
Complement of a bipartite Graph

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

**Lemma**

The complement of a bipartite graph is \( \chi \)-perfect.

**Proof:**

- Note, that the class is hereditary.
- Show \( \chi(G) = \omega(G) \).
- So we have to prove: \( \kappa(G) = \alpha(G) \).
- By the theorem of König we get:
  - Take a maximum matching \( M \) with \( |M| = a \).
  - Assume that \( b \) nodes are not covered by \( M \).
  - Then we have: \( \alpha(G) = a + b \) and
Lemma

The complement of a bipartite graph is \( \chi \)-perfect.

Proof:

- Note, that the class is hereditary.
- Show \( \chi(G) = \omega(G) \).
- So we have to prove: \( \kappa(G) = \alpha(G) \).
- By the theorem of König we get:
  - Take a maximum matching \( M \) with \( |M| = a \).
  - Assume that \( b \) nodes are not covered by \( M \).
  - Then we have: \( \alpha(G) = a + b \) and
  - \( \kappa(G) = a + b \).
Linegraphs of Bipartite Graphs

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

**Lemma**

*Linegraphs of bipartite graphs are \( \chi \)-perfect.*

**Proof:**
Linegraphs of Bipartite Graphs

Lemma

Linegraphs of bipartite graphs are $\chi$-perfect.

Proof:

- Note, that the class is hereditary.
Linegraphs of Bipartite Graphs

Lemma

Linegraphs of bipartite graphs are $\chi$-perfect.

Proof:

- Note, that the class is hereditary.
- Let $G$ bipartite graph and $H = L(G)$. 

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]
**Lemma**

*Linegraphs of bipartite graphs are $\chi$-perfect.*

**Proof:**

- Note, that the class is hereditary.
- Let $G$ bipartite graph and $H = L(G)$.
- Then we have by the construction of the linegraph:
Linegraphs of Bipartite Graphs

**Lemma**

Linegraphs of bipartite graphs are $\chi$-perfect.

Proof:

- Note, that the class is hereditary.
- Let $G$ bipartite graph and $H = L(G)$.
- Then we have by the construction of the linegraph:
  - $\omega(H) = \Delta(G)$ and
Linegraphs of Bipartite Graphs

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

Lemma

**Linegraphs of bipartite graphs are \( \chi \)-perfect.**

Proof:

- Note, that the class is hereditary.
- Let \( G \) bipartite graph and \( H = L(G) \).
- Then we have by the construction of the linegraph:
  - \( \omega(H) = \Delta(G) \) and
  - \( \chi(H) = \chi'(G) \).
Lemma

Linegraphs of bipartite graphs are $\chi$-perfect.

Proof:

- Note, that the class is hereditary.
- Let $G$ bipartite graph and $H = L(G)$.
- Then we have by the construction of the linegraph:
  - $\omega(H) = \Delta(G)$ and
  - $\chi(H) = \chi'(G)$.
- Furthermore is already known: $\chi'(G) = \Delta(G)$. 
Lemma

Linegraphs of bipartite graphs are $\chi$-perfect.

Proof:

- Note, that the class is hereditary.
- Let $G$ bipartite graph and $H = L(G)$.
- Then we have by the construction of the linegraph:
  - $\omega(H) = \Delta(G)$ and
  - $\chi(H) = \chi'(G)$.
- Furthermore is already known: $\chi'(G) = \Delta(G)$.
- Thus we have: $\omega(H) = \Delta(G) = \chi'(G) = \chi(H)$. 
Definition

A relation $\leq$ is called **partial ordering**, iff:
- Reflexive: $x \leq x$

$\omega(G) = \overline{\alpha}(G)$, $\alpha(G) = \overline{\omega}(G) = \beta_0(G)$, $\kappa(G) = \overline{\chi}(G)$
Definition

A relation $\leq$ is called a partial ordering, iff:

- Reflexive: $x \leq x$
- Transitive: $x \leq y \land y \leq z \implies x \leq z$

$$\omega(G) = \varnothing(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G)$$
Definition

A relation $\leq$ is called **partial ordering**, iff:

- Reflexive: $x \leq x$
- Transitive: $x \leq y \land y \leq z \implies x \leq z$
- Antisymmetric: $x \leq y \land y \leq x \implies x = y$

\[
\omega(G) = \alpha(G), \alpha(G) = \omega(G) = \beta_0(G), \kappa(G) = \chi(G)
\]
Definition

A relation $\leq$ is called partial ordering, iff:

- Reflexive: $x \leq x$
- Transitive: $x \leq y \land y \leq z \implies x \leq z$
- Antisymmetric: $x \leq y \land y \leq x \implies x = y$

$$\omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G)$$
A relation \( \leq \) is called partial ordering, iff:

- Reflexive: \( x \leq x \)
- Transitive: \( x \leq y \land y \leq z \implies x \leq z \)
- Antisymmetric: \( x \leq y \land y \leq x \implies x = y \)

Two elements are called comparable, if \( x \leq y \) oder \( y \leq x \).
Definition

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

A relation \( \leq \) is called partial ordering, iff:

- Reflexive: \( x \leq x \)
- Transitive: \( x \leq y \land y \leq z \implies x \leq z \)
- Antisymmetric: \( x \leq y \land y \leq x \implies x = y \)

- Two elements are called comparable, if \( x \leq y \) oder \( y \leq x \).
- A set of pairwise comparable elements is called a chain.
Definition

A relation $\leq$ is called a partial ordering, iff:
- Reflexive: $x \leq x$
- Transitive: $x \leq y \land y \leq z \implies x \leq z$
- Antisymmetric: $x \leq y \land y \leq x \implies x = y$

- Two elements are called comparable, if $x \leq y$ oder $y \leq x$.
- A set of pairwise comparable elements is called a chain.
- A set of pairwise not comparable elements is called an anti-chain.
Theorem 5.13 Comparability Graphs

Definition

A relation $\leq$ is called partial ordering, iff:

- Reflexive: $x \leq x$
- Transitive: $x \leq y \land y \leq z \implies x \leq z$
- Antisymmetric: $x \leq y \land y \leq x \implies x = y$

- Two elements are called comparable, if $x \leq y$ oder $y \leq x$.
- A set of pairwise comparable elements is called a chain.
- A set of pairwise not comparable elements is called an anti-chain.
- $y$ covers $x$ ($x \leq y$), if $x \leq y$ and $x \leq a \leq y \implies a \in \{x, y\}$.

\[ \omega(G) = \overline{\alpha}(G), \quad \alpha(G) = \overline{\omega}(G) = \beta_0(G), \quad \kappa(G) = \overline{\chi}(G) \]
Definition

A relation $\leq$ is called a partial ordering, iff:

- Reflexive: $x \leq x$
- Transitive: $x \leq y \land y \leq z \implies x \leq z$
- Antisymmetric: $x \leq y \land y \leq x \implies x = y$

- Two elements are called comparable, if $x \leq y$ oder $y \leq x$.
- A set of pairwise comparable elements is called a chain.
- A set of pairwise not comparable elements is called an anti-chain.
- $y$ covers $x$ ($x \leq y$), if $x \leq y$ and $x \leq a \leq y \implies a \in \{x, y\}$.
- This is called a PO-set

$\omega(G) = \overline{\alpha}(G)$, $\alpha(G) = \overline{\omega}(G) = \beta_0(G)$, $\kappa(G) = \overline{\chi}(G)$
Definition

A relation $\leq$ is called **partial ordering**, iff:

- **Reflexive:** $x \leq x$
- **Transitive:** $x \leq y \land y \leq z \implies x \leq z$
- **Antisymmetric:** $x \leq y \land y \leq x \implies x = y$

- Two elements are called comparable, if $x \leq y$ oder $y \leq x$.
- A set of pairwise comparable elements is called a chain.
- A set of pairwise not comparable elements is called an anti-chain.
- $y$ covers $x$ ($x \preceq y$), if $x \leq y$ and $x \leq a \leq y \implies a \in \{x, y\}$.
- This is called a PO-set
- The PO-set is denoted by $P_{\leq}$. 

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]
Definition

A graph $G = (V, E)$ is called **comparability graph**, if there is a partial ordering $\leq$ on $V$, with: $\{x, y\} \in E$ iff. $x$ and $y$ are comparable.

$$\omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G)$$
Definition

A graph $G = (V, E)$ is called **comparability graph**, if there is a partial ordering $\leq$ on $V$, with: $\{x, y\} \in E$ iff. $x$ and $y$ are comparable.

- Example: bipartite graphs.
Definition

A graph \( G = (V, E) \) is called comparability graph, if there is a partial ordering \( \leq \) on \( V \), with: \( \{x, y\} \in E \) iff. \( x \) and \( y \) are comparable.

- Example: bipartite graphs.
- Comparability graphs are transitive orientable.
A graph $G = (V, E)$ is called comparability graph, if there is a partial ordering $\leq$ on $V$, with: $\{x, y\} \in E$ iff. $x$ and $y$ are comparable.

- Example: bipartite graphs.
- Comparability graphs are transitive orientable.
- Example: transitive orientation of a bipartite graph.
Lemma

Let \( P \preceq \) be a PO-set. The maximal length of a chain is equal to the minimal number of anti-chains in which \( P \preceq \) may be partitioned.

\[
\omega(G) = \overline{\alpha}(G), \quad \alpha(G) = \omega(G) = \beta_0(G), \quad \kappa(G) = \overline{\chi}(G)
\]

\( \preceq \): Clear!
Statements

\[ \omega(G) = \bar{\alpha}(G), \alpha(G) = \bar{\omega}(G) = \beta_0(G), \kappa(G) = \bar{\chi}(G) \]

Lemma

Let \( P \subseteq \) be a PO-set. The maximal length of a chain is equal to the minimal number of anti-chains in which \( P \subseteq \) may be partitioned.

\( \leq \): Clear!

\( \geq \):
Lemma

Let $P \leq$ be a PO-set. The maximal length of a chain is equal to the minimal number of anti-chains in which $P \leq$ may be partitioned.

$\leq$: Clear!

$\geq$:

- $x$ minimal: $\forall a \in P \leq : a \leq x \implies a = x$
Lemma

Let \( P \preceq \) be a PO-set. The maximal length of a chain is equal to the minimal number of anti-chains in which \( P \preceq \) may be partitioned.

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

\( \preceq \): Clear!

\( \succeq \):
- \( x \) minimal: \( \forall a \in P \preceq : a \preceq x \implies a = x \)
- From this we may define a height function \( h(x) \).
ω(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G)

Lemma

Let \( P \subseteq \) be a PO-set. The maximal length of a chain is equal to the minimal number of anti-chains in which \( P \subseteq \) may be partitioned.

\( \leq \) : Clear!

\( \geq \) :

- \( x \) minimal: \( \forall a \in P \subseteq : a \leq x \implies a = x \)
- From this we may define a height function \( h(x) \).
- Let \( x = z_1 \leq z_1 \leq \ldots \leq z_{hy} = y \) be the longest chain of length \( h(y) \).
Statements

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

**Lemma**

Let \( P \preceq \) be a PO-set. The maximal length of a chain is equal to the minimal number of anti-chains in which \( P \preceq \) may be partitioned.

\( \preceq \) : Clear!

\( \succeq \) :

- \( x \) minimal: \( \forall a \in P \preceq : a \preceq x \implies a = x \)
- From this we may define a height function \( h(x) \).
- Let \( x = z_1 \preceq z_1 \preceq \ldots \preceq z_{h_y} = y \) be the longest chain of length \( h(y) \).
- The elements of the same height form an anti-chain.
\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

**Lemma**

Let \( P \subseteq \) be a PO-set. The maximal length of a chain is equal to the minimal number of anti-chains in which \( P \subseteq \) may be partitioned.

\( \subseteq \): Clear!

\( \supseteq \):

- \( x \) minimal: \( \forall a \in P \subseteq : a \leq x \implies a = x \)
- From this we may define a height function \( h(x) \).
- Let \( x = z_1 \leq z_1 \leq \ldots \leq z_{h(y)} = y \) be the longest chain of length \( h(y) \).
- The elements of the same height form an anti-chain.
- We have defined a partition of \( h(y) \) anti-chains.
Statements

Theorem

Comparability graphs are $\chi$-perfect.

Proof: clear!
 Statements

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

**Theorem**

*Comparability graphs are \( \chi \)-perfect.*

Proof: clear!

Note: \( \chi(G) \leq \omega(G) \) holds.
Statements

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

**Theorem**

*Comparability graphs are \( \chi \)-perfect.*

Proof: clear!

Note: \( \chi(G) \leq \omega(G) \) holds.

**Lemma**

*Let \( P \leq \) be a PO-set. The maximal length of an anti-chain is equal to the minimal number of chains in which \( P \leq \) may be partitioned.*
Theorem

Comparability graphs are $\chi$-perfect.

Proof: clear!

Note: $\chi(G) \leq \omega(G)$ holds.

Lemma

Let $P \leq$ be a PO-set. The maximal length of a anti-chain is equal to the minimal number of chains in which $P \leq$ may be partitioned.

Definition

A topological ordering of $G = (V, A)$ is an ordering of the nodes $\rho : V \mapsto \{1, 2, \ldots, n\}$ with: $(u, v) \in A \implies \rho(u) < \rho(v)$. 

\[
\omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G)
\]
**Theorem**

*Comparability graphs are $\chi$-perfect.*

**Proof:** clear!

**Note:** $\chi(G) \leq \omega(G)$ holds.

---

**Lemma**

*Let $P_{\leq}$ be a PO-set. The maximal length of a anti-chain is equal to the minimal number of chains in which $P_{\leq}$ may be partitioned.*

---

**Definition**

*A topological ordering of $G = (V, A)$ is an ordering of the nodes $\rho : V \mapsto \{1, 2, \ldots, n\}$ with: $(u, v) \in A \implies \rho(u) < \rho(v).$*

---

**Lemma**

*The colouring problem may be solved in linear time on comparability graphs by using a topological ordering.*
Statements

\[ \omega(G) = \overline{\alpha}(G), \ \alpha(G) = \overline{\omega}(G) = \beta_0(G), \ \kappa(G) = \overline{\chi}(G) \]

**Theorem**

*Interval graphs are \( \chi \)-perfect.*

**Theorem**

*The complement of an interval graph is a comparability graph.*
Statements

$\omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G)$

Theorem

*Interval graphs are χ-perfect.*

Theorem

*The complement of an interval graph is a comparability graph.*
**Statements**

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

**Theorem**

*Interval graphs are \( \chi \)-perfect.*

**Theorem**

*The complement of an interval graph is a comparability graph.*

**Theorem**

*For a graph \( G \) are the following statements equivalent:*

- \( G \) is an interval graph.
Statements

\[\omega(G) = \bar{\alpha}(G), \alpha(G) = \bar{\omega}(G) = \beta_0(G), \kappa(G) = \bar{\chi}(G)\]

**Theorem**

*Interval graphs are \(\chi\)-perfect.*

**Theorem**

*The complement of an interval graph is a comparability graph.*

**Theorem**

*For a graph \(G\) are the following statements equivalent:*

- \(G\) is an interval graph.
- \(G\) contains no induced \(C_4\) and \(\bar{G}\) is a comparability graph.*
Statements

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

**Theorem**

*Interval graphs are \( \chi \)-perfect.*

**Theorem**

*The complement of an interval graph is a comparability graph.*

**Theorem**

*For a graph \( G \) are the following statements equivalent:*

- \( G \) is an interval graph.
- \( G \) contains no induced \( C_4 \) and \( \overline{G} \) is a comparability graph.
- The maximal cliques of \( G \) may be ordered such that, the cliques which have a common node, follow in the ordering each other.*
First Observations

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

**Theorem**

The disjoint union of \( \chi \)-perfect graphs is a \( \chi \)-perfect graph.
First Observations

\[ \omega(G) = \overline{\alpha}(G), \ \alpha(G) = \overline{\omega}(G) = \beta_0(G), \ \kappa(G) = \overline{\chi}(G) \]

**Theorem**

The disjoint union of \( \chi \)-perfect graphs is a \( \chi \)-perfect graph.

**Theorem**

The identification of two \( \chi \)-perfect graphs at a clique gives a \( \chi \)-perfect graph.
First Observations

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

**Theorem**

The disjoint union of \( \chi \)-perfect graphs is a \( \chi \)-perfect graph.

**Theorem**

The identification of two \( \chi \)-perfect graphs at a clique gives a \( \chi \)-perfect graph.

**Theorem**

A graph \( G \) is \( \chi \)-perfect, iff in all induced subgraphs exists an independent set, which hits all maximum-cliques: \( \forall H \subseteq G : \exists I : \omega(H - I) \leq \omega(H) - 1 \) and \( I \) is an independent set.
A graph $G$ is $\chi$-perfect, iff in all induced subgraphs exists an independent set, which hits all maximum-cliques: $\forall H \subset G : \exists I : \omega(H - I) \leq \omega(H) - 1$. 

Proof:

$\omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G)$
A graph $G$ is $\chi$-perfect, iff in all induced subgraphs exists an independent set, which hits all maximum-cliques: $\forall H \subset G : \exists I : \omega(H - I) \leq \omega(H) - 1$.

Proof:

$\implies$ : 

Because $\chi(G) = \omega(G)$ holds,
Proof

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

Theorem

A graph \( G \) is \( \chi \)-perfect, iff in all induced subgraphs exists an independent set, which hits all maximum-cliques: \( \forall H \subset G : \exists I : \omega(H - I) \leq \omega(H) - 1 \).

Proof:

\[ \implies : \]

- Because \( \chi(G) = \omega(G) \) holds,
- will each colour-class hit all maximum-cliques.
**Theorem**

A graph $G$ is $\chi$-perfect, iff in all induced subgraphs exists an independent set, which hits all maximum-cliques: $\forall H \subset G : \exists I : \omega(H - I) \leq \omega(H) - 1$.

**Proof:**

\[
\begin{align*}
\Rightarrow & \quad : \\
& \quad \because \chi(G) = \omega(G) \text{ holds,} \\
& \quad \because \text{will each colour-class hit all maximum-cliques.}
\end{align*}
\]
A graph $G$ is $\chi$-perfect, iff in all induced subgraphs exists an independent set, which hits all maximum-cliques: $\forall H \subset G : \exists I : \omega(H - I) \leq \omega(H) - 1$.

Proof:

$\implies$:

- Because $\chi(G) = \omega(G)$ holds,
- will each colour-class hit all maximum-cliques.

$\impliedby$:

- We may show by induction over $|V(H)|$:

$$\chi(H) \leq \chi(H - I) + 1$$
Theorem

A graph $G$ is $\chi$-perfect, iff in all induced subgraphs exists an independent set, which hits all maximum-cliques: $\forall H \subset G : \exists I : \omega(H - I) \leq \omega(H) - 1$.

Proof:

$\implies$:

- Because $\chi(G) = \omega(G)$ holds,
- will each colour-class hit all maximum-cliques.

$\impliedby$:

- We may show by induction over $|V(H)|$:

$$\chi(H) \leq \chi(H - I) + 1 \implies \omega(H - I) + 1$$
Theorem

A graph $G$ is $\chi$-perfect, iff in all induced subgraphs exists an independent set, which hits all maximum-cliques: $\forall H \subset G : \exists I : \omega(H - I) \leq \omega(H) - 1$.

Proof:

$\implies$:

- Because $\chi(G) = \omega(G)$ holds,
- will each colour-class hit all maximum-cliques.

$\impliedby$:

- We may show by induction over $|V(H)|$:

$$\chi(H) \leq \chi(H - I) + 1 \overset{1. V.}{=} \omega(H - I) + 1 \leq \omega(H).$$
Strong perfect Graphs

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

**Definition**

A graph \( G = (V, E) \) is called strong perfect, iff for each node-induced subgraph exists an independent set, which hits all maximal cliques.
Strong perfect Graphs

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

**Definition**

A graph \( G = (V, E) \) is called strong perfect, iff for each node-induced subgraph exists an independent set, which hits all maximal cliques.

**Theorem**

A *strong perfect graph is also perfect.*
Strong perfect Graphs

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

**Definition**

A graph \( G = (V, E) \) is called strong perfect, iff for each node-induced subgraph exists an independent set, which hits all maximal cliques.

**Theorem**

A strong perfect graph is also perfect.

**Theorem**

The problems for \( \chi(G), \alpha(G), \omega(G), \kappa(G) \) are on \( \chi \)-perfect graphs solvable in polynomial time.
Strong perfect Graphs

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \chi(G) \]

**Definition**

A graph \( G = (V, E) \) is called strong perfect, iff for each node-induced subgraph exists an independent set, which hits all maximal cliques.

**Theorem**

*strong perfect graph is also perfect.*

**Theorem**

*The problems for \( \chi(G), \alpha(G), \omega(G), \kappa(G) \) are on \( \chi \)-perfect graphs solvable in polynomial time.*

Note: Proof uses the Ellipsoid Method.
The following statements are equivalent for graphs $G = (V, E)$:

1. $G$ is $\chi$-perfect.
2. $G$ is $\alpha$-perfect
3. For all node-induced subgraphs $H = (V', E')$ of $G$ holds:
   $$\alpha(H) \cdot \omega(H) \geq |V'|.$$
### Statements

\[ \omega(G) = \overline{\alpha}(G), \ \alpha(G) = \overline{\omega}(G) = \beta_0(G), \ \kappa(G) = \overline{\chi}(G) \]

**Theorem**

The following statements are equivalent for graphs \( G = (V, E) \):

1. \( G \) is \( \chi \)-perfect.
2. \( G \) is \( \alpha \)-perfect
3. For all node-induced subgraphs \( H = (V', E') \) of \( G \) holds:
   \[ \alpha(H) \cdot \omega(H) \geq |V'|. \]

**Theorem**

Perfect Graphs are closed under complement.
Lemma

If a node \( x \) of a \( \chi \)-perfect graph \( G \) is substituted by a \( \chi \)-perfect graph \( H \), then we get a \( \chi \)-perfect graph \( G_H \).

Proof:

- Construct an independent set \( I \), which hits all maximum cliques.
Lemma

If a node $x$ of a $\chi$-perfect graph $G$ is substituted by a $\chi$-perfect graph $H$, then we get a $\chi$-perfect graph $G_H$.

Proof:

- Construct an independent set $I$, which hits all maximum cliques.
- Colour $G$ with $\chi(G)$ colours.
Lemma

If a node $x$ of a $\chi$-perfect graph $G$ is substituted by a $\chi$-perfect graph $H$, then we get a $\chi$-perfect graph $G_H$.

Proof:

- Construct an independent set $I$, which hits all maximum cliques.
- Colour $G$ with $\chi(G)$ colours.
- Let $I_x$ be the set of nodes with the same colour as $x$. 

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]
Statements II

Lemma

If a node \( x \) of a \( \chi \)-perfect graph \( G \) is substituted by a \( \chi \)-perfect graph \( H \), then we get a \( \chi \)-perfect graph \( G_H \).

Proof:

- Construct an independent set \( I \), which hits all maximum cliques.
- Colour \( G \) with \( \chi(G) \) colours.
- Let \( I_x \) be the set of nodes with the same colour as \( x \).
- Let \( I_H \) be an independent set in \( H \), which hits all maximum-Cliques in \( H \).
Lemma

If a node $x$ of a $\chi$-perfect graph $G$ is substituted by a $\chi$-perfect graph $H$, then we get a $\chi$-perfect graph $G_H$.

Proof:

- Construct an independent set $I$, which hits all maximum cliques.
- Colour $G$ with $\chi(G)$ colours.
- Let $I_x$ be the set of nodes with the same colour as $x$.
- Let $I_H$ be an independent set in $H$, which hits all maximum-Cliques in $H$.
- Let: $I = I_x \setminus \{x\} \cup I_H$
Lemma

If a node $x$ of a $\chi$-perfect graph $G$ is substituted by a $\chi$-perfect graph $H$, then we get a $\chi$-perfect graph $G_H$.

Proof:

- Construct an independent set $I$, which hits all maximum cliques.
- Colour $G$ with $\chi(G)$ colours.
- Let $I_x$ be the set of nodes with the same colour as $x$.
- Let $I_H$ be an independent set in $H$, which hits all maximum-Cliques in $H$.
- Let: $I = I_x \setminus \{x\} \cup I_H$
- Let $C$ be a maximum-clique in $G_H$. 

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]
Statements II

\[ \omega(G) = \alpha(G), \quad \alpha(G) = \bar{\omega}(G) = \beta_0(G), \quad \kappa(G) = \chi(G) \]

**Lemma**

If a node \( x \) of a \( \chi \)-perfect graph \( G \) is substituted by a \( \chi \)-perfect graph \( H \), then we get a \( \chi \)-perfect graph \( G_H \).

**Proof:**

- Construct an independent set \( I \), which hits all maximum cliques.
- Colour \( G \) with \( \chi(G) \) colours.
- Let \( I_x \) be the set of nodes with the same colour as \( x \).
- Let \( I_H \) be an independent set in \( H \), which hits all maximum-Cliques in \( H \).
- Let: \( I = I_x \setminus \{x\} \cup I_H \)
- Let \( C \) be a maximum-clique in \( G_H \).
  - If \( C \cap V(H) = \emptyset \) holds, then is \( C \) in \( G - x \) and
Lemma

If a node \( x \) of a \( \chi \)-perfect graph \( G \) is substituted by a \( \chi \)-perfect graph \( H \), then we get a \( \chi \)-perfect graph \( G_H \).

Proof:

- Construct an independent set \( I \), which hits all maximum cliques.
- Colour \( G \) with \( \chi(G) \) colours.
- Let \( I_x \) be the set of nodes with the same colour as \( x \).
- Let \( I_H \) be an independent set in \( H \), which hits all maximum-Cliques in \( H \).
- Let: \( I = I_x \setminus \{x\} \cup I_H \)
- Let \( C \) be a maximum-clique in \( G_H \).
  - If \( C \cap V(H) = \emptyset \) holds, then is \( C \) in \( G - x \) and
  - because \( \omega(G) \geq \chi(G) \) holds, we get \( C \cap I_x \neq \emptyset \).
Lemma

If a node $x$ of a $\chi$-perfect graph $G$ is substituted by a $\chi$-perfect graph $H$, then we get a $\chi$-perfect graph $G_H$.

Proof:

- Construct an independent set $I$, which hits all maximum cliques.
- Colour $G$ with $\chi(G)$ colours.
- Let $I_x$ be the set of nodes with the same colour as $x$.
- Let $I_H$ be an independent set in $H$, which hits all maximum-Cliques in $H$.
- Let: $I = I_x \setminus \{x\} \cup I_H$
- Let $C$ be a maximum-clique in $G_H$.
  - If $C \cap V(H) = \emptyset$ holds, then is $C$ in $G - x$ and
  - because $\omega(G) \geq \chi(G)$ holds, we get $C \cap I_x \neq \emptyset$.
  - If $C \cap V(H) \neq \emptyset$, than contains $C$ a maximum-clique of $H$.
Statements II

\[ \omega(G) = \overline{\alpha}(G), \, \alpha(G) = \overline{\omega}(G) = \beta_0(G), \, \kappa(G) = \overline{\chi}(G) \]

**Lemma**

*If a node \( x \) of a \( \chi \)-perfect graph \( G \) is substituted by a \( \chi \)-perfect graph \( H \), then we get a \( \chi \)-perfect graph \( G_H \).*

**Proof:**

- Construct an independent set \( I \), which hits all maximum cliques.
- Colour \( G \) with \( \chi(G) \) colours.
- Let \( I_x \) be the set of nodes with the same colour as \( x \).
- Let \( I_H \) be an independent set in \( H \), which hits all maximum-Cliques in \( H \).
- Let: \( I = I_x \setminus \{x\} \cup I_H \)
- Let \( C \) be a maximum-clique in \( G_H \).
  - If \( C \cap V(H) = \emptyset \) holds, then is \( C \) in \( G - x \) and
  - because \( \omega(G) \geq \chi(G) \) holds, we get \( C \cap I_x \neq \emptyset \).
  - If \( C \cap V(H) \neq \emptyset \), than contains \( C \) a maximum-clique of \( H \)
  - and therefore hits \( I_H \) also \( C \).
Theorem

If a node \( x \) of a \( \alpha \)-perfect graph \( G \) is substituted by an independent set \( S \), then we get a \( \alpha \)-perfect graph \( G_S \).

It is sufficient to add just one node \( y \) as a copy of \( x \).
Lemma

If a node $x$ of a $\alpha$-perfect graph $G$ is substituted by an independent set $S$, then we get a $\alpha$-perfect graph $G_S$.

- It is sufficient to add just one node $y$ as a copy of $x$.
- We consider two cases:
Lemma

If a node \( x \) of a \( \alpha \)-perfect graph \( G \) is substituted by an independent set \( S \), then we get a \( \alpha \)-perfect graph \( G_S \).

- It is sufficient to add just one node \( y \) as a copy of \( x \).
- We consider two cases:
  - \( x \) is in an independent set \( S \) of size \( \alpha(G) \).
Lemma

If a node $x$ of a $\alpha$-perfect graph $G$ is substituted by an independent set $S$, then we get a $\alpha$-perfect graph $G_S$.

- It is sufficient to add just one node $y$ as a copy of $x$.
- We consider two cases:
  - $x$ is in an independent set $S$ of size $\alpha(G)$.
  - $x$ is not in an independent set $S$ of size $\alpha(G)$. 

\[ \omega(G) = \bar{\alpha}(G), \alpha(G) = \bar{\omega}(G) = \beta_0(G), \kappa(G) = \bar{\chi}(G) \]
Statements II

- Let $\mathcal{K}$ be a clique cover of $G$ with $|\mathcal{K}| = \kappa(G) = \alpha(G)$.
Let $\mathcal{K}$ be a clique cover of $G$ with $|\mathcal{K}| = \kappa(G) = \alpha(G)$.

$x$ is in an independent set $S$ of size $\alpha(G)$.
Statements II

- Let $\mathcal{K}$ be a clique cover of $G$ with $|\mathcal{K}| = \kappa(G) = \alpha(G)$.
- $x$ is in an independent set $S$ of size $\alpha(G)$.
  - Thus $S \cup \{y\}$ is an independent set and

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]
Let $\mathcal{K}$ be a clique cover of $G$ with $|\mathcal{K}| = \kappa(G) = \alpha(G)$.

$x$ is in an independent set $S$ of size $\alpha(G)$.

Thus $S \cup \{y\}$ is an independent set and

$\alpha(G_{\{y\}}) = \alpha(G) + 1$ holds.
Let $\mathcal{K}$ be a clique cover of $G$ with $|\mathcal{K}| = \kappa(G) = \alpha(G)$.

$x$ is in an independent set $S$ of size $\alpha(G)$.

Thus $S \cup \{y\}$ is an independent set and

$\alpha(G_{\{y\}}) = \alpha(G) + 1$ holds.

Because $\mathcal{K} \cup \{y\}$ is a clique cover of $G_{\{y\}}$, we get:

$\omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G)$
Let $\mathcal{K}$ be a clique cover of $G$ with $|\mathcal{K}| = \kappa(G) = \alpha(G)$.

$x$ is in an independent set $S$ of size $\alpha(G)$.

Thus $S \cup \{y\}$ is an independent set and

$\alpha(G_{\{y\}}) = \alpha(G) + 1$ holds.

Because $\mathcal{K} \cup \{y\}$ is a clique cover of $G_{\{y\}}$, we get:

$\kappa(G_{\{y\}}) \leq \kappa(G) + 1 = \alpha(G) + 1 = \alpha(G_{\{y\}}) \leq \kappa(G_{\{y\}})$. 

$\omega(G) = \overline{\omega}(G), \alpha(G) = \overline{\alpha}(G) = \beta_0(G), \kappa(G) = \overline{\kappa}(G)$
Let \( \mathcal{K} \) be a clique cover of \( G \) with \( |\mathcal{K}| = \kappa(G) = \alpha(G) \).

\( x \) is not in an independent set \( S \) of size \( \alpha(G) \).

Thus we have \( \alpha(G_{\{y\}}) = \alpha(G) \).
Statements II

- Let $\mathcal{K}$ be a clique cover of $G$ with $|\mathcal{K}| = \kappa(G) = \alpha(G)$.

- $x$ is not in an independent set $S$ of size $\alpha(G)$.
  - Thus we have $\alpha(G_{\{y\}}) = \alpha(G)$.
  - Because of $\kappa(G) = \alpha(G)$ each clique from $\mathcal{K}$ hits each maximum independent set.

$$\omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G)$$
Statements II

- Let $\mathcal{K}$ be a clique cover of $G$ with $|\mathcal{K}| = \kappa(G) = \alpha(G)$.
- $x$ is not in an independent set $S$ of size $\alpha(G)$.
  - Thus we have $\alpha(G \setminus \{y\}) = \alpha(G)$.
  - Because of $\kappa(G) = \alpha(G)$ each clique from $\mathcal{K}$ hits each maximum independent set.
  - Therefore hits $K_x$ (the clique, which contains $x$) each maximum independent set precisely once.
Let $\mathcal{K}$ be a clique cover of $G$ with $|\mathcal{K}| = \kappa(G) = \alpha(G)$.

$x$ is not in an independent set $S$ of size $\alpha(G)$.

- Thus we have $\alpha(G_{\{y\}}) = \alpha(G)$.
- Because of $\kappa(G) = \alpha(G)$ each clique from $\mathcal{K}$ hits each maximum independent set.
- Therefore hits $K_x$ (the clique, which contains $x$) each maximum independent set precisely once.
- And $D = K_x \setminus \{x\}$ hits each maximum independent set precisely once.

$\omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G)$
Let \( \mathcal{K} \) be a clique cover of \( G \) with 
\[ |\mathcal{K}| = \kappa(G) = \alpha(G). \]

\( x \) is not in an independent set \( S \) of size \( \alpha(G) \).

- Thus we have \( \alpha(G_{\{y\}}) = \alpha(G) \).
- Because of \( \kappa(G) = \alpha(G) \) each clique from \( \mathcal{K} \) hits each maximum independent set.
- Therefore hits \( K_x \) (the clique, which contains \( x \)) each maximum independent set precisely once.
- And \( D = K_x \setminus \{x\} \) hits each maximum independent set precisely once.
- Thus we get: \( \alpha(G[V \setminus D]) = \alpha(G) - 1. \)

\( \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \)
Let $\mathcal{K}$ be a clique cover of $G$ with $|\mathcal{K}| = \kappa(G) = \alpha(G)$.

- $x$ is not in an independent set $S$ of size $\alpha(G)$.
  - Thus we have $\alpha(G_{\{y\}}) = \alpha(G)$.
  - Because of $\kappa(G) = \alpha(G)$ each clique from $\mathcal{K}$ hits each maximum independent set.
  - Therefore hits $K_x$ (the clique, which contains $x$) each maximum independent set precisely once.
  - And $D = K_x \setminus \{x\}$ hits each maximum independent set precisely once.
  - Thus we get: $\alpha(G[V \setminus D]) = \alpha(G) - 1$.
  - By induction we get: $\kappa(G[V \setminus D]) = \alpha(G[V \setminus D]) = \alpha(G) - 1 = \alpha(G_{\{y\}}) - 1$. 

\[
\omega(G) = \overline{\alpha}(G), \quad \alpha(G) = \overline{\omega}(G) = \beta_0(G), \quad \kappa(G) = \overline{\chi}(G)
\]
Let $\mathcal{K}$ be a clique cover of $G$ with $|\mathcal{K}| = \kappa(G) = \alpha(G)$.

$x$ is not in an independent set $S$ of size $\alpha(G)$.

- Thus we have $\alpha(G_{\{y\}}) = \alpha(G)$.
- Because of $\kappa(G) = \alpha(G)$ each clique from $\mathcal{K}$ hits each maximum independent set.
- Therefore hits $K_x$ (the clique, which contains $x$) each maximum independent set precisely once.
- And $D = K_x \setminus \{x\}$ hits each maximum independent set precisely once.
- Thus we get: $\alpha(G[V \setminus D]) = \alpha(G) - 1$.
- By induction we get: $\kappa(G[V \setminus D]) = \alpha(G[V \setminus D]) = \alpha(G) - 1 = \alpha(G_{\{y\}}) - 1$.
- Thus there is a clique cover of $G[V \setminus D]$ of size $\alpha(G_{\{y\}}) - 1$. 

$$\omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G)$$
Let $\mathcal{K}$ be a clique cover of $G$ with $|\mathcal{K}| = \kappa(G) = \alpha(G)$.

- $x$ is not in an independent set $S$ of size $\alpha(G)$.
  - Thus we have $\alpha(G \setminus \{y\}) = \alpha(G)$.
  - Because of $\kappa(G) = \alpha(G)$ each clique from $\mathcal{K}$ hits each maximum independent set.
  - Therefore hits $K_x$ (the clique, which contains $x$) each maximum independent set precisely once.
  - And $D = K_x \setminus \{x\}$ hits each maximum independent set precisely once.
  - Thus we get: $\alpha(G[V \setminus D]) = \alpha(G) - 1$.
  - By induction we get: $\kappa(G[V \setminus D]) = \alpha(G[V \setminus D]) = \alpha(G) - 1 = \alpha(G \setminus \{y\}) - 1$.
  - Thus there is a clique cover of $G[V \setminus D]$ of size $\alpha(G \setminus \{y\}) - 1$.
  - Finally we get $\kappa(G \setminus \{y\}) = \alpha(G \setminus \{y\})$ (Covering: $D \cup \{y\}$).
Theorem (Lovász)

The complement of a perfect graph is perfect.

Proof (we will show that $\alpha$-perfect induces $\chi$-perfect):

- Let $G$ be a $\alpha$-perfect graph.
Theorem (Lovász)

The complement of a perfect graph is perfect.

Proof (we will show that \( \alpha \)-perfect induces \( \chi \)-perfect):

- Let \( G \) be a \( \alpha \)-perfect graph.
- We will use induction over \( n = |V(G)| \).
Theorem (Lovász)

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \chi(G) \]

**The complement of a perfect graph is perfect.**

Proof (we will show that \( \alpha \)-perfect induces \( \chi \)-perfect):

- Let \( G \) be a \( \alpha \)-perfect graph.
- We will use induction over \( n = |V(G)| \).
- The statement holds clearly for \( n \leq 3 \). Let \( n \geq 4 \).
Theorem (Lovász)

The complement of a perfect graph is perfect.

Proof (we will show that $\alpha$-perfect induces $\chi$-perfect):

- Let $G$ be a $\alpha$-perfect graph.
- We will use induction over $n = |V(G)|$.
- The statement holds clearly for $n \leq 3$. Let $n \geq 4$.
- For all induces real subgraphs of $G$ holds the statement.
Theorem (Lovász)

The complement of a perfect graph is perfect.

Proof (we will show that $\alpha$-perfect induces $\chi$-perfect):

- Let $G$ be a $\alpha$-perfect graph.
- We will use induction over $n = |V(G)|$.
- The statement holds clearly for $n \leq 3$. Let $n \geq 4$.
- For all induces real subgraphs of $G$ holds the statement.
- Thus we have to show $\chi(G) \leq \omega(G)$.
Statements III

\[ \omega(G) = \overline{\alpha}(G), \ \alpha(G) = \overline{\omega}(G) = \beta_0(G), \ \kappa(G) = \overline{\chi}(G) \]

**Theorem (Lovász)**

*The complement of a perfect graph is perfect.*

Proof (we will show that \( \alpha \)-perfect induces \( \chi \)-perfect):

- Let \( G \) be a \( \alpha \)-perfect graph.
- We will use induction over \( n = |V(G)| \).
- The statement holds clearly for \( n \leq 3 \). Let \( n \geq 4 \).
- For all induces real subgraphs of \( G \) holds the statement.
- Thus we have to show \( \chi(G) \leq \omega(G) \).
- If \( G \) has an independent set \( S \), which hists all maximum cliques,
Theorem (Lovász)

The complement of a perfect graph is perfect.

Proof (we will show that $\alpha$-perfect induces $\chi$-perfect):

- Let $G$ be a $\alpha$-perfect graph.
- We will use induction over $n = |V(G)|$.
- The statement holds clearly for $n \leq 3$. Let $n \geq 4$.
- For all induces real subgraphs of $G$ holds the statement.
- Thus we have to show $\chi(G) \leq \omega(G)$.
- If $G$ has an independent set $S$, which hists all maximum cliques,
- then $\omega(G \setminus S) = \omega(G) - 1$ holds.
The complement of a perfect graph is perfect.

Proof (we will show that $\alpha$-perfect induces $\chi$-perfect):

- Let $G$ be a $\alpha$-perfect graph.
- We will use induction over $n = |V(G)|$.
- The statement holds clearly for $n \leq 3$. Let $n \geq 4$.
- For all induces real subgraphs of $G$ holds the statement.
- Thus we have to show $\chi(G) \leq \omega(G)$.
- If $G$ has an independent set $S$, which hist all maximum cliques,
- then $\omega(G \setminus S) = \omega(G) - 1$ holds.
- Thus we get: $\chi(G) \leq \chi(G \setminus S) + 1 = \omega(G \setminus S) + 1 \leq \omega(G)$. 

$$\omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G)$$
Theorem (Lovász)

The complement of a perfect graph is perfect.

Proof (we will show that \(\alpha\)-perfect induces \(\chi\)-perfect):

- Let \(G\) be a \(\alpha\)-perfect graph.
- We will use induction over \(n = |V(G)|\).
- The statement holds clearly for \(n \leq 3\). Let \(n \geq 4\).
- For all induces real subgraphs of \(G\) holds the statement.
- Thus we have to show \(\chi(G) \leq \omega(G)\).
- If \(G\) has an independent set \(S\), which hists all maximum cliques,
  then \(\omega(G \setminus S) = \omega(G) - 1\) holds.
- Thus we get: \(\chi(G) \leq \chi(G \setminus S) + 1 = \omega(G \setminus S) + 1 \leq \omega(G)\).
- Therefore we assume in the following, that \(G\) has not an independent set \(S\), which hists all maximum cliques.
Proof

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

- \( G \) has not an independent set \( S \), which hits all maximum cliques.
Proof

\[ \omega(G) = \overline{\alpha}(G) \]
\[ \alpha(G) = \overline{\omega}(G) = \beta_0(G) \]
\[ \kappa(G) = \overline{\chi}(G) \]

- G has not an independent set S, which hists all maximum cliques.
- For each independent set S holds: \( G \setminus S \) contains a clique \( C_S \), with \( C_S \cap S = \emptyset \) and \( |C_S| = \omega(G) \).
Proof

\( \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \)

- \( G \) has not an independent set \( S \), which lists all maximum cliques.
- For each independent set \( S \) holds: \( G \setminus S \) contains a clique \( C_S \), with \( C_S \cap S = \emptyset \) and \( |C_S| = \omega(G) \).
- Let \( S \) be the set of independent sets in \( G \).
Proof

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

- G has not an independent set S, which hits all maximum cliques.
- For each independent set S holds: G \ S contains a clique \( C_S \), with \( C_S \cap S = \emptyset \) and \( |C_S| = \omega(G) \).
- Let S be the set of independent sets in G.
- For \( v_i \in V(G) \) let \( h_i = |\{S \in S \mid v_i \in C_S\}|. \)
Proof

\[ \omega(G) = \overline{\alpha}(G), \ \alpha(G) = \overline{\omega}(G) = \beta_0(G), \ \kappa(G) = \overline{\chi}(G) \]

- \( G \) has not an independent set \( S \), which hists all maximum cliques.
- For each independent set \( S \) holds: \( G \setminus S \) contains a clique \( C_S \), with \( C_S \cap S = \emptyset \) and \( |C_S| = \omega(G) \).
- Let \( S \) be the set of independent sets in \( G \).
- For \( v_i \in V(G) \) let \( h_i = |\{S \in S \mid v_i \in C_S\}| \).
- We replace each node \( v_i \in V(G) \) by an independent set of size \( h_i \).
G has not an independent set $S$, which hits all maximum cliques.

For each independent set $S$ holds: $G \setminus S$ contains a clique $C_S$, with $C_S \cap S = \emptyset$ and $|C_S| = \omega(G)$.

Let $S$ be the set of independent sets in $G$.

For $v_i \in V(G)$ let $h_i = |\{S \in S \mid v_i \in C_S\}|$.

We replace each node $v_i \in V(G)$ by an independent set of size $h_i$.

This new graph $H$ is also $\alpha$-perfect.
Proof

\[ \omega(G) = \overline{\alpha}(G), \; \alpha(G) = \overline{\omega}(G) = \beta_0(G), \; \kappa(G) = \overline{\chi}(G) \]

- \( G \) has not an independent set \( S \), which hits all maximum cliques.
- For each independent set \( S \) holds: \( G \setminus S \) contains a clique \( C_S \), with \( C_S \cap S = \emptyset \) and \( |C_S| = \omega(G) \).
- Let \( S \) be the set of independent sets in \( G \).
- For \( v_i \in V(G) \) let \( h_i = |\{ S \in S \mid v_i \in C_S \}|. \)
- We replace each node \( v_i \in V(G) \) by an independent set of size \( h_i \).
- This new graph \( H \) is also \( \alpha \)-perfect.
- Furthermore we get:

\[ |V(H)| = \sum_{v_i \in V(G)} h_i \]
Proof

- \( G \) has not an independent set \( S \), which hists all maximum cliques.
- For each independent set \( S \) holds: \( G \setminus S \) contains a clique \( C_S \), with \( C_S \cap S = \emptyset \) and \( |C_S| = \omega(G) \).
- Let \( S \) be the set of independent sets in \( G \).
- For \( v_i \in V(G) \) let \( h_i = |\{S \in S \mid v_i \in C_S\}| \).
- We replace each node \( v_i \in V(G) \) by an independent set of size \( h_i \).
- This new graph \( H \) is also \( \alpha \)-perfect.
- Furthermore we get:

\[
|V(H)| = \sum_{v_i \in V(G)} h_i = \sum_{v_i \in V(G)} \sum_{S \in S} |v_i \cap C_S|
\]

\[
\omega(G) = \overline{\alpha}(G), \quad \alpha(G) = \overline{\omega}(G) = \beta_0(G), \quad \kappa(G) = \overline{\chi}(G)
\]
Proof

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

- \( G \) has not an independent set \( S \), which hits all maximum cliques.
- For each independent set \( S \) holds: \( G \setminus S \) contains a clique \( C_S \), with \( C_S \cap S = \emptyset \) and \( |C_S| = \omega(G) \).
- Let \( S \) be the set of independent sets in \( G \).
- For \( v_i \in V(G) \) let \( h_i = |\{ S \in S \mid v_i \in C_S \}| \).
- We replace each node \( v_i \in V(G) \) by an independent set of size \( h_i \).
- This new graph \( H \) is also \( \alpha \)-perfect.
- Furthermore we get:

\[
|V(H)| = \sum_{v_i \in V(G)} h_i = \sum_{v_i \in V(G)} \sum_{S \in S} |v_i \cap C_S| = \sum_{S \in S} \sum_{v_i \in V(G)} |v_i \cap C_S|
\]
Proof

\[ \omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G) \]

- \( G \) has not an independent set \( S \), which hints all maximum cliques.
- For each independent set \( S \) holds: \( G \setminus S \) contains a clique \( C_S \), with \( C_S \cap S = \emptyset \) and \( |C_S| = \omega(G) \).
- Let \( S \) be the set of independent sets in \( G \).
- For \( v_i \in V(G) \) let \( h_i = |\{ S \in S \mid v_i \in C_S \}| \).
- We replace each node \( v_i \in V(G) \) by an independent set of size \( h_i \).
- This new graph \( H \) is also \( \alpha \)-perfect.
- Furthermore we get:

\[
|V(H)| = \sum_{v_i \in V(G)} h_i \\
= \sum_{v_i \in V(G)} \sum_{S \in S} |v_i \cap C_S| \\
= \sum_{S \in S} \sum_{v_i \in V(G)} |v_i \cap C_S| \\
= \sum_{S \in S} |C_S|
\]
Proof

$\omega(G) = \overline{\alpha}(G), \alpha(G) = \overline{\omega}(G) = \beta_0(G), \kappa(G) = \overline{\chi}(G)$

- $G$ has not an independent set $S$, which hists all maximum cliques.
- For each independent set $S$ holds: $G \setminus S$ contains a clique $C_S$, with $C_S \cap S = \emptyset$ and $|C_S| = \omega(G)$.
- Let $S$ be the set of independent sets in $G$.
- For $v_i \in V(G)$ let $h_i = |\{S \in S \mid v_i \in C_S\}|$.
- We replace each node $v_i \in V(G)$ by an independent set of size $h_i$.
- This new graph $H$ is also $\alpha$-perfect.
- Furthermore we get:

$$|V(H)| = \sum_{v_i \in V(G)} h_i$$

$$= \sum_{v_i \in V(G)} \sum_{S \in S} |v_i \cap C_S|$$

$$= \sum_{S \in S} \sum_{v_i \in V(G)} |v_i \cap C_S|$$

$$= \sum_{S \in S} |C_S|$$

$$= \omega(G) \cdot |S|$$
Proof

- By Construction of $H$ we have $\omega(H) \leq \omega(G)$.
Proof

- By Construction of $H$ we have $\omega(H) \leq \omega(G)$.
- Then it holds (note in the following: $|T \cap C_S| \leq 1$ and $|S \cap C_S| = 0$):

  $$\alpha(H) = \max_{T \in S} \sum_{x_i \in T} h_i$$
**Proof**

- By Construction of $H$ we have $\omega(H) \leq \omega(G)$.
- Then it holds (note in the following: $|T \cap C_S| \leq 1$ and $|S \cap C_S| = 0$):

$$\alpha(H) = \max_{T \in S} \sum_{x_i \in T} h_i \quad = \max_{T \in S} \sum_{S \in S} |T \cap C_S|$$

$$h_i = |\{S \in S \mid v_i \in C_S\}|$$
Proof

- By Construction of $H$ we have $\omega(H) \leq \omega(G)$.
- Then it holds (note in the following: $|T \cap C_S| \leq 1$ and $|S \cap C_S| = 0$):

\[
\alpha(H) = \max_{T \in S} \sum_{x_i \in T} h_i
= \max_{T \in S} \sum_{S \in S} |T \cap C_S|
\leq |S| - 1
\]
**Proof**

- By Construction of $H$ we have $\omega(H) \leq \omega(G)$.
- Then it holds (note in the following: $|T \cap C_S| \leq 1$ and $|S \cap C_S| = 0$):
  
  $$\alpha(H) = \max_{T \in S} \sum_{x_i \in T} h_i$$
  $$= \max_{T \in S} \sum_{S \in S} |T \cap C_S|$$
  $$\leq |S| - 1$$

- Furthermore we get:
  
  $$\kappa(H) \geq \frac{|V(H)|}{\omega(H)}$$
Proof

- By Construction of $H$ we have $\omega(H) \leq \omega(G)$.
- Then it holds (note in the following: $|T \cap C_S| \leq 1$ and $|S \cap C_S| = 0$):

$$\alpha(H) = \max_{T \in S} \sum_{x_i \in T} h_i$$

$$= \max_{T \in S} \sum_{S \in S} |T \cap C_S|$$

$$\leq |S| - 1$$

- Furthermore we get:

$$\kappa(H) \geq \frac{|V(H)|}{\omega(H)} = \frac{|V(H)|}{\omega(G)}$$
Proof

- By Construction of $H$ we have $\omega(H) \leq \omega(G)$.
- Then it holds (note in the following: $|T \cap C_S| \leq 1$ and $|S \cap C_S| = 0$):

\[
\alpha(H) = \max_{T \in S} \sum_{x_i \in T} h_i = \max_{T \in S} \sum_{S \in S} |T \cap C_S| \leq |S| - 1
\]

- Furthermore we get:

\[
\kappa(H) \geq \frac{|V(H)|}{\omega(H)} = \frac{|V(H)|}{\omega(G)} = |S|.
\]
Proof

- By Construction of $H$ we have $\omega(H) \leq \omega(G)$.
- Then it holds (note in the following: $|T \cap C_S| \leq 1$ and $|S \cap C_S| = 0$):
  \[
  \alpha(H) = \max_{T \in S} \sum_{x_i \in T} h_i = \max_{T \in S} \sum_{S \in S} |T \cap C_S| \leq |S| - 1
  \]

- Furthermore we get:
  \[
  \kappa(H) \geq \frac{|V(H)|}{\omega(H)} = \frac{|V(H)|}{\omega(G)} = |S|.
  \]

- Thus we get the following contradiction:
  \[
  \kappa(H) \geq |S| > |S| - 1 \geq \alpha(H).
  \]
Definition

A graph $G = (V, E)$ is called minimal imperfect, iff it is not perfect ist and each node induced real subgraph is perfect.
Definition

A graph \( G = (V, E) \) is called minimal imperfect, iff it is not perfect ist and each node induced real subgraph is perfect.

Strong Perfect Graph Theorem

A minimal imperfect graph is either an odd cycle of length \( \geq 5 \) or its complement.
Definition

A graph $G = (V, E)$ is called minimal imperfect, iff it is not perfect ist and each node induced real subgraph is perfect.

Strong Perfect Graph Theorem

A minimal imperfect graph is either an odd cycle of length $\geq 5$ or its complement.

Theorem

*The Recognition of perfect graphs is in $\mathcal{P}$.***
Definition

A graph $G = (V, E)$ is called minimal imperfect, iff it is not perfect ist and each node induced real subgraph is perfect.

Strong Perfect Graph Theorem

A minimal imperfect graph is either an odd cycle of length $\geq 5$ or its complement.

Theorem

*The Recognition of perfect graphs is in $P$.***
A graph $G$ is called chordal, iff it induces no $C_k$ for $k \geq 4$. 

Note: i.e. $G$ does not contain a $C_k$ as induced subgraph. 

Note: are sometimes also called triangulated.
A graph $G$ is called chordal, iff it induces no $C_k$ for $k \geq 4$.

Note: I.e. $G$ does not contain a $C_k$ as induced subgraph.
**Definition**

A graph $G$ is called chordal, iff it induces no $C_k$ for $k \geq 4$.

Note: I.e. $G$ does not contain a $C_k$ as induced subgraph.

Note: are sometimes also called triangulated.
A graph $G$ is called chordal, iff it induces no $C_k$ for $k \geq 4$.

Note: I.e. $G$ does not contain a $C_k$ as induced subgraph.
Note: are sometimes also called triangulated.
Examples:
- Intervall-graphs
A graph $G$ is called chordal, iff it induces no $C_k$ for $k \geq 4$.

Note: i.e. $G$ does not contain a $C_k$ as induced subgraph.
Note: are sometimes also called triangulated.
Examples:

- Intervall-graphs
- Maximal outer-planar graphs
A graph $G$ is called chordal, iff it induces no $C_k$ for $k \geq 4$.

Note: I.e. $G$ does not contain a $C_k$ as induced subgraph.

Note: are sometimes also called triangulated.

Examples:

- Intervall-graphs
- Maximal outer-planar graphs
- K-trees
A graph $G$ is chordal, iff each inclusion minimal separator is a clique.

Proof ($\implies$):

1. Let $S$ be an inclusion minimal separator.
2. $S$ separates $H_1$ and $H_2$.
3. All nodes from $S$ have neighbors in $H_1$ and $H_2$.
4. Let $u, v$ be from $S$.
5. There is shortest path $P_i$ from $u$ to $v$ in $H_i$.
6. Thus, there is a cycle given by $P_1$ and $P_2$.
7. There is an edge $\{u, v\}$.
Theorem

A graph $G$ is chordal, iff each inclusion minimal separator is a clique.

Proof ($\implies$):

Let $S$ be a inclusion minimal separator is a clique.
A graph $G$ is chordal, iff each inclusion minimal separator is a clique.

Proof ($\Rightarrow$):

- Let $S$ be a inclusion minimal separator is a clique.
- $S$ separates $H_1$ and $H_2$. 

Diagram:

```
       H_1
       /
      /  \
 S    /    S
       \
       H_2
```
Theorem

A graph $G$ is chordal, iff each inclusion minimal separator is a clique.

Proof ($\Rightarrow$):

- Let $S$ be a inclusion minimal separator is a clique.
- $S$ separates $H_1$ and $H_2$.
- All nodes from $S$ have neighbours in $H_1$ and $H_2$. 
Theorem

A graph $G$ is chordal, iff each inclusion minimal separator is a clique.

Proof ($\Rightarrow$):
- Let $S$ be a inclusion minimal separator is a clique.
- $S$ separates $H_1$ and $H_2$.
- All nodes from $S$ have neighbours in $H_1$ and $H_2$.
- Let $u, v$ be from $S$. 

Proof ($\Leftarrow$):
- Let $S$ be a inclusion minimal separator is a clique.
- All nodes from $S$ have neighbours in $H_1$ and $H_2$.
- Let $u, v$ be from $S$. 

Inclusion minimal separator $S$ separates $H_1$ and $H_2$. Nodes from $S$ have neighbours in $H_1$ and $H_2$.
Theorem

A graph $G$ is chordal, iff each inclusion minimal separator is a clique.

Proof ($\Rightarrow$):

- Let $S$ be a inclusion minimal separator is a clique.
- $S$ separates $H_1$ and $H_2$.
- All nodes from $S$ have neighbours in $H_1$ and $H_2$.
- Let $u, v$ be from $S$.
- There is shortest path $P_i$ from $u$ to $v$ in $H_i$. 

\[
\begin{align*}
H_1 & : e_1 \quad a_1 \quad z_1 \quad e_2 \\
S & : u \quad v \\
H_2 & : c_2 \quad a_2 \quad z_2 \quad c_1
\end{align*}
\]
A graph $G$ is chordal, iff each inclusion minimal separator is a clique.

Proof $(\implies)$:

- Let $S$ be a inclusion minimal separator is a clique.
- $S$ separates $H_1$ and $H_2$.
- All nodes from $S$ have neighbours in $H_1$ and $H_2$.
- Let $u, v$ be from $S$.
- There is shortest path $P_i$ from $u$ to $v$ in $H_i$.
- Thus three is a cycle given by $P_1$ and $P_2$. 
Statements

Theorem

A graph $G$ is chordal, iff each inclusion minimal separator is a clique.

Proof ($\implies$):

- Let $S$ be a inclusion minimal separator is a clique.
- $S$ separates $H_1$ and $H_2$.
- All nodes from $S$ have neighbours in $H_1$ and $H_2$.
- Let $u, v$ be from $S$.
- There is shortest path $P_i$ from $u$ to $v$ in $H_i$.
- Thus three is a cycle given by $P_1$ and $P_2$.
- There is an edges $\{u, v\}$.
A graph $G$ is chordal, iff each inclusion minimal separator is a clique.

Proof ($\Longleftrightarrow$):
A graph $G$ is chordal, iff each inclusion minimal separator is a clique.

Proof ($\iff$):
- Let $C$ be a cycle of length $\geq 4$. 
Statements

**Theorem**

A graph $G$ is chordal, iff each inclusion minimal separator is a clique.

Proof ($\iff$):

- Let $C$ be a cycle of length $\geq 4$.
- Let $u, v$ non-neighboured nodes in $C$. 
A graph $G$ is chordal, iff each inclusion minimal separator is a clique.

Proof ($\Longleftrightarrow$):

- Let $C$ be a cycle of length $\geq 4$.
- Let $u, v$ non-neighboured nodes in $C$.
- If $\{u, v\} \in E$, the statement holds.
Theorem

A graph $G$ is chordal, iff each inclusion minimal separator is a clique.

Proof ($\iff$):

- Let $C$ be a cycle of length $\geq 4$.
- Let $u, v$ non-neighboured nodes in $C$.
- If $\{u, v\} \in E$, the statement holds.
- On the other side:
A graph $G$ is chordal, iff each inclusion minimal separator is a clique.

Proof ($\iff$):

- Let $C$ be a cycle of length $\geq 4$.
- Let $u, v$ non-neighbour nodes in $C$.
- If $\{u, v\} \in E$, the statement holds.
- On the other side:
  - Let $S$ be a minimal separator for $u$ and $v$. 

\begin{align*}
&H_1 \\
&\begin{array}{c}
H_2 \\
S \\
\{a_1, a_2\} \\
\{u, v\}
\end{array}
\end{align*}


**Theorem**

A graph $G$ is chordal, iff each inclusion minimal separator is a clique.

Proof ($\iff$):

- Let $C$ be a cycle of length $\geq 4$.
- Let $u, v$ non-neighoured nodes in $C$.
- If $\{u, v\} \in E$, the statement holds.
- On the other side:
  - Let $S$ be a minimal separator for $u$ and $v$.
  - This separator is a clique.
**Theorem**

*A graph $G$ is chordal, iff each inclusion minimal separator is a clique.*

Proof ($\iff$):

- Let $C$ be a cycle of length $\geq 4$.
- Let $u, v$ non-neighboured nodes in $C$.
- If $\{u, v\} \in E$, the statement holds.
- On the other side:
  - Let $S$ be a minimal separator for $u$ and $v$.
  - This separator is a clique.
  - This contains two other nodes from $C$. 

![Diagram](image-url)
Theorem

A graph $G$ is chordal, iff each inclusion minimal separator is a clique.

Proof ($\iff$):

- Let $C$ be a cycle of length $\geq 4$.
- Let $u, v$ non-neighboured nodes in $C$.
- If $\{u, v\} \in E$, the statement holds.
- On the other side:
  - Let $S$ be a minimal separator for $u$ and $v$.
  - This separator is a clique.
  - This contains two other nodes from $C$.
  - These other nodes are connected.
**Theorem**

A graph $G$ is chordal, iff each inclusion minimal separator is a clique.

**Proof ($\iff$):**

- Let $C$ be a cycle of length $\geq 4$.
- Let $u, v$ non-neighboured nodes in $C$.
- If $\{u, v\} \in E$, the statement holds.
- On the other side:
  - Let $S$ be a minimal separator for $u$ and $v$.
  - This separator is a clique.
  - This contains two other nodes from $C$.
  - These other nodes are connected.
Definition

A node is called simplicial, iff all its neighbours induce a complete subgraph.
Simplicial Nodes

Definition
A node is called simplicial, iff all its neighbours induce a complete subgraph.

Theorem
Each Clique has a simplicial node and each chordal graph, who is not a clique, has two simplicial nodes, which are not connected.

- Proof by induction. (Statement holds for $|V| \leq 3$.)
Definition

A node is called simplicial, iff all its neighbours induce a complete subgraph.

Theorem

Each Clique has a simplicial node and each chordal graph, who is not a clique, has two simplicial nodes, which are not connected.

- Proof by induction. (Statement holds for $|V| \leq 3$.)
- Let $u, v$ be two non-neighbour nodes.
Simplicial Nodes

Definition
A node is called simplicial, iff all its neighbours induce a complete subgraph.

Theorem
Each Clique has a simplicial node and each chordal graph, who is not a clique, has two simplicial nodes, which are not connected.

Proof by induction. (Statement holds for $|V| \leq 3$.)
- Let $u, v$ be two non-neighboured nodes.
- Identify a minimal separator $S$ for $u, v$. 

\[
\begin{array}{c}
H_1 \\
S \\
H_2
\end{array}
\]

\( \Sigma = 0 \)
**Definition**

A node is called simplicial, iff all its neighbours induce a complete subgraph.

**Theorem**

*Each Clique has a simplicial node and each chordal graph, who is not a clique, has two simplicial nodes, which are not connected.*

- Proof by induction. (Statement holds for $|V| \leq 3$.)
- Let $u, v$ be two non-neighboured nodes.
- Identify a minimal separator $S$ for $u, v$.
- $G - S$ splits into components $H_i$, with $i \geq 2$. 
Simplicial Nodes

**Definition**
A node is called simplicial, iff all its neighbours induce a complete subgraph.

**Theorem**
Each Clique has a simplicial node and each chordal graph, who is not a clique, has two simplicial nodes, which are not connected.

- Proof by induction. (Statement holds for $|V| \leq 3$.)
- Let $u, v$ be two non-neighboured nodes.
- Identify a minimal separator $S$ for $u, v$.
- $G - S$ splits into components $H_i$, with $i \geq 2$.
- $S$ is a clique.
Definition

A node is called simplicial, iff all its neighbours induce a complete subgraph.

Theorem

Each Clique has a simplicial node and each chordal graph, who is not a clique, has two simplicial nodes, which are not connected.

Proof by induction. (Statement holds for $|V| \leq 3$.)

- Let $u, v$ be two non-neighboured nodes.
- Identify a minimal separator $S$ for $u, v$.
- $G - S$ splits into components $H_i$, with $i \geq 2$.
- $S$ is a clique.
- $H_i \cup S$ contains a simplicial node.
Simplicial Nodes

**Definition**

A node is called simplicial, iff all its neighbours induce a complete subgraph.

**Theorem**

*Each Clique has a simplicial node and each chordal graph, who is not a clique, has two simplicial nodes, which are not connected.*

- Proof by induction. (Statement holds for $|V| \leq 3$.)
- Let $u, v$ be two non-neighboured nodes.
- Identify a minimal separator $S$ for $u, v$.
- $G - S$ splits into components $H_i$, with $i \geq 2$.
- $S$ is a clique.
- $H_i \cup S$ contains a simplicial node.
- This node is also simplicial node in $G$. 
Theorem

Chordal graphs and their complements are perfect.

Proof (just using chordal graphs):
Theorem

Chordal graphs and their complements are perfect.

Proof (just using chordal graphs):

- By induction.
Statements

Theorem

Chordal graphs and their complements are perfect.

- Proof (just using chordal graphs):
  - By induction.
  - Let \( G \) be no clique.
Chordal graphs and their complements are perfect.

Proof (just using chordal graphs):
- By induction.
- Let $G$ be no clique.
- Then contains $G$ a separating clique $C$. 
Theorem

Chordal graphs and their complements are perfect.

Proof (just using chordal graphs):

- By induction.
- Let $G$ be no clique.
- Then contains $G$ a separating clique $C$.
- $G - C$ splits into components $H_i$, with $i \geq 2$. 
Theorem

*Chordal graphs and their complements are perfect.*

- Proof (just using chordal graphs):
  - By induction.
  - Let $G$ be no clique.
  - Then contains $G$ a separating clique $C$.
  - $G - C$ splits into components $H_i$, with $i \geq 2$.
  - $H_i \cup C$ are perfect.
Theorem

Chordal graphs and their complements are perfect.

Proof (just using chordal graphs):

- By induction.
- Let $G$ be no clique.
- Then contains $G$ a separating clique $C$.
- $G - C$ splits into components $H_i$, with $i \geq 2$.
- $H_i \cup C$ are perfect.
- Thus $G$ is perfect.
Theorem

Chordal graphs and their complements are perfect.

Proof (just using chordal graphs):

- By induction.
- Let $G$ be no clique.
- Then contains $G$ a separating clique $C$.
- $G - C$ splits into components $H_i$, with $i \geq 2$.
- $H_i \cup C$ are perfect.
- Thus $G$ is perfect.

Proof (using the complement of chordal graphs):
Theorem

**Chordal graphs and their complements are perfect.**

- **Proof (just using chordal graphs):**
  - By induction.
  - Let $G$ be no clique.
  - Then contains $G$ a separating clique $C$.
  - $G - C$ splits into components $H_i$, with $i \geq 2$.
  - $H_i \cup C$ are perfect.
  - Thus $G$ is perfect.

- **Proof (using the complement of chordal graphs):**
  - Identify clique in $G$, which hists all independent sets.
Chordal graphs and their complements are perfect.

Proof (just using chordal graphs):

- By induction.
- Let $G$ be no clique.
- Then contains $G$ a separating clique $C$.
- $G - C$ splits into components $H_i$, with $i \geq 2$.
- $H_i \cup C$ are perfect.

Thus $G$ is perfect.

Proof (using the complement of chordal graphs):

- Identify clique in $G$, which hists all independent sets.
- Choose simplicial node $s$, i.e. $C = \{s\} \cup \Gamma(s)$. 
Definition

Let $G = (V, E)$ be a graph with $|V| = n$. A total ordering $\rho : V \mapsto \{1, \ldots, n\}$ is called perfect node-elimination scheme, iff each node $v$ is a simplicial node in $G[\{u \in V \mid \rho(u) \geq \rho(v)\}]$. 

![Graph Diagram]
Definition

Let $G = (V, E)$ be a graph with $|V| = n$. A total ordering $\rho : V \mapsto \{1, \ldots, n\}$ is called perfect node-elimination scheme, iff each node $v$ is a simplicial node in $G[\{u \in V \mid \rho(u) \geq \rho(v)\}]$. 
Definition

Let $G = (V, E)$ be a graph with $|V| = n$. A total ordering $\rho : V \mapsto \{1, \ldots, n\}$ is called perfect node-elimination scheme, iff each node $v$ is a simplicial node in $G[\{u \in V \mid \rho(u) \geq \rho(v)\}]$. 
Definition

Let $G = (V, E)$ be a graph with $|V| = n$. A total ordering $\rho : V \mapsto \{1, \ldots, n\}$ is called perfect node-elimination scheme, iff each node $v$ is a simplicial node in $G[\{u \in V \mid \rho(u) \geq \rho(v)\}]$. 

\[ \Sigma = 0 \]
Definition

Let $G = (V, E)$ be a graph with $|V| = n$. A total ordering $\rho : V \mapsto \{1, \ldots, n\}$ is called perfect node-elimination scheme, iff each node $v$ is a simplicial node in $G[\{u \in V \mid \rho(u) \geq \rho(v)\}]$. 

Definition

Let $G = (V, E)$ be a graph with $|V| = n$. A total ordering $\rho : V \mapsto \{1, \ldots, n\}$ is called perfect node-elimination scheme, iff each node $v$ is a simplicial node in $G[\{u \in V \mid \rho(u) \geq \rho(v)\}]$. 
Definition

Let $G = (V, E)$ be a graph with $|V| = n$. A total ordering $\rho : V \mapsto \{1, \ldots, n\}$ is called a perfect node-elimination scheme, iff each node $v$ is a simplicial node in $G[\{u \in V \mid \rho(u) \geq \rho(v)\}]$. 
Definition

Let $G = (V, E)$ be a graph with $|V| = n$. A total ordering $\rho : V \mapsto \{1, \ldots, n\}$ is called perfect node-elimination scheme, iff each node $v$ is a simplicial node in $G[\{u \in V \mid \rho(u) \geq \rho(v)\}]$. 
Definition

Let $G = (V, E)$ be a graph with $|V| = n$. A total ordering $\rho : V \mapsto \{1, \ldots, n\}$ is called perfect node-elimination scheme, iff each node $v$ is a simplicial node in $G[\{u \in V \mid \rho(u) \geq \rho(v)\}]$. 
Definition

Let $G = (V, E)$ be a graph with $|V| = n$. A total ordering $\rho : V \mapsto \{1, \ldots, n\}$ is called perfect node-elimination scheme, iff each node $v$ is a simplicial node in $G[\{u \in V \mid \rho(u) \geq \rho(v)\}]$. 

$$
\begin{align*}
\text{Definition} \\
\text{Let } G = (V, E) \text{ be a graph with } |V| = n. \text{ A total ordering } \rho : V \mapsto \{1, \ldots, n\} \text{ is called perfect node-elimination scheme, iff each node } v \text{ is a simplicial node in } G[\{u \in V \mid \rho(u) \geq \rho(v)\}].
\end{align*}
$$
Definition

Let $G = (V, E)$ be a graph with $|V| = n$. A total ordering $\rho : V \mapsto \{1, \ldots, n\}$ is called perfect node-elimination scheme, iff each node $v$ is a simplicial node in $G[\{u \in V \mid \rho(u) \geq \rho(v)\}]$. 
Definition

Let $G = (V, E)$ be a graph with $|V| = n$. A total ordering $\rho : V \mapsto \{1, \ldots, n\}$ is called perfect node-elimination scheme, iff each node $v$ is a simplicial node in $G[\{u \in V | \rho(u) \geq \rho(v)\}]$. 
Definition

Let $G = (V, E)$ be a graph with $|V| = n$. A total ordering $\rho : V \mapsto \{1, \ldots, n\}$ is called perfect node-elimination scheme, iff each node $v$ is a simplicial node in $G[\{u \in V \mid \rho(u) \geq \rho(v)\}]$. 
Definition

Let $G = (V, E)$ be a graph with $|V| = n$. A total ordering $\rho : V \mapsto \{1, \ldots, n\}$ is called perfect node-elimination scheme, iff each node $v$ is a simplicial node in $G[\{u \in V \mid \rho(u) \geq \rho(v)\}]$. 
Definition

Let $G = (V, E)$ be a graph with $|V| = n$. A total ordering $\rho : V \mapsto \{1, \ldots, n\}$ is called perfect node-elimination scheme, iff each node $v$ is a simplicial node in $G[\{u \in V \mid \rho(u) \geq \rho(v)\}]$. 
Definition

Let $G = (V, E)$ be a graph with $|V| = n$. A total ordering $\rho : V \mapsto \{1, \ldots, n\}$ is called perfect node-elimination scheme, iff each node $v$ is a simplicial node in $G[\{u \in V \mid \rho(u) \geq \rho(v)\}]$. 
Chordal Graphs and PES

**Theorem**

A graph is chordal, iff it has a PES.

Show: $\leftarrow$.

Show: $\rightarrow$.

Let $C$ be a cycle in $G$. 
A graph is chordal, iff it has a PES.

Show: $\iff$.

- Let $C$ be a cycle in $G$.
- Let $u$ be the first node in $C$ under the ordering $\rho$. 

Show: $\implies$.
Chordal Graphs and PES

**Theorem**

*A graph is chordal, iff it has a PES.*

Show: $\Leftarrow$.

- Let $C$ be a cycle in $G$.
- Let $u$ be the first node in $C$ under the ordering $\rho$.
- Thus the neighbours of $u$ are connected.

Show: $\Rightarrow$.

Choose simplicial node $v$ and let $\rho(v) = 1$.
Compute recursively more nodes of $G - v$. 

\[ \Sigma = 0 \]
Chordal Graphs and PES

**Theorem**

A graph is chordal, iff it has a PES.

Show: $\Leftarrow$.

- Let $C$ be a cycle in $G$.
- Let $u$ be the first node in $C$ under the ordering $\rho$.
- Thus the neighbours of $u$ are connected.
- **Thus $G$ is chordal.**

Show: $\Rightarrow$.

Choose simplicial node $v$ and let $\rho(v) = 1$.

Compute recursively more nodes of $G - v$.
Chordal Graphs and PES

Theorem

A graph is chordal, iff it has a PES.

Show: \( \iff \).

- Let \( C \) be a cycle in \( G \).
- Let \( u \) be the first node in \( C \) under the ordering \( \rho \).
- Thus the neighbours of \( u \) are connected.
- Thus \( G \) is chordal.

Show: \( \implies \).

- Choose simplicial node \( v \) and let \( \rho(v) = 1 \).
Chordal Graphs and PES

Theorem

A graph is chordal, iff it has a PES.

Show: $\iff$.

- Let $C$ be a cycle in $G$.
- Let $u$ be the first node in $C$ under the ordering $\rho$.
- Thus the neighbours of $u$ are connected.
- Thus $G$ is chordal.

Show: $\implies$.

- Choose simplicial node $v$ und let $\rho(v) = 1$.
- Compute recursively more nodes of $G - v$. 
Theorem

Chordal graphs could be recognized in polynomial time.
Theorem

Chordal graphs could be recognized in polynomial time.

Proof: determine a PES (on the next slides).
Theorem

Chordal graphs could be recognized in polynomial time.

Proof: determine a PES (on the next slides).

Theorem

Chordal graphs could be recognized in time $O(n^2 \cdot m)$. 
Recognition

Theorem

Chordal graphs could be recognized in polynomial time.

Proof: determine a PES (on the next slides).

Theorem

Chordal graphs could be recognized in time \(O(n^2 \cdot m)\).

Theorem

Chordal graphs could be recognized in time \(O(n + m)\).
Overview and Simple Algorithm

- Compute an ordering for $G$. 
Overview and Simple Algorithm

- Compute an ordering for $G$.
- Compute this ordering simply by using the node degrees.
Overview and Simple Algorithm

- Compute an ordering for $G$.
- Compute this ordering simply by using the node degrees.
- Show that this ordering is always a PES, if $G$ is chordal.
Overview and Simple Algorithm

- Compute an ordering for $G$.
- Compute this ordering simply by using the node degrees.
- Show that this ordering is always a PES, if $G$ is chordal.
- We will get the following algorithm:
Overview and Simple Algorithm

- Compute an ordering for $G$.
- Compute this ordering simply by using the node degrees.
- Show that this ordering is always a PES, if $G$ is chordal.
- We will get the following algorithm:
  - Compute ordering using the node degrees.
Overview and Simple Algorithm

- Compute an ordering for $G$.
- Compute this ordering simply by using the node degrees.
- Show that this ordering is always a PES, if $G$ is chordal.
- We will get the following algorithm:
  - Compute ordering using the node degrees.
  - Test if this ordering is a PES.
Overview and Simple Algorithm

- Compute an ordering for $G$.
- Compute this ordering simply by using the node degrees.
- Show that this ordering is always a PES, if $G$ is chordal.
- We will get the following algorithm:
  - Compute ordering using the node degrees.
  - Test if this ordering is a PES.

Simple Algorithm:
Overview and Simple Algorithm

- Compute an ordering for $G$.
- Compute this ordering simply by using the node degrees.
- Show that this ordering is always a PES, if $G$ is chordal.
- We will get the following algorithm:
  - Compute ordering using the node degrees.
  - Test if this ordering is a PES.
- Simple Algorithm:
  - Compute the PES in a reverse fashion.
Overview and Simple Algorithm

- Compute an ordering for $G$.
- Compute this ordering simply by using the node degrees.
- Show that this ordering is always a PES, if $G$ is chordal.
- We will get the following algorithm:
  - Compute ordering using the node degrees.
  - Test if this ordering is a PES.

- Simple Algorithm:
  - Compute the PES in a reverse fashion.
  - Start with an arbitrary node $v_n$. 
Overview and Simple Algorithm

- Compute an ordering for $G$.
- Compute this ordering simply by using the node degrees.
- Show that this ordering is always a PES, if $G$ is chordal.

We will get the following algorithm:

- Compute ordering using the node degrees.
- Test if this ordering is a PES.

Simple Algorithm:

- Compute the PES in a reverse fashion.
- Start with an arbitrary node $v_n$.
- Choose $v_{i-1}$ such that $v_{i-1}$ is connected to as many as possible nodes from $v_i, v_{i+1}, \ldots, v_n$. 
Overview and Simple Algorithm

- Compute an ordering for $G$.
- Compute this ordering simply by using the node degrees.
- Show that this ordering is always a PES, if $G$ is chordal.
- We will get the following algorithm:
  - Compute ordering using the node degrees.
  - Test if this ordering is a PES.

- Simple Algorithm:
  - Compute the PES in a reverse fashion.
  - Start with an arbitrary node $v_n$.
  - Choose $v_{i-1}$ such that $v_{i-1}$ is connected to as many as possible nodes from $v_i, v_{i+1}, \ldots, v_n$.
  - Show $v_1, v_2, \ldots, v_n$ is a PES.
Helpfull Lemma

Lemma

A total ordering $\rho$ auf $V$ is a PES, iff for all pairs of nodes $v_i, v_j$, which are connected by a path, for which for all inner nodes $u$ $\rho(u) < \min(\rho(v_i), \rho(v_j))$ holds,
Helpful Lemma

Lemma

A total ordering \( \rho \) auf \( V \) is a PES, iff for all pairs of nodes \( v_i, v_j \), which are connected by a path, for which for all inner nodes \( u \) \( \rho(u) < \min(\rho(v_i), \rho(v_j)) \) holds, then follows that these nodes \( v_i, v_j \) are connected by an edge.
Helpfull Lemma

**Lemma**

A total ordering ρ auf V is a PES, iff for all pairs of nodes v_i, v_j, which are connected by a path, for which for all inner nodes u ρ(u) < min(ρ(v_i), ρ(v_j)) holds, then follows that these nodes v_i, v_j are connected by an edge.
Lemma

A total ordering $\rho$ auf $V$ is a PES, iff for all pairs of nodes $v_i, v_j$, which are connected by a path, for which for all inner nodes $u$ $\rho(u) < \min(\rho(v_i), \rho(v_j))$ holds, then follows that these nodes $v_i, v_j$ are connected by an edge.

Proof $\implies$ by contradiction.

Proof $\impliedby$ is simple.
Lemma

A total ordering $\rho$ auf $V$ is a PES, iff for all pairs of nodes $v_i, v_j$, which are connected by a path, for which for all inner nodes $u$ $\rho(u) < \min(\rho(v_i), \rho(v_j))$ holds, then follows that these nodes $v_i, v_j$ are connected by an edge.

- Proof $\implies$ by contradiction.
- Let $v_i, v_j$ be as above with $\{v_i, v_j\} \notin E$.
- Proof $\impliedby$ is simple.
**A total ordering** \( \rho \) **auf** \( V \) **is a PES**, **iff** for all pairs of nodes \( v_i, v_j \), which are connected by a path, for which for all inner nodes \( u \) \( \rho(u) < \min(\rho(v_i), \rho(v_j)) \) holds, then follows that these nodes \( v_i, v_j \) are connected by an edge.

- **Proof \( \Rightarrow \) by contradiction.**
- Let \( v_i, v_j \) be as above with \( \{v_i, v_j\} \notin E \).
- Let \( P \) the shortest path from \( v_i \) to \( v_j \) and let \( u \) be the leftmost node from \( P \) in \( \rho \).

- **Proof \( \Leftarrow \) is simple.**

![Diagram](image-url)
Helpful Lemma

**Lemma**

A total ordering $\rho$ auf $V$ is a PES, iff for all pairs of nodes $v_i, v_j$, which are connected by a path, for which for all inner nodes $u$ $\rho(u) < \min(\rho(v_i), \rho(v_j))$ holds, then follows that these nodes $v_i, v_j$ are connected by an edge.

- Proof $\implies$ by contradiction.
- Let $v_i, v_j$ be as above with $\{v_i, v_j\} \notin E$.
- Let $P$ the shortest path from $v_i$ to $v_j$ and let $u$ be the leftmost node from $P$ in $\rho$.
- The neighbours of $u$ on $P$ are connected by an edge.

- Proof $\iff$ is simple.
Helpfull Lemma

Lemma

A total ordering \( \rho \) auf \( V \) is a PES, iff for all pairs of nodes \( v_i, v_j \), which are connected by a path, for which for all inner nodes \( u \) \( \rho(u) < \min(\rho(v_i), \rho(v_j)) \) holds, then follows that these nodes \( v_i, v_j \) are connected by an edge.

- Proof \( \implies \) by contradiction.
- Let \( v_i, v_j \) be as above with \( \{v_i, v_j\} \notin E \).
- Let \( P \) the shortest path from \( v_i \) to \( v_j \) and let \( u \) be the leftmost node from \( P \) in \( \rho \).
- The neighbours of \( u \) on \( P \) are connected by an edge.

- Contradiction to the minimality of the path \( P \).

- Proof \( \impliedby \) is simple.

\[ \Sigma = 0 \]
Theorem

The simple algorithm computes for chordal graphs a PES.
Recognition

Theorem

*The simple algorithm computes for chordal graphs a PES.*

Claim

- Assume \( \rho(u) < \rho(v) < \rho(w) \) holds, with

![Diagram](image.png)
Theorem

The simple algorithm computes for chordal graphs a PES.

Claim

- Assume $\rho(u) < \rho(v) < \rho(w)$ holds, with
- $\{u, w\} \in E$ and $\{v, w\} \notin E$. 

$\Sigma = 0$
Recognition

**Theorem**

*The simple algorithm computes for chordal graphs a PES.*

**Claim**

- Assume $\rho(u) < \rho(v) < \rho(w)$ holds, with
- $\{u, w\} \in E$ and $\{v, w\} \notin E$.
- Then there is a node $z$ with:
**Theorem**

*The simple algorithm computes for chordal graphs a PES.*

**Claim**

- Assume $\rho(u) < \rho(v) < \rho(w)$ holds, with
- $\{u, w\} \in E$ and $\{v, w\} \notin E$.
- Then there is a node $z$ with:
- $\rho(v) < \rho(z)$, $\{u, z\} \notin E$ and $\{v, z\} \in E$. 

![Graph diagram](attachment://graph.png)
Theorem

The simple algorithm computes for chordal graphs a PES.

Claim

Assume $\rho(u) < \rho(v) < \rho(w)$ holds, with

- $\{u, w\} \in E$ and $\{v, w\} \notin E$.
- Then there is a node $z$ with:
  - $\rho(v) < \rho(z)$, $\{u, z\} \notin E$ and $\{v, z\} \in E$. 

Proof:
Holds due to the chosen ordering. $v$ has at least as many neighbours as $u$. 

Diagram:

- Nodes: $u, v, z, w, z'$
- Edges: $u-v, v-z, z-w, z'-w$
Recognition

**Theorem**

*The simple algorithm computes for chordal graphs a PES.*

**Claim**

- Assume $\rho(u) < \rho(v) < \rho(w)$ holds, with
- $\{u, w\} \in E$ and $\{v, w\} \not\in E$.
- Then there is a node $z$ with:
  - $\rho(v) < \rho(z)$, $\{u, z\} \not\in E$ and $\{v, z\} \in E$.

![Graph Diagram](image.png)

**Proof:**

- Holds due to the chosen ordering.
Recognition

**Theorem**

*The simple algorithm computes for chordal graphs a PES.*

**Claim**

- Assume $\rho(u) < \rho(v) < \rho(w)$ holds, with
- $\{u, w\} \in E$ and $\{v, w\} \notin E$.
- Then there is a node $z$ with:
- $\rho(v) < \rho(z)$, $\{u, z\} \notin E$ and $\{v, z\} \in E$.

**Proof:**

- Holds due to the chosen ordering.
- $v$ has at least as many neighbours as $u$. 
Recognition (Show, $\rho$ defines a PES)

Assume that this does not hold:

\begin{itemize}
\item Assume that this does not hold:
\end{itemize}
Recognition (Show, \( \rho \) defines a PES)

- Assume that this does not hold:
- There are \( v, w \) with \( \{v, w\} \not\in E \)
  and
Recognition (Show, $\rho$ defines a PES)

- Assume that this does not hold:
- There are $v, w$ with $\{v, w\} \not\in E$
  and
- for all inner nodes $u$ on the path $P$ of $v, w$ holds:
Recognition (Show, $\rho$ defines a PES)

- Assume that this does not hold:
- There are $v, w$ with $\{v, w\} \not\in E$ and
- for all inner nodes $u$ on the path $P$ of $v, w$ holds:
- $\rho(u) < \min(\rho(v), \rho(w))$. 
Recognition (Show, $\rho$ defines a PES)

- Assume that this does not hold:
- There are $v, w$ with $\{v, w\} \not\in E$ and
- for all inner nodes $u$ on the path $P$ of $v, w$ holds:
  - $\rho(u) < \min(\rho(v), \rho(w))$.
- Choose $\rho(w)$ maximal and after that $\rho(v)$ maximal.
Recognition (Show, $\rho$ defines a PES)

- Assume that this does not hold:
- There are $v, w$ with $\{v, w\} \notin E$ and
- for all inner nodes $u$ on the path $P$ of $v, w$ holds:
  - $\rho(u) < \min(\rho(v), \rho(w))$.
- Choose $\rho(w)$ maximal and after that $\rho(v)$ maximal.
- Choose shortest path $P$ from $w$ to $v$. 

\[ \Sigma = 0 \]
Recognition (Show, $\rho$ defines a PES)

- Assume that this does not hold:
- There are $v, w$ with $\{v, w\} \notin E$ and
- for all inner nodes $u$ on the path $P$ of $v, w$ holds:
  - $\rho(u) < \min(\rho(v), \rho(w))$.
- Choose $\rho(w)$ maximal and after that $\rho(v)$ maximal.
- Choose shortest path $P$ from $w$ to $v$.
- This path contains inner node $u$. 
Recognition (Show, $\rho$ defines a PES)

- Assume that this does not hold:
  - There are $v, w$ with $\{v, w\} \notin E$ and
  - for all inner nodes $u$ on the path $P$ of $v, w$ holds:
    - $\rho(u) < \min(\rho(v), \rho(w))$.
  - Choose $\rho(w)$ maximal and after that $\rho(v)$ maximal.
  - Choose shortest path $P$ from $w$ to $v$.
  - This path contains inner node $u$.

- There exists $z$ with: $\rho(v) < \rho(z)$, $\{u, z\} \notin E$ and $\{v, z\} \in E$. 
Recognition (Show, $\rho$ defines a PES)

- Assume that this does not hold:
- There are $v, w$ with $\{v, w\} \not\in E$ and
- for all inner nodes $u$ on the path $P$ of $v, w$ holds:
  $\rho(u) < \min(\rho(v), \rho(w))$.
- Choose $\rho(w)$ maximal and after that $\rho(v)$ maximal.
- Choose shortest path $P$ from $w$ to $v$.
- This path contains inner node $u$.

- There exists $z$ with: $\rho(v) < \rho(z)$, $\{u, z\} \not\in E$ and $\{v, z\} \in E$.
- Therefore is $w$ with $z$ connected by a path.
Recognition (Show, $\rho$ defines a PES)

- Assume that this does not hold:
- There are $v, w$ with $\{v, w\} \not\in E$ and
- for all inner nodes $u$ on the path $P$ of $v, w$ holds:
  - $\rho(u) < \min(\rho(v), \rho(w))$.
- Choose $\rho(w)$ maximal and after that $\rho(v)$ maximal.
- Choose shortest path $P$ from $w$ to $v$.
- This path contains inner node $u$.

- There exists $z$ with: $\rho(v) < \rho(z)$, $\{u, z\} \not\in E$ and $\{v, z\} \in E$.
- Therefore is $w$ with $z$ connected by a path.
- Because of the choosing of $v$ and $w$ holds $\{z, w\} \in E$. 

\[ \Sigma = 0 \]
Recognition (Show, $\rho$ defines a PES)

- Assume that this does not hold:
- There are $v, w$ with $\{v, w\} \not\in E$ and
- for all inner nodes $u$ on the path $P$ of $v, w$ holds:
- $\rho(u) < \min(\rho(v), \rho(w))$.
- Choose $\rho(w)$ maximal and after that $\rho(v)$ maximal.
- Choose shortest path $P$ from $w$ to $v$.
- This path contains inner node $u$.

- There exists $z$ with: $\rho(v) < \rho(z)$, $\{u, z\} \not\in E$ and $\{v, z\} \in E$.
- Therefore is $w$ with $z$ connected by a path.
- Because of the choosing of $v$ and $w$ holds $\{z, w\} \in E$.
- There is a cycle traversing $P$, $\{v, z\}$ and $\{z, w\}$. 

\[ \Sigma = 0 \]
Recognition (Show, $\rho$ defines a PES)

- Assume that this does not hold:
- There are $v, w$ with $\{v, w\} \notin E$ and
- for all inner nodes $u$ on the path $P$ of $v, w$ holds:
  - $\rho(u) < \min(\rho(v), \rho(w))$.
- Choose $\rho(w)$ maximal and after that $\rho(v)$ maximal.
- Choose shortest path $P$ from $w$ to $v$.
- This path contains inner node $u$.

- There exists $z$ with: $\rho(v) < \rho(z)$, $\{u, z\} \notin E$ and $\{v, z\} \in E$.
- Therefore is $w$ with $z$ connected by a path.
- Because of the choosing of $v$ and $w$ holds $\{z, w\} \in E$.
- There is a cycle traversing $P$, $\{v, z\}$ and $\{z, w\}$.
- Choose the shortest path between $u$ and $v$. 

![Diagram of a graph with nodes and edges showing the path and cycle traversed by $P$.]
Recognition (Show, $\rho$ defines a PES)

- Assume that this does not hold:
- There are $v, w$ with $\{v, w\} \notin E$ and
- for all inner nodes $u$ on the path $P$ of $v, w$ holds:
  - $\rho(u) < \min(\rho(v), \rho(w))$.
- Choose $\rho(w)$ maximal and after that $\rho(v)$ maximal.
- Choose shortest path $P$ from $w$ to $v$.
- This path contains inner node $u$.
- There exists $z$ with: $\rho(v) < \rho(z)$, $\{u, z\} \notin E$ and $\{v, z\} \in E$.
- Therefore is $w$ with $z$ connected by a path.
- Because of the choosing of $v$ and $w$ holds $\{z, w\} \in E$.
- There is a cycle traversing $P$, $\{v, z\}$ and $\{z, w\}$.
- Choose the shortest path between $u$ and $v$.
- Thus we have a non chordal cycle containing $\geq 4$ nodes.
Recognition (Running Time)

- The test of the clique property may be more consuming.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
The test of the clique property may be more consuming.

Test of the clique property may be done just by using data from the leftmost node of the clique.

Therefore the edges are considered only once.

Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
The test of the clique property may be more consuming.

Test of the clique property may be done just by using data from the leftmost node of the clique.

Therefore the edges are considered only once.

Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
The test of the clique property may be more consuming.

Test of the clique property may be done just by using data from the leftmost node of the clique.

Therefore the edges are considered only once.

Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Recognition (Running Time)

- The test of the clique property may be more consuming.
- Test of the clique property may be done just by using data from the leftmost node of the clique.
- Therefore the edges are considered only once.
- Thus the recognition could be done in linear time.
Test PES Property

- The algorithm:

  \[ N_i = \{ v_j \in \Gamma(v_i) \mid j > i \} \]
  \[ R_i = |\{ v_j \in \Gamma(v_i) \mid j > i \}| \]
Test PES Property

The algorithm:

- Start with an arbitrary node $v_n$. 

What is necessary to compute the ordering:

$N_i = \{ v_j \in \Gamma(v_i) \mid j > i \}$

$R_i = |\{ v_j \in \Gamma(v_i) \mid j > i \}|$
Test PES Property

- The algorithm:
  - Start with an arbitrary node $v_n$.
  - Choose $v_{i-1}$ such that it is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.
Test PES Property

- The algorithm:
  - Start with an arbitrary node $v_n$.
  - Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.
  - Show $v_1, v_2, \ldots, v_n$ is a PES.
Test PES Property

- The algorithm:
  - Start with an arbitrary node $v_n$.
  - Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.
  - Show $v_1, v_2, \ldots, v_n$ is a PES.

- What is necessary to compute the ordering:
Test PES Property

The algorithm:

- Start with an arbitrary node $v_n$.
- Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.
- Show $v_1, v_2, \ldots, v_n$ is a PES.

What is necessary to compute the ordering:

- $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$
Test PES Property

- The algorithm:
  - Start with an arbitrary node $v_n$.
  - Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.
  - Show $v_1, v_2, \ldots, v_n$ is a PES.

- What is necessary to compute the ordering:
  - $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$
  - $R_i = |\{v_j \in \Gamma(v_i) \mid j > i\}|$
Test PES Property

- The algorithm:
  - Start with an arbitrary node $v_n$.
  - Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.
  - Show $v_1, v_2, \ldots, v_n$ is a PES.

- What is necessary to compute the ordering:
  - $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$
  - $R_i = |\{v_j \in \Gamma(v_i) \mid j > i\}|$

- What is necessary to do the following test:
Test PES Property

- The algorithm:
  - Start with an arbitrary node $v_n$.
  - Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.
  - Show $v_1, v_2, \ldots, v_n$ is a PES.

- What is necessary to compute the ordering:
  - $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$
  - $R_i = |\{v_j \in \Gamma(v_i) \mid j > i\}|$

- What is necessary to do the following test:
  - Test $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$ induces a clique.
Compute $R_i$: Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.

- Let $B_0 = V$, $D = \emptyset$ and $l = n$. 

\[ R_i = |\{ v_j \in \Gamma(v_i) \mid j > i \}|. \]
Compute $R_i$

Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.

- Let $B_0 = V$, $D = \emptyset$ and $l = n$.
- Let for $1 \leq i \leq n - 1$ be: $B_i = \emptyset$. 

Let $R_i = |\{v_j \in \Gamma(v_i) | j > i\}|$. If a node $x = v_i$ as chosen, then $R_i(x)$ is not changed any more. Then:

$$R_i = R_i(x) = |\{v_j \in \Gamma(v_i) | j > i\}|.$$
Compute $R_i$

Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.

- Let $B_0 = V$, $D = \emptyset$ and $l = n$.
- Let for $1 \leq i \leq n - 1$ be: $B_i = \emptyset$.
- Let for all $v \in V$ be: $R(v) = 0$. 

If a node $x = v_i$ as chosen, then $R(x)$ is not changed any more.

Then: $R_i = R(x) = |\{v_j \in \Gamma(v_i) | j > i\}|$ holds.
Compute $R_i$:

Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.

- Let $B_0 = V$, $D = \emptyset$ and $l = n$.
- Let for $1 \leq i \leq n - 1$ be: $B_i = \emptyset$.
- Let for all $v \in V$ be: $R(v) = 0$.
- While $B_i \neq \emptyset$ for an $i$ do for the minimal $i$:
Compute $R_i$:

Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.

- Let $B_0 = V$, $D = \emptyset$ and $l = n$.
- Let for $1 \leq i \leq n-1$ be: $B_i = \emptyset$.
- Let for all $v \in V$ be: $R(v) = 0$.
- While $B_i \neq \emptyset$ for an $i$ do for the minimal $i$:
  1. Choose $x \in B_i$. 

- Let $R_{i-1} = |\{v_j \in \Gamma(v_i) | j > i\}|$. 
- If a node $x = v_i$ as chosen, then $R(x)$ is not changed anymore.
Compute $R_i$:

Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.

- Let $B_0 = V$, $D = \emptyset$ and $l = n$.
- Let for $1 \leq i \leq n - 1$ be: $B_i = \emptyset$.
- Let for all $v \in V$ be: $R(v) = 0$.
- While $B_i \neq \emptyset$ for an $i$ do for the minimal $i$:
  1. Choose $x \in B_i$.
  2. Let $v_l = x$ and $D = D \cup \{x\}$.
Compute $R_i$

Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.

- Let $B_0 = V$, $D = \emptyset$ and $l = n$.
- Let for $1 \leq i \leq n - 1$ be: $B_i = \emptyset$.
- Let for all $v \in V$ be: $R(v) = 0$.
- While $B_i \neq \emptyset$ for an $i$ do for the minimal $i$:
  1. Choose $x \in B_i$.
  2. Let $v_i = x$ and $D = D \cup \{x\}$.
  3. Let $\rho(x) = l$. 
Compute $R_i$:

Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.

- Let $B_0 = V$, $D = \emptyset$ and $l = n$.
- Let for $1 \leq i \leq n - 1$ be: $B_i = \emptyset$.
- Let for all $v \in V$ be: $R(v) = 0$.
- While $B_i \neq \emptyset$ for an $i$ do for the minimal $i$:
  1. Choose $x \in B_i$.
  2. Let $v_i = x$ and $D = D \cup \{x\}$.
  3. Let $\rho(x) = l$.
  4. Let $l = l - 1$. 
Compute $R_i$:

Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.

- Let $B_0 = V$, $D = \emptyset$ and $l = n$.
- Let for $1 \leq i \leq n - 1$ be: $B_i = \emptyset$.
- Let for all $v \in V$ be: $R(v) = 0$.
- While $B_i \neq \emptyset$ for an $i$ do for the minimal $i$:
  1. Choose $x \in B_i$.
  2. Let $v_i = x$ and $D = D \cup \{x\}$.
  3. Let $\rho(x) = l$.
  4. Let $l = l - 1$.
  5. Let $B_i = B_i \setminus \{x\}$.
Compute $R_i$

Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.

- Let $B_0 = V, D = \emptyset$ and $l = n$.
- Let for $1 \leq i \leq n - 1$ be: $B_i = \emptyset$.
- Let for all $v \in V$ be: $R(v) = 0$.
- While $B_i \neq \emptyset$ for an $i$ do for the minimal $i$:
  1. Choose $x \in B_i$.
  2. Let $v_i = x$ and $D = D \cup \{x\}$.
  3. Let $\rho(x) = l$.
  4. Let $l = l - 1$.
  5. Let $B_i = B_i \setminus \{x\}$.
  6. For all $v \in \Gamma(x) \setminus D$ do:
Compute $R_i$

Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.

- Let $B_0 = V$, $D = \emptyset$ and $l = n$.
- Let for $1 \leq i \leq n - 1$ be: $B_i = \emptyset$.
- Let for all $v \in V$ be: $R(v) = 0$.
- While $B_i \neq \emptyset$ for an $i$ do for the minimal $i$:
  1. Choose $x \in B_i$.
  2. Let $v_i = x$ and $D = D \cup \{x\}$.
  3. Let $\rho(x) = l$.
  4. Let $l = l - 1$.
  5. Let $B_i = B_i \setminus \{x\}$.
  6. For all $v \in \Gamma(x) \setminus D$ do:
     - Let $B_{R(v)} = B_{R(v)} \setminus \{v\}$.
Compute $R_i$

Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.

- Let $B_0 = V$, $D = \emptyset$ and $l = n$.
- Let for $1 \leq i \leq n - 1$ be: $B_i = \emptyset$.
- Let for all $v \in V$ be: $R(v) = 0$.
- While $B_i \neq \emptyset$ for an $i$ do for the minimal $i$:
  1. Choose $x \in B_i$.
  2. Let $v_i = x$ and $D = D \cup \{x\}$.
  3. Let $\rho(x) = l$.
  4. Let $l = l - 1$.
  5. Let $B_i = B_i \setminus \{x\}$.
  6. For all $v \in \Gamma(x) \setminus D$ do:
     - Let $B_{R(v)} = B_{R(v)} \setminus \{v\}$.
     - Let $R(v) = R(v) + 1$. 
Compute $R_i$

Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.

- Let $B_0 = V$, $D = \emptyset$ and $l = n$.
- Let for $1 \leq i \leq n - 1$ be: $B_i = \emptyset$.
- Let for all $v \in V$ be: $R(v) = 0$.
- While $B_i \neq \emptyset$ for an $i$ do for the minimal $i$:
  1. Choose $x \in B_i$.
  2. Let $v_i = x$ and $D = D \cup \{x\}$.
  3. Let $\rho(x) = l$.
  4. Let $l = l - 1$.
  5. Let $B_i = B_i \setminus \{x\}$.
  6. For all $v \in \Gamma(x) \setminus D$ do:
     - Let $B_{R(v)} = B_{R(v)} \setminus \{v\}$.
     - Let $R(v) = R(v) + 1$.
     - Let $B_{R(v)} = B_{R(v)} \cup \{v\}$.
Compute $R_i$

Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.

- Let $B_0 = V$, $D = \emptyset$ and $l = n$.
- Let for $1 \leq i \leq n - 1$ be: $B_i = \emptyset$.
- Let for all $v \in V$ be: $R(v) = 0$.
- While $B_i \neq \emptyset$ for an $i$ do for the minimal $i$:
  1. Choose $x \in B_i$.
  2. Let $v_i = x$ and $D = D \cup \{x\}$.
  3. Let $\rho(x) = l$.
  4. Let $l = l - 1$.
  5. Let $B_i = B_i \setminus \{x\}$.
  6. For all $v \in \Gamma(x) \setminus D$ do:
     - Let $B_{R(v)} = B_{R(v)} \setminus \{v\}$.
     - Let $R(v) = R(v) + 1$.
     - Let $B_{R(v)} = B_{R(v)} \cup \{v\}$.

- Task was to compute: $R_i = |\{v_j \in \Gamma(v_i) \mid j > i\}|$. 
Compute $R_i$

Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.

- Let $B_0 = V$, $D = \emptyset$ and $l = n$.
- Let for $1 \leq i \leq n - 1$ be: $B_i = \emptyset$.
- Let for all $v \in V$ be: $R(v) = 0$.
- While $B_i \neq \emptyset$ for an $i$ do for the minimal $i$:
  1. Choose $x \in B_i$.
  2. Let $v_i = x$ and $D = D \cup \{x\}$.
  3. Let $\rho(x) = l$.
  4. Let $l = l - 1$.
  5. Let $B_i = B_i \setminus \{x\}$.
  6. For all $v \in \Gamma(x) \setminus D$ do:
     - Let $B_{R(v)} = B_{R(v)} \setminus \{v\}$.
     - Let $R(v) = R(v) + 1$.
     - Let $B_{R(v)} = B_{R(v)} \cup \{v\}$.

- Task was to compute: $R_i = |\{v_j \in \Gamma(v_i) \mid j > i\}|$.
- If a node $x = v_i$ as chosen, then $R(x)$ is not changed any more.
Compute $R_i$:

Choose $v_{i-1}$ such that is connected with as many as possible nodes $v_i, v_{i+1}, \ldots, v_n$.

- Let $B_0 = V$, $D = \emptyset$ and $l = n$.
- Let for $1 \leq i \leq n-1$ be: $B_i = \emptyset$.
- Let for all $v \in V$ be: $R(v) = 0$.
- While $B_i \neq \emptyset$ for an $i$ do for the minimal $i$:
  1. Choose $x \in B_i$.
  2. Let $v_i = x$ and $D = D \cup \{x\}$.
  3. Let $\rho(x) = l$.
  4. Let $l = l - 1$.
  5. Let $B_i = B_i \setminus \{x\}$.
  6. For all $v \in \Gamma(x) \setminus D$ do:
     - Let $B_{R(v)} = B_{R(v)} \setminus \{v\}$.
     - Let $R(v) = R(v) + 1$.
     - Let $B_{R(v)} = B_{R(v)} \cup \{v\}$.

Task was to compute: $R_i = |\{v_j \in \Gamma(v_i) \mid j > i\}|$.

- If a node $x = v_i$ as chosen, then $R(x)$ is not changed any more.
- Then: $R_i = R(x) = |\{v_j \in \Gamma(v_i) \mid j > i\}|$ holds.
Test $N_i$

- Getting the idea:

\[ N_i = \{ v_j \in \Gamma(v_i) \mid j > i \} \text{ induces a clique.} \]
Test $N_i$

- Getting the idea:
- Check the nodes from left to right.

Test $N_i = \{ v_j \in \Gamma(v_i) \mid j > i \}$ induces a clique.
Test $N_i$

- Getting the idea:
- Check the nodes from left to right.
- For some node $v_i$ do not at once the test of $N_i$ to be a clique.

Test $N_i = \{ v_j \in \Gamma(v_i) \mid j > i \}$ induces a clique.
Test $N_i$

- Getting the idea:
- Check the nodes from left to right.
- For some node $v_i$ do not at once the test of $N_i$ to be a clique.
- Instead delay the test on for each neighbour $v_j$ of $v_i$. 

Test $N_i = \{ v_j \in \Gamma(v_i) \mid j > i \}$ induces a clique.
Test $N_i$

- Getting the idea:
- Check the nodes from left to right.
- For some node $v_i$ do not at once the test of $N_i$ to be a clique.
- Instead delay the test on for each neighbour $v_j$ of $v_i$.
- But prepare, the set of neighbours which $v_j$ should have.

Test $N_i = \{ v_j \in \Gamma(v_i) \mid j > i \}$ induces a clique.
Test $N_i$

- Getting the idea:
- Check the nodes from left to right.
- For some node $v_i$ do not at once the test of $N_i$ to be a clique.
- Instead delay the test on for each neighbour $v_j$ of $v_i$.
- But prepare, the set of neighbours which $v_j$ should have.
- Store this in tables $T[v_j]$.

Test $N_i = \{ v_j \in \Gamma(v_i) \mid j > i \}$ induces a clique.
Test \( N_i \)

- For all \( v_j \in V \) do \( T[v_j] = \emptyset \).

\( N_i = \{ v_j \in \Gamma(v_i) \mid j > i \} \) induces a clique.
Test $N_i$

- For all $v_j \in V$ do $T[v_j] = \emptyset$.
- For all $i$ from 1 to $n$ do:

Test $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$ induces a clique.
Test $N_i$

- For all $v_j \in V$ do $T[v_j] = \emptyset$.
- For all $i$ from 1 to $n$ do:
  1. Consider the node $v_i$.

Test $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$ induces a clique.
Test $N_i$

- For all $v_j \in V$ do $T[v_j] = \emptyset$.
- For all $i$ from 1 to $n$ do:
  1. Consider the node $v_i$.
  2. Let $N = \{v_j \in \Gamma(v_i) \mid j > i\}$.

Test $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$ induces a clique.
Test $N_i$

- For all $v_j \in V$ do $T[v_j] = \emptyset$.
- For all $i$ from 1 to $n$ do:
  1. Consider the node $v_i$.
  2. Let $N = \{ v_j \in \Gamma(v_i) \mid j > i \}$.
  3. If $T[v_i] \not\subset N$ holds, the stop with message “no PES”.

Test $N_i = \{ v_j \in \Gamma(v_i) \mid j > i \}$ induces a clique.
Test $N_i$

- For all $v_j \in V$ do $T[v_j] = \emptyset$.

- For all $i$ from 1 to $n$ do:
  1. Consider the node $v_i$.
  2. Let $N = \{v_j \in \Gamma(v_i) \mid j > i\}$.
  3. If $T[v_i] \not\subseteq N$ holds, the stop with message “no PES”.
  4. If $N \neq \emptyset$ then:
     - Let $v_l$ be the first (left) node of $N$.
     - Let $T[v_l] = T[v_l] \cup (N \setminus \{v_l\})$.
     - Output: the ordering is a PES.

Test $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$ induces a clique.
Test $N_i$

- For all $v_j \in V$ do $T[v_j] = \emptyset$.
- For all $i$ from 1 to $n$ do:
  1. Consider the node $v_i$.
  2. Let $N = \{v_j \in \Gamma(v_i) \mid j > i\}$.
  3. If $T[v_i] \not\subset N$ holds, the stop with message “no PES”.
  4. If $N \neq \emptyset$ the do:
     - Let $v_l$ be the first (left) node of $N$. 

Test $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$ induces a clique.
Test $N_i$

- For all $v_j \in V$ do $T[v_j] = \emptyset$.
- For all $i$ from 1 to $n$ do:
  1. Consider the node $v_i$.
  2. Let $N = \{v_j \in \Gamma(v_i) \mid j > i\}$.
  3. If $T[v_i] \not\subset N$ holds, the stop with message “no PES”.
  4. If $N \neq \emptyset$ the do:
     - Let $v_l$ be the first (left) node of $N$.
     - Let $T[v_l] = T[v_l] \cup (N \setminus \{v_l\})$. 

Test $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$ induces a clique.
Test $N_i$

- For all $v_j \in V$ do $T[v_j] = \emptyset$.

- For all $i$ from 1 to $n$ do:
  1. Consider the node $v_i$.
  2. Let $N = \{v_j \in \Gamma(v_i) \mid j > i\}$.
  3. If $T[v_i] \not\subset N$ holds, the stop with message "no PES".
  4. If $N \neq \emptyset$ the do:
     - Let $v_l$ be the first (left) node of $N$.
     - Let $T[v_l] = T[v_l] \cup (N \setminus \{v_l\})$.

- Output: the ordering is a PES.
Test $N_i$.

For all $v_j \in V$ do $T[v_j] = \emptyset$.

Test $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$ induces a clique.

For all $v_j \in V$ do $S[v_j] = 0$.

For all $i$ from 1 to $n$ do:

1. Consider the node $v_i$.
2. Let $N = \{v_j \in \Gamma(v_i) \mid j > i\}$.
3. For all $v \in N$ do $S[v] = 1$.
4. For all $u \in T[v_i]$ do
   - If $S[u] = 0$ holds, then stop with message "No PES".
5. For all $v_j \in V$ do $S[v_j] = 0$.
6. If $N \neq \emptyset$ then
   - Let $v_l$ be the first (left) node of $N$.
   - Let $T[v_l] = T[v_l] \cup (N \setminus \{v_l\})$.

Output: the ordering is a PES.
Teste $N_i$:

- For all $v_j \in V$ do $T[v_j] = \emptyset$.
- For all $v_j \in V$ do $S[v_j] = 0$.
Theorem: For all $v_j \in V$ do $T[v_j] = \emptyset$.

- For all $v_j \in V$ do $S[v_j] = 0$.
- For all $i$ from 1 to $n$ do:

\[ N_i = \{ v_j \in \Gamma(v_i) \mid j > i \} \] induces a clique.
Teste $N_i$

- For all $v_j \in V$ do $T[v_j] = \emptyset$.
- For all $v_j \in V$ do $S[v_j] = 0$.
- For all $i$ from 1 to $n$ do:
  1. Consider the node $v_i$.

Test $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$ induces a clique.
Teste $N_i$

- For all $v_j \in V$ do $T[v_j] = \emptyset$.
- For all $v_j \in V$ do $S[v_j] = 0$.
- For all $i$ from 1 to $n$ do:
  1. Consider the node $v_i$.
  2. Let $N = \{v_j \in \Gamma(v_i) \mid j > i\}$.

Test $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$ induces a clique.
Teste $N_i$

- For all $v_j \in V$ do $T[v_j] = \emptyset$.
- For all $v_j \in V$ do $S[v_j] = 0$.
- For all $i$ from 1 to $n$ do:
  1. Consider the node $v_i$.
  2. Let $N = \{v_j \in \Gamma(v_i) \mid j > i\}$.
  3. For all $v \in N$ do $S[v] = 1$.

Test $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$ induces a clique.
Teste $N_i$

- For all $v_j \in V$ do $T[v_j] = \emptyset$.
- For all $v_j \in V$ do $S[v_j] = 0$.
- For all $i$ from 1 to $n$ do:
  1. Consider the node $v_i$.
  2. Let $N = \{v_j \in \Gamma(v_i) \mid j > i\}$.
  3. For all $v \in N$ do $S[v] = 1$.
  4. For all $u \in T[v_i]$ do

$$\text{Test } N_i = \{v_j \in \Gamma(v_i) \mid j > i\} \text{ induces a clique.}$$
Teste $N_i$

- For all $v_j \in V$ do $T[v_j] = \emptyset$.
- For all $v_j \in V$ do $S[v_j] = 0$.
- For all $i$ from 1 to $n$ do:
  1. Consider the node $v_i$.
  2. Let $N = \{v_j \in \Gamma(v_i) \mid j > i\}$.
  3. For all $v \in N$ do $S[v] = 1$.
  4. For all $u \in T[v_i]$ do
     - If $S[u] = 0$ holds, then stop with message “No PES”.

Test $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$ induces a clique.
Teste $N_i$

- For all $v_j \in V$ do $T[v_j] = \emptyset$.
- For all $v_j \in V$ do $S[v_j] = 0$.
- For all $i$ from 1 to $n$ do:
  1. Consider the node $v_i$.
  2. Let $N = \{v_j \in \Gamma(v_i) \mid j > i\}$.
  3. For all $v \in N$ do $S[v] = 1$.
  4. For all $u \in T[v_i]$ do
     - If $S[u] = 0$ holds, then stop with message “No PES”.
  5. For all $v_j \in V$ do $S[v_j] = 0$.

Test $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$ induces a clique.
For all \( v_j \in V \) do \( T[v_j] = \emptyset \).

For all \( v_j \in V \) do \( S[v_j] = 0 \).

For all \( i \) from 1 to \( n \) do:

1. Consider the node \( v_i \).
2. Let \( N = \{v_j \in \Gamma(v_i) \mid j > i\} \).
3. For all \( v \in N \) do \( S[v] = 1 \).
4. For all \( u \in T[v_i] \) do
   - If \( S[u] = 0 \) holds, then stop with message “No PES”.
5. For all \( v_j \in V \) do \( S[v_j] = 0 \).
6. If \( N \neq \emptyset \) the do:

Test \( N_i = \{v_j \in \Gamma(v_i) \mid j > i\} \) induces a clique.
Consider the node $v_i$.

Let $N = \{v_j \in \Gamma(v_i) \mid j > i\}$.

For all $v \in N$ do $S[v] = 1$.

For all $u \in T[v_i]$ do

- If $S[u] = 0$ holds, then stop with message “No PES”.

For all $v_j \in V$ do $S[v_j] = 0$.

If $N \neq \emptyset$ the do:

- Let $v_l$ be the first (left) node of $N$. 

Test $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$ induces a clique.
Teste $N_i$

- For all $v_j \in V$ do $T[v_j] = \emptyset$.
- For all $v_j \in V$ do $S[v_j] = 0$.
- For all $i$ from 1 to $n$ do:
  1. Consider the node $v_i$.
  2. Let $N = \{v_j \in \Gamma(v_i) \mid j > i\}$.
  3. For all $v \in N$ do $S[v] = 1$.
  4. For all $u \in T[v_i]$ do
     - If $S[u] = 0$ holds, then stop with message “No PES”.
  5. For all $v_j \in V$ do $S[v_j] = 0$.
  6. If $N \neq \emptyset$ the do:
     - Let $v_l$ be the first (left) node of $N$.
     - Let $T[v_l] = T[v_l] \cup (N \setminus \{v_l\})$.

Test $N_i = \{v_j \in \Gamma(v_i) \mid j > i\}$ induces a clique.
Teste $N_i$

- For all $v_j \in V$ do $T[v_j] = \emptyset$.
- For all $v_j \in V$ do $S[v_j] = 0$.
- For all $i$ from 1 to $n$ do:
  - Consider the node $v_i$.
  - Let $N = \{ v_j \in \Gamma(v_i) \mid j > i \}$.
  - For all $v \in N$ do $S[v] = 1$.
  - For all $u \in T[v_i]$ do
    - If $S[u] = 0$ holds, then stop with message “No PES”.
  - For all $v_j \in V$ do $S[v_j] = 0$.
  - If $N \neq \emptyset$ the do:
    - Let $v_l$ be the first (left) node of $N$.
    - Let $T[v_l] = T[v_l] \cup (N \setminus \{v_l\})$.

- Output: the ordering is a PES.
The standard graph problems could be solved in polynomial time.
Algorithms for Graph Problems

- The standard graph problems could be solved in polynomial time.
- Idea: Greedy algorithm using the PES ordering.
Algorithms for Graph Problems

- The standard graph problems could be solved in polynomial time.
- Idea: Greedy algorithm using the PES ordering.
- Note: Chordal Graphs have at most $|V|$ maximum cliques.
Algorithms for Graph Problems

- The standard graph problems could be solved in polynomial time.
- Idea: Greedy algorithm using the PES ordering.
- Note: Chordal Graphs have at most $|V|$ maximum cliques.
- Thus only the simplicial nodes have to be considered for the clique problem.
The standard graph problems could be solved in polynomial time.

Idea: Greedy algorithm using the PES ordering.

Note: Chordal Graphs have at most $|V|$ maximum cliques.

Thus only the simplicial nodes have to be considered for the clique problem.

For the colouring problem use greedy on the revers PES ordering.
The standard graph problems could be solved in polynomial time.

Idea: Greedy algorithm using the PES ordering.

Note: Chordal Graphs have at most $|V|$ maximum cliques.

Thus only the simplicial nodes have to be considered for the clique problem.

For the colouring problem use greedy on the revers PES ordering.

Similar ideas work for the other problems.
Lemma

Let $\mathcal{T} = \{T_i \mid 1 \leq i \leq n\}$ be a family of subtrees of some base tree and each pair of trees from $\mathcal{T}$ intersect each other.

- Then they have a common node.
Lemma

Let $\mathcal{T} = \{ T_i | 1 \leq i \leq n \}$ be a family of subtrees of some base tree and each pair of trees from $\mathcal{T}$ intersect each other.

- Then they have a common node.
- I.e. $\bigcap_{1 \leq i \leq n} T_i \neq \emptyset$
Lemma

Let $\mathcal{T} = \{T_i \mid 1 \leq i \leq n\}$ be a family of subtrees of some base tree and each pair of trees from $\mathcal{T}$ intersect each other.

- Then they have a common node.
- i.e. $\bigcap_{1 \leq i \leq n} T_i \neq \emptyset$
Lemma

Let $\mathcal{T} = \{T_i \mid 1 \leq i \leq n\}$ be a family of subtrees of some base tree and each pair of trees from $\mathcal{T}$ intersect each other.

- Then they have a common node.
- I.e. $\bigcap_{1 \leq i \leq n} T_i \neq \emptyset$

The union of all subtrees $T_i$ induces a subtree $T'$.
Lemma

Let $\mathcal{T} = \{T_i \mid 1 \leq i \leq n\}$ be a family of subtrees of some base tree and each pair of trees from $\mathcal{T}$ intersect each other.

- Then they have a common node.
- I.e. $\bigcap_{1 \leq i \leq n} T_i \neq \emptyset$

- The union of all subtrees $T_i$ induces a subtree $T'$.
- A leave of $T'$ which is not in all $T_i$ could be deleted without changing the intersections of the $T_i$. 
Lemma

Let $\mathcal{T} = \{T_i \mid 1 \leq i \leq n\}$ be a family of subtrees of some base tree and each pair of trees from $\mathcal{T}$ intersect each other.

- Then they have a common node.
- I.e. $\bigcap_{1 \leq i \leq n} T_i \neq \emptyset$

- The union of all subtrees $T_i$ induces a subtree $T'$.
- A leave of $T'$ which is not in all $T_i$ could be deleted without changing the intersections of the $T_i$.
- By repeating we find a node which is common to all $T_i$. 
Lemma

Let $\mathcal{T} = \{T_i \mid 1 \leq i \leq n\}$ be a family of subtrees of some base tree and each pair of trees from $\mathcal{T}$ intersect each other.

- Then they have a common node.
- I.e. $\bigcap_{1 \leq i \leq n} T_i \neq \emptyset$

The union of all subtrees $T_i$ induces a subtree $T'$.

A leave of $T'$ which is not in all $T_i$ could be deleted without changing the intersections of the $T_i$.

By repeating we find a node which is common to all $T_i$. 
Lemma

Let $\mathcal{T} = \{T_i \mid 1 \leq i \leq n\}$ be a family of subtrees of some base tree and each pair of trees from $\mathcal{T}$ intersect each other.

- Then they have a common node.
- I.e. $\bigcap_{1 \leq i \leq n} T_i \neq \emptyset$

- The union of all subtrees $T_i$ induces a subtree $T'$.
- A leave of $T'$ which is not in all $T_i$ could be deleted without changing the intersections of the $T_i$.
- By repeating we find a node which is common to all $T_i$. 

\[ \Sigma = 0 \]
Statements

Theorem

Let $G = (\{v_1, v_2, \ldots, v_n\}, E)$ be a Graph. The following statements are equivalent:

1. $G$ is chordal.
Statements

Theorem

Let $G = (\{v_1, v_2, \ldots, v_n\}, E)$ be a Graph. The following statements are equivalent:

1. $G$ is chordal.
2. $G$ is the intersection graph of a family of subtrees.
Theorem

Let \( G = (\{v_1, v_2, \ldots, v_n\}, E) \) be a Graph. The following statements are equivalent:

1. \( G \) is chordal.
2. \( G \) is the intersection graph of a family of subtrees.
3. There is a tree \( B \) on the set of maximal cliques of \( G \) such that for a pair of cliques \( C', C'' \) holds:
Statements

Theorem

Let $G = (\{v_1, v_2, \ldots, v_n\}, E)$ be a Graph. The following statements are equivalent:

1. $G$ is chordal.
2. $G$ is the intersection graph of a family of subtrees.
3. There is a tree $B$ on the set of maximal cliques of $G$ such that for a pair of cliques $C', C''$ holds:
   - The clique $C' \cap C''$ is part of each maximal clique, which
Statements

Theorem

Let $G = (\{v_1, v_2, \ldots, v_n\}, E)$ be a Graph. The following statements are equivalent:

1. $G$ is chordal.
2. $G$ is the intersection graph of a family of subtrees.
3. There is a tree $B$ on the set of maximal cliques of $G$ such that for a pair of cliques $C', C''$ holds:
   - The clique $C' \cap C''$ is part of each maximal clique, which
   - is on the path from $C'$ to $C''$ in $B$. 

Proof I

Show: $G$ is chordal $\implies G$ is intersection graph of a family of subtrees.

- Proof by Induction.
Proof I

Show: $G$ is chordal $\implies G$ is intersection graph of a family of subtrees.

- Proof by Induction.
- $n = 1$ clear.
Proof I

Show: $G$ is chordal $\iff G$ is intersection graph of a family of subtrees.

- **Proof by Induction.**
- $n = 1$ clear.
- **Induction step:** $n - 1 \rightarrow n$
Proof I

Show: $G$ is chordal $\iff G$ is intersection graph of a family of subtrees.

- Proof by Induction.
- $n = 1$ clear.
- Induction step: $n - 1 \rightarrow n$
  - Nodes $v_1, v_2, \ldots, v_n$ and $s = v_n$ a simplicial node.
Proof I

Show: \( G \) is chordal \( \implies \) \( G \) is intersection graph of a family of subtrees.

- Proof by Induction.
- \( n = 1 \) clear.
- Induction step: \( n - 1 \to n \)
  - Nodes \( v_1, v_2, \ldots, v_n \) and \( s = v_n \) a simplicial node.
  - Let \( (B_{n-1}, \{T_1, T_2, \ldots, T_{n-1}\}) \) intersection graph representation for \( v_1, v_2, \ldots, v_{n-1} \)
Proof I

Show: $G$ is chordal $\Rightarrow$ $G$ is intersection graph of a family of subtrees.

- Proof by Induction.
- $n = 1$ clear.

- Induction step: $n - 1 \rightarrow n$
  - Nodes $v_1, v_2, \ldots, v_n$ and $s = v_n$ a simplicial node.
  - Let $(B_{n-1}, \{T_1, T_2, \ldots, T_{n-1}\})$ intersection graph representation for $v_1, v_2, \ldots, v_{n-1}$
  - $\Gamma(s) \setminus \{s\}$ is a clique.
Proof I

Show: \( G \) is chordal \( \iff \) \( G \) is intersection graph of a family of subtrees.

- Proof by Induction.
- \( n = 1 \) clear.
- Induction step: \( n - 1 \to n \)
  - Nodes \( v_1, v_2, \ldots, v_n \) and \( s = v_n \) a simplicial node.
  - Let \((B_{n-1}, \{T_1, T_2, \ldots, T_{n-1}\})\) intersection graph representation for \( v_1, v_2, \ldots, v_{n-1} \)
  - \( \Gamma(s) \setminus \{s\} \) is a clique.
  - There is a common node \( a \) in \( \bigcap_{v \in \Gamma(s)} V(T_v) \).
Proof I

Show: $G$ is chordal $\implies G$ is intersection graph of a family of subtrees.

- Proof by Induction.
- $n = 1$ clear.
- Induction step: $n - 1 \rightarrow n$
  - Nodes $v_1, v_2, \ldots, v_n$ and $s = v_n$ a simplicial node.
  - Let $(B_{n-1}, \{T_1, T_2, \ldots, T_{n-1}\})$ intersection graph representation for $v_1, v_2, \ldots, v_{n-1}$
  - $\Gamma(s) \setminus \{s\}$ is a clique.
  - There is a common node $a$ in $\cap_{v \in \Gamma(s)} V(T_v)$.
  - Add to $B_{n-1}$ a new leave $b$ for $a$. 

![Diagram of tree structures](image-url)
Proof I

Show: $G$ is chordal $\iff G$ is intersection graph of a family of subtrees.

- Proof by Induction.
- $n = 1$ clear.
- Induction step: $n - 1 \to n$
  - Nodes $v_1, v_2, \ldots, v_n$ and $s = v_n$ a simplicial node.
  - Let $(B_{n-1}, \{T_1, T_2, \ldots, T_{n-1}\})$ intersection graph representation for $v_1, v_2, \ldots, v_{n-1}$
  - $\Gamma(s) \setminus \{s\}$ is a clique.
  - There is a common node $a$ in $\cap_{v \in \Gamma(s)} V(T_v)$.
  - Add to $B_{n-1}$ a new leave $b$ for $a$.
  - And generate a new subtree, which consists of $b$. 

\[ \Sigma = \varnothing \]
Proof 1

Show: $G$ is chordal $\implies G$ is intersection graph of a family of subtrees.

- Proof by Induction.
- $n = 1$ clear.
- Induction step: $n - 1 \rightarrow n$
  - Nodes $v_1, v_2, \ldots, v_n$ and $s = v_n$ a simplicial node.
  - Let $(B_{n-1}, \{T_1, T_2, \ldots, T_{n-1}\})$ intersection graph representation for $v_1, v_2, \ldots, v_{n-1}$
  - $\Gamma(s) \setminus \{s\}$ is a clique.
  - There is a common node $a$ in $\bigcap_{v \in \Gamma(s)} V(T_v)$.
  - Add to $B_{n-1}$ a new leave $b$ for $a$.
  - And generate a new subtree, which consists of $b$.
  - And enlarge each subtree from $\Gamma(s)$ with $b$.

\begin{center}
\begin{tikzpicture}
  \node (T1) at (0,0) {$T_1$};
  \node (T2) at (1,0) {$T_2$};
  \node (T3) at (2,0) {$T_3$};
  \node (T4) at (3,0) {$T_4$};
  \node (T5) at (4,0) {$T_5$};
  \node (T6) at (0,-1) {$T_2 T_3 T_4 T_6$};
  \node (T7) at (1,-1) {$T_1 T_2 T_3 T_4 T_5$};
  \draw (T1) -- (T2) -- (T3) -- (T4) -- (T5);
  \draw (T2) -- (T3) -- (T4) -- (T5);
  \draw (T3) -- (T2) -- (T1);
\end{tikzpicture}
\end{center}
Proof II

Show: $G$ is intersection graph of a family of subtrees $\implies G$ is chordal.

- Let $C = (v_0, v_1, \ldots, v_{k-1})$ cycle of length $k \geq 4$. 
Proof II

Show: $G$ is intersection graph of a family of subtrees $\implies G$ is chordal.

- Let $C = (v_0, v_1, \ldots, v_{k-1})$ cycle of length $k \geq 4$.
- Let $T_0, T_1, \ldots, T_{k-1}$ be the corresponding trees.
Proof II

Show: $G$ is intersection graph of a family of subtrees $\implies G$ is chordal.

- Let $C = (v_0, v_1, \ldots, v_{k-1})$ cycle of length $k \geq 4$.
- Let $T_0, T_1, \ldots, T_{k-1}$ be the corresponding trees.
- These subtrees will form a cycle in the base tree.
Proof II

Show: $G$ is intersection graph of a family of subtrees $\iff G$ is chordal.

- Let $C = (v_0, v_1, \ldots, v_{k-1})$ cycle of length $k \geq 4$.
- Let $T_0, T_1, \ldots, T_{k-1}$ be the corresponding trees.
- These subtrees will form a cycle in the base tree.
Proof II

Show: $G$ is intersection graph of a family of subtrees $\implies G$ is chordal.

- Let $C = (v_0, v_1, \ldots, v_{k-1})$ cycle of length $k \geq 4$.
- Let $T_0, T_1, \ldots, T_{k-1}$ be the corresponding trees.
- These subtrees will form a cycle in the base tree.

The other part of the proof follows in a similar way.
Simple Statements

**Lemma**

Let $G$ be a chordal graph. A node $v$ of $G$ is simplicial, iff it is contained in only one maximal clique.
Simple Statements

Lemma

Let $G$ be a chordal graph. A node $v$ of $G$ is simplicial, iff it is contain in only one maximal clique.

Lemma

Let $G$ be a chordal graph and $C$ a clique in $G$. Then exitst a PES, which enumerates the nodes from $C$ last.
Theorem

Any chordal graph with \( n \) nodes has a \((\omega(G), 1/2)\)-separator, which is a clique.
**Theorem**

*Any chordal graph with n nodes has a \((\omega(G), 1/2)\)-separator, which is a clique.*

- **Note:** A separator of size \(\omega(G)\) must not be a Clique.
Result

Theorem

Any chordal graph with $n$ nodes has a $(\omega(G), 1/2)$-separator, which is a clique.

- Note: A separator of size $\omega(G)$ must not be a Clique.
- Note: A clique-separator must not be minimal separating.
Proof

- Algorithm to compute a chordal separator:
Proof

- Algorithm to compute a chordal separator:
  - $C := \emptyset$

Note: At the start $a$ is freely chosen. $C$ is always minimal separating for $A$ and $V \setminus (C \cup A)$. All nodes from $C$ have neighbours in $A$. There is at most one component $A$ with $|A| > n/2$. At each round, one node will be removed from that component. There are at most $\lceil n/2 \rceil$ iterations. Show $\exists a: C \subset \Gamma(a)$.
Proof

- Algorithm to compute a chordal separator:
  - $C := \emptyset$
  - As long a component $A$ in $G[V \setminus C]$ exists with $|A| > n/2$ do:
Proof

- Algorithm to compute a chordal separator:
  - $C := \emptyset$
  - As long a component $A$ in $G[V \setminus C]$ exists with $|A| > n/2$ do:
    - $C := \{c \in C \mid \Gamma(c) \cap A \neq \emptyset\}$
Proof

- Algorithm to compute a chordal separator:
  - \( C := \emptyset \)
  - As long a component \( A \) in \( G[V \setminus C] \) exists with \( |A| > n/2 \) do:
    - \( C := \{ c \in C \mid \Gamma(c) \cap A \neq \emptyset \} \)
    - Choose \( a \in A \) with: \( C \subset \Gamma(a) \)
Proof

- Algorithm to compute a chordal separator:
  
  - \( C := \emptyset \)
  
  As long a component \( A \) in \( G[V \setminus C] \) exists with \( |A| > n/2 \) do:
    
    - \( C := \{ c \in C \mid \Gamma(c) \cap A \neq \emptyset \} \)
    
    Choose \( a \in A \) with: \( C \subseteq \Gamma(a) \)
    
    - \( C := C \cup \{ a \} \)
Proof

- Algorithm to compute a chordal separator:
  - $C := \emptyset$
  - As long a component $A$ in $G[V \setminus C]$ exists with $|A| > n/2$ do:
    - $C := \{c \in C \mid \Gamma(c) \cap A \neq \emptyset\}$
    - Choose $a \in A$ with: $C \subset \Gamma(a)$
    - $C := C \cup \{a\}$
  - There is at most one component $A$ with: $|A| > n/2$. 

Note: At the start $a$ is freely chosen.

$C$ is always minimal separating for $A$ and $V \setminus (C \cup A)$. All nodes from $C$ have neighbours in $A$. 
Proof

- Algorithm to compute a chordal separator:
  - \( C := \emptyset \)
  - As long a component \( A \) in \( G[V \setminus C] \) exists with \( |A| > n/2 \) do:
    - \( C := \{ c \in C \mid \Gamma(c) \cap A \neq \emptyset \} \)
    - Choose \( a \in A \) with: \( C \subset \Gamma(a) \)
    - \( C := C \cup \{a\} \)

- There is at most one component \( A \) with: \( |A| > n/2 \).

- At each round, one node will be removed from that component.
Proof

- Algorithm to compute a chordal separator:
  - $C := \emptyset$
  - As long a component $A$ in $G[V \setminus C]$ exists with $|A| > n/2$ do:
    - $C := \{c \in C \mid \Gamma(c) \cap A \neq \emptyset\}$
    - Choose $a \in A$ with: $C \subset \Gamma(a)$
    - $C := C \cup \{a\}$

- There is at most one component $A$ with: $|A| > n/2$.
- At each round, one node will be removed from that component.
- There are at most $\lceil n/2 \rceil$ iterations.
Proof

- Algorithm to compute a chordal separator:
  - \( C := \emptyset \)
  - As long a component \( A \) in \( G[V \setminus C] \) exists with \( |A| > n/2 \) do:
    - \( C := \{ c \in C \mid \Gamma(c) \cap A \neq \emptyset \} \)
    - Choose \( a \in A \) with: \( C \subseteq \Gamma(a) \)
    - \( C := C \cup \{a\} \)

- There is at most one component \( A \) with: \( |A| > n/2 \).
- At each round, one node will be removed from that component.
- There are at most \( \lceil n/2 \rceil \) iterations.
- Show \( \exists a : C \subseteq \Gamma(a) \).
Proof

- Algorithm to compute a chordal separator:
  - $C := \emptyset$
  - As long a component $A$ in $G[V \setminus C]$ exists with $|A| > n/2$ do:
    - $C := \{c \in C \mid \Gamma(c) \cap A \neq \emptyset\}$
    - Choose $a \in A$ with: $C \subseteq \Gamma(a)$
    - $C := C \cup \{a\}$

- There is at most one component $A$ with: $|A| > n/2$.
- At each round, one node will be removed from that component.
- There are at most $\lceil n/2 \rceil$ iterations.
- Show $\exists a : C \subseteq \Gamma(a)$.

- Note:
Proof

- Algorithm to compute a chordal separator:
  - $C := \emptyset$
  - As long a component $A$ in $G[V \setminus C]$ exists with $|A| > n/2$ do:
    - $C := \{c \in C \mid \Gamma(c) \cap A \neq \emptyset\}$
    - Choose $a \in A$ with: $C \subseteq \Gamma(a)$
    - $C := C \cup \{a\}$

- There is at most one component $A$ with: $|A| > n/2$.
- At each round, one node will be removed from that component.
- There are at most $\lceil n/2 \rceil$ iterations.
- Show $\exists a : C \subseteq \Gamma(a)$.
- Note:
  - At the start $a$ is freely chosen.
Proof

- Algorithm to compute a chordal separator:
  - $C := \emptyset$
  - As long a component $A$ in $G[V \setminus C]$ exists with $|A| > n/2$ do:
    - $C := \{ c \in C \mid \Gamma(c) \cap A \neq \emptyset \}$
    - Choose $a \in A$ with: $C \subset \Gamma(a)$
    - $C := C \cup \{a\}$

- There is at most one component $A$ with: $|A| > n/2$.
- At each round, one node will be removed from that component.
- There are at most $\lceil n/2 \rceil$ iterations.
- Show $\exists a : C \subset \Gamma(a)$.

Note:
- At the start $a$ is freely chosen.
- $C$ is always minimal separating for $A$ and $V \setminus (C \cup A)$. 
Algorithm to compute a chordal separator:

- \( C := \emptyset \)
- As long a component \( A \) in \( G[V \setminus C] \) exists with \( |A| > n/2 \) do:
  - \( C := \{ c \in C \mid \Gamma(c) \cap A \neq \emptyset \} \)
  - Choose \( a \in A \) with: \( C \subset \Gamma(a) \)
  - \( C := C \cup \{ a \} \)

There is at most one component \( A \) with: \( |A| > n/2 \).

At each round, one node will be removed from that component.

There are at most \( \lceil n/2 \rceil \) iterations.

Show \( \exists a : C \subset \Gamma(a) \).

Note:

- At the start \( a \) is freely chosen.
- \( C \) is always minimal separating for \( A \) and \( V \setminus (C \cup A) \).
- All nodes from \( C \) have neighbours in \( A \).
Proof

- $C := \emptyset$
- As long a component $A$ in $G[V \setminus C]$ exists with $|A| > n/2$ do:
  - $C := \{c \in C \mid \Gamma(c) \cap A \neq \emptyset\}$
  - Choose $a \in A$ with: $C \subset \Gamma(a)$
  - $C := C \cup \{a\}$
- Show $\exists a : C \subset \Gamma(a)$. 
Proof

- \( C := \emptyset \)
- As long a component \( A \) in \( G[V \setminus C] \) exists with \(|A| > n/2\) do:
  - \( C := \{ c \in C | \Gamma(c) \cap A \neq \emptyset \} \)
  - Choose \( a \in A \) with: \( C \subseteq \Gamma(a) \)
  - \( C := C \cup \{ a \} \)

- Show \( \exists a : C \subseteq \Gamma(a) \).
- Let \( \rho = (a_1, a_2, \ldots, a_{|A|}, c_1, c_2, \ldots, c_{|C|}) \) be a PES for \( G[A \cup C] \).
Proof

- \( C := \emptyset \)

- As long a component \( A \) in \( G[V \setminus C] \) exists with \( |A| > n/2 \) do:
  - \( C := \{ c \in C \mid \Gamma(c) \cap A \neq \emptyset \} \)
  - Choose \( a \in A \) with: \( C \subset \Gamma(a) \)
  - \( C := C \cup \{ a \} \)

- Show \( \exists a : C \subset \Gamma(a) \).

- Let \( \rho = (a_1, a_2, \ldots, a_{|A|}, c_1, c_2, \ldots, c_{|C|}) \) be a PES for \( G[A \cup C] \).

- Consider now \( a = a_{|A|} \):
Proof

- $C := \emptyset$

- As long a component $A$ in $G[V \setminus C]$ exists with $|A| > n/2$ do:
  - $C := \{c \in C \mid \Gamma(c) \cap A \neq \emptyset\}$
  - Choose $a \in A$ with: $C \subset \Gamma(a)$
  - $C := C \cup \{a\}$

- Show $\exists a : C \subset \Gamma(a)$.

- Let $\rho = (a_1, a_2, \ldots, a_{|A|}, c_1, c_2, \ldots, c_{|C|})$ be a PES for $G[A \cup C]$.

- Consider now $a = a_{|A|}$:

- Each node from $C$ is connected by a path with $a$. 
Proof

- $C := \emptyset$
- As long a component $A$ in $G[V \setminus C]$ exists with $|A| > n/2$ do:
  - $C := \{c \in C \mid \Gamma(c) \cap A \neq \emptyset\}$
  - Choose $a \in A$ with: $C \subset \Gamma(a)$
  - $C := C \cup \{a\}$

- Show $\exists a : C \subset \Gamma(a)$.
- Let $\rho = (a_1, a_2, \ldots, a_{|A|}, c_1, c_2, \ldots, c_{|C|})$ be a PES for $G[A \cup C]$.
- Consider now $a = a_{|A|}$:
- Each node from $C$ is connected by a path with $a$.
- Thus each node from $C$ is directly connected with $a$. 
Proof

- $C := \emptyset$

- As long a component $A$ in $G[V \setminus C]$ exists with $|A| > n/2$ do:
  - $C := \{c \in C \mid \Gamma(c) \cap A \neq \emptyset\}$
  - Choose $a \in A$ with: $C \subset \Gamma(a)$
  - $C := C \cup \{a\}$

- Show $\exists a : C \subset \Gamma(a)$.

- Let $\rho = (a_1, a_2, \ldots, a_{|A|}, c_1, c_2, \ldots, c_{|C|})$ be a PES for $G[A \cup C]$.

- Consider now $a = a_{|A|}$:

  - Each node from $C$ is connected by a path with $a$.
  - Thus each node from $C$ is directly connected with $a$.

- Furthermore $\{a\} \cup C$ is a clique.
Proof

- $C := \emptyset$
- As long a component $A$ in $G[V \setminus C]$ exists with $|A| > n/2$ do:
  - $C := \{c \in C \mid \Gamma(c) \cap A \neq \emptyset\}$
  - Choose $a \in A$ with: $C \subset \Gamma(a)$
  - $C := C \cup \{a\}$

- Show $\exists a : C \subset \Gamma(a)$.
- Let $\rho = (a_1, a_2, \ldots, a_{|A|}, c_1, c_2, \ldots, c_{|C|})$ be a PES for $G[A \cup C]$.
- Consider now $a = a_{|A|}$:
  - Each node from $C$ is connected by a path with $a$.
  - Thus each node from $C$ is directly connected with $a$.
  - Furthermore $\{a\} \cup C$ is a clique.
- The computation could be done in time $O(n \cdot m)$. 
Proof

- \( C := \emptyset \)
- As long a component \( A \) in \( G[V \setminus C] \) exists with \( |A| > n/2 \) do:
  - \( C := \{ c \in C \mid \Gamma(c) \cap A \neq \emptyset \} \)
  - Choose \( a \in A \) with: \( C \subset \Gamma(a) \)
  - \( C := C \cup \{ a \} \)

- Show \( \exists a : C \subset \Gamma(a) \).
- Let \( \rho = (a_1, a_2, \ldots, a_{|A|}, c_1, c_2, \ldots, c_{|C|}) \) be a PES for \( G[A \cup C] \).
- Consider now \( a = a_{|A|} \):
  - Each node from \( C \) is connected by a path with \( a \).
  - Thus each node from \( C \) is directly connected with \( a \).
  - Furthermore \( \{ a \} \cup C \) is a clique.
  - The computation could be done in time \( O(n \cdot m) \).
  - Using an other algorithm a linear running-time is possible.
Definition (Clique-Separator)

Clique $C$ in $G = (V, E)$ is called Clique-Separator, iff $G[V \setminus C]$ is disconnected.
Definition (Clique-Separator)

Clique $C$ in $G = (V, E)$ is called Clique-Separator, iff $G[V \setminus C]$ is disconnected.

Definition (Clique-Separator-Tree)

A clique-separator-tree $T$ is defined recursively:

- If $G = (V, E)$ contains no clique-separator:
Definition (Clique-Separator)

Clique $C$ in $G = (V, E)$ is called Clique-Separator, iff $G[V \setminus C]$ is disconnected.

Definition (Clique-Separator-Tree)

A clique-separator-tree $T$ is defined recursively:

- If $G = (V, E)$ contains no clique-separator:
  - $T$ consists only of the node $w$. 
Definition (Clique-Separator)

Clique $C$ in $G = (V, E)$ is called Clique-Separator, iff $G[V \setminus C]$ is disconnected.

Definition (Clique-Separator-Tree)

A clique-separator-tree $T$ is defined recursively:

- If $G = (V, E)$ contains no clique-separator:
  - $T$ consists only of the node $w$.
  - To $w$ is the set $V$ associated.

The leaves of the clique-separator-tree are called atoms.
Introduction

Definition (Clique-Separator)

Clique $C$ in $G = (V, E)$ is called Clique-Separator, iff $G[V \setminus C]$ is disconnected.

Definition (Clique-Separator-Tree)

A clique-separator-tree $T$ is defined recursively:

- If $G = (V, E)$ contains no clique-separator:
  - $T$ consists only of the node $w$.
  - To $w$ is the set $V$ associated.

- If $G = (V, E)$ has a clique-separator $C$:
Introduction

**Definition (Clique-Separator)**

Clique $C$ in $G = (V, E)$ is called Clique-Separator, iff $G[V \setminus C]$ is disconnected.

**Definition (Clique-Separator-Tree)**

A clique-separator-tree $T$ is defined recursively:

- If $G = (V, E)$ contains no clique-separator:
  - $T$ consists only of the node $w$.
  - To $w$ is the set $V$ associated.

- If $G = (V, E)$ has a clique-separator $C$:
  - Let $A_1, A_2, \cdots, A_l$ be the components of $G[V \setminus C]$
Introduction

Definition (Clique-Separator)

Clique $C$ in $G = (V, E)$ is called Clique-Separator, iff $G[V \setminus C]$ is disconnected.

Definition (Clique-Separator-Tree)

A clique-separator-tree $T$ is defined recursively:

- If $G = (V, E)$ contains no clique-separator:
  - $T$ consists only of the node $w$.
  - To $w$ is the set $V$ associated.

- If $G = (V, E)$ has a clique-separator $C$:
  - Let $A_1, A_2, \ldots, A_l$ be the components of $G[V \setminus C]$
  - $T$ consists of the root $w$ and subtrees $T_1, T_2, \ldots, T_l$. 

Introduction

Definition (Clique-Separator)

Clique $C$ in $G = (V, E)$ is called Clique-Separator, iff $G[V \backslash C]$ is disconnected.

Definition (Clique-Separator-Tree)

A clique-separator-tree $T$ is defined recursively:

- If $G = (V, E)$ contains no clique-separator:
  - $T$ consists only of the node $w$.
  - To $w$ is the set $V$ associated.

- If $G = (V, E)$ has a clique-separator $C$:
  - Let $A_1, A_2, \ldots, A_l$ be the components of $G[V \backslash C]$
  - $T$ consists of the root $w$ and subtrees $T_1, T_2, \ldots, T_l$.
  - To a tree $T_i$ is the graph $G[A_i \cup C]$ associated.
Introduction

Definition (Clique-Separator)

Clique $C$ in $G = (V, E)$ is called Clique-Separator, iff $G[V \setminus C]$ is disconnected.

Definition (Clique-Separator-Tree)

A clique-separator-tree $T$ is defined recursively:

- If $G = (V, E)$ contains no clique-separator:
  - $T$ consists only of the node $w$.
  - To $w$ is the set $V$ associated.

- If $G = (V, E)$ has a clique-separator $C$:
  - Let $A_1, A_2, \ldots, A_l$ be the components of $G[V \setminus C]$
  - $T$ consists of the root $w$ and subtrees $T_1, T_2, \ldots, T_l$.
  - To a tree $T_i$ is the graph $G[A_i \cup C]$ associated.
  - To $w$ is the set $C$ associated.
**Definition (Clique-Separator)**

Clique $C$ in $G = (V, E)$ is called Clique-Separator, iff $G[V \setminus C]$ is disconnected.

**Definition (Clique-Separator-Tree)**

A clique-separator-tree $T$ is defined recursively:

- If $G = (V, E)$ contains no clique-separator:
  - $T$ consists only of the node $w$.
  - To $w$ is the set $V$ associated.

- If $G = (V, E)$ has a clique-separator $C$:
  - Let $A_1, A_2, \ldots, A_l$ be the components of $G[V \setminus C]$.
  - $T$ consists of the root $w$ and subtrees $T_1, T_2, \ldots, T_l$.
  - To a tree $T_i$ is the graph $G[A_i \cup C]$ associated.
  - To $w$ is the set $C$ associated.

The leaves of the clique-separator-tree are called atoms.
Basics, Motivation

- A clique-separator-tree has at most \( \binom{n}{2} - m \) atoms (Exercise).
A clique-separator-tree has at most \( \binom{n}{2} - m \) atoms (Exercise).

Each chordal graph has a clique-separator-tree, where all atoms are cliques.
Basics, Motivation

- A clique-separator-tree has at most $\binom{n}{2} - m$ atoms (Exercise).
- Each chordal graph has a clique-separator-tree, where all atoms are cliques.
- If the atoms are “simple”, then many problems become easy solvable.
Basics, Motivation

- A clique-separator-tree has at most $\binom{n}{2} - m$ atoms (Exercise).
- Each chordal graph has a clique-separator-tree, where all atoms are cliques.
- If the atoms are "simple", then many problems become easy solvable.
- We will now introduce the MES, which is similar to PES.
Reminder

**Definition**

A node is called simplicial, iff all its neighbours are connected by an edge.

**Theorem**

Each Clique has a simplicial node and each chordal graph, who is not a clique, has two simplicial nodes, which are not connected.

**Definition**

Let $G = (V, E)$ be a graph with $|V| = n$. A total ordering $\rho : V \mapsto \{1, \ldots, n\}$ is called perfect node-elimination scheme, iff each node $v$ is a simplicial node in $G[\{u \in V \mid \rho(u) \geq \rho(v)\}]$.

**Theorem**

A graph is chordal, iff it has a PES.
Definition (Fill-in)

Let $G = (V, E)$ be a graph with $|V| = n$ and $\rho : V \mapsto \{1, \ldots, n\}$ an ordering of the nodes. The fill-in for $\rho$ is:

$$F_\rho := \left\{ \{v, w\} : \begin{array}{l} v \neq w \land \{v, w\} \notin E \land \text{there is a path } v = x_1x_2 \ldots x_l = w \text{ with:} \\
\rho(x_i) < \min(\rho(v), \rho(w)) \forall i = 2, 3, \ldots, l - 1 \end{array} \right\}$$

- Notation: $G_\rho = (V, E \cup F_\rho)$
Definition (Fill-in)

Let $G = (V, E)$ be a graph with $|V| = n$ and $\rho : V \mapsto \{1, \ldots, n\}$ an ordering of the nodes. The fill-in for $\rho$ is:

$$F_\rho := \left\{ \{v, w\} : \begin{array}{l}
v \neq w \land \{v, w\} \notin E \land \\
\text{there is a path } v = x_1x_2 \ldots x_l = w \text{ with:} \\
\rho(x_i) < \min(\rho(v), \rho(w)) \forall i = 2, 3, \ldots, l - 1
\end{array} \right\}$$

- Notation: $G_\rho = (V, E \cup F_\rho)$
- Any ordering $\rho$ is a PES for $G_\rho$. 
**Definition (Fill-in)**

Let $G = (V, E)$ be a graph with $|V| = n$ and $\rho : V \mapsto \{1, \ldots, n\}$ an ordering of the nodes. The fill-in for $\rho$ is:

$$F_\rho := \left\{ \{v, w\} : v \neq w \land \{v, w\} \not\in E \land \right.$$ 

there is a path $v = x_1 x_2 \ldots x_l = w$ with:

$$\rho(x_i) < \min(\rho(v), \rho(w)) \forall i = 2, 3, \ldots, l - 1$$

- **Notation**: $G_\rho = (V, E \cup F_\rho)$
- Any ordering $\rho$ is a PES for $G_\rho$.
- The fill-in for $\rho$ in $G_\rho$ is the empty set.
Definition (Fill-in)

Let $G = (V, E)$ be a graph with $|V| = n$ and $\rho : V \mapsto \{1, \ldots, n\}$ an ordering of the nodes. The fill-in for $\rho$ is:

$$F_\rho := \left\{ \{v, w\} : v \neq w \land \{v, w\} \notin E \land \right.$$ 

$$\exists \text{ a path } v = x_1x_2\ldots x_l = w \text{ with: }$$

$$\rho(x_i) < \min(\rho(v), \rho(w)) \forall i = 2, 3, \ldots, l - 1$$

- Notation: $G_\rho = (V, E \cup F_\rho)$
- Any ordering $\rho$ is a PES for $G_\rho$.
- The fill-in for $\rho$ in $G_\rho$ is the empty set.
- Thus $G_\rho$ is chordal.
Fill-In

Definition (Fill-in)

Let $G = (V, E)$ be a graph with $|V| = n$ and $\rho : V \mapsto \{1, \ldots, n\}$ an ordering of the nodes. The fill-in for $\rho$ is:

$$F_\rho := \left\{ \{v, w\} : v \neq w \land \{v, w\} \notin E \land \right.$$ 
$$\left. \text{there is a path } v = x_1 x_2 \ldots x_l = w \text{ with:} \right.$$ 
$$\rho(x_i) < \min(\rho(v), \rho(w)) \forall i = 2, 3, \ldots, l - 1 \right\}$$

- Notation: $G_\rho = (V, E \cup F_\rho)$
- Any ordering $\rho$ is a PES for $G_\rho$.
- The fill-in for $\rho$ in $G_\rho$ is the empty set.
- Thus $G_\rho$ is chordal.
- $\Gamma_{\rho,F}(v) := \{w \mid \{v, w\} \in E \cup F \land \rho(w) > \rho(v)\}$
Definition (Fill-in)

Let $G = (V, E)$ be a graph with $|V| = n$ and $\rho : V \mapsto \{1, \ldots, n\}$ an ordering of the nodes. The fill-in for $\rho$ is:

$$F_\rho := \left\{ \{v, w\} : v \neq w \land \{v, w\} \not\in E \land \text{there is a path } v = x_1 x_2 \ldots x_l = w \text{ with: } \rho(x_i) < \min(\rho(v), \rho(w)) \forall i = 2, 3, \ldots, l - 1 \right\}$$

- Notation: $G_\rho = (V, E \cup F_\rho)$
- Any ordering $\rho$ is a PES for $G_\rho$.
- The fill-in for $\rho$ in $G_\rho$ is the empty set.
- Thus $G_\rho$ is chordal.
- $\Gamma_{\rho,F}(v) := \{w \mid \{v, w\} \in E \cup F \land \rho(w) > \rho(v)\}$
- $m_F(v)$ the node $u$ with: $\rho(u) = \min\{\rho(w) \mid w \in \Gamma_{\rho,F}(v)\}$. 
Lemma

Let $G = (V, E)$ be a graph and $\rho$ a ordering. Then is the fill-in $F_\rho$ the smallest set $F$, such that for all $v \in V$ holds:

$$\Gamma_{\rho,F}(v) \subseteq \Gamma_{\rho,F}(m_F(v)) \cup m_F(v)$$

Proof:

- Show that for $F = F_\rho$ the above equation holds.
Lemma

Let $G = (V, E)$ be graph and $\rho$ a ordering. Then is the fill-in $F_\rho$ the smallest set $F$, such that for all $v \in V$ holds:

$$\Gamma_{\rho,F}(v) \subseteq \Gamma_{\rho,F}(m_F(v)) \cup m_F(v)$$

Proof:

- Show that for $F = F_\rho$ the above equation holds.
  - Let $v$ be a node.
Lemma

Let $G = (V, E)$ be graph and $\rho$ a ordering.
Then is the fill-in $F_\rho$ the smallest set $F$, such that for all $v \in V$ holds:

$$\Gamma_{\rho,F}(v) \subseteq \Gamma_{\rho,F}(m_F(v)) \cup m_F(v)$$

Proof:

- Show that for $F = F_\rho$ the above equation holds.
  - Let $v$ be a node.
  - Let $w \in \Gamma_{\rho,F_\rho}(v)$ and $w \neq m_F(v) = x$. 

Lemma

Let $G = (V, E)$ be graph and $\rho$ a ordering. Then is the fill-in $F_\rho$ the smallest set $F$, such that for all $v \in V$ holds:

$$\Gamma_{\rho,F}(v) \subseteq \Gamma_{\rho,F}(m_F(v)) \cup m_F(v)$$

Proof:

- Show that for $F = F_\rho$ the above equation holds.
  - Let $v$ be a node.
  - Let $w \in \Gamma_{\rho,F_\rho}(v)$ and $w \neq m_F(v) = x$.
  - Then is $m_F(v), v, w$ a path in $G_\rho$ with $\rho(v) < \min(\rho(m_F(v)), \rho(w))$. 
Results

Lemma

Let \( G = (V, E) \) be graph and \( \rho \) a ordering. Then is the fill-in \( F_\rho \) the smallest set \( F \), such that for all \( v \in V \) holds:

\[
\Gamma_{\rho, F}(v) \subseteq \Gamma_{\rho, F}(m_F(v)) \cup m_F(v)
\]

Proof:

- Show that for \( F = F_\rho \) the above equation holds.
  - Let \( v \) be a node.
  - Let \( w \in \Gamma_{\rho, F_\rho}(v) \) and \( w \neq m_F(v) = x \).
  - Then is \( m_F(v), v, w \) a path in \( G_\rho \) with \( \rho(v) < \min(\rho(m_F(v)), \rho(w)) \).
  - Thus \( \{w, m_F(v)\} \in E \cup F_\rho \) holds.
Lemma

Let $G = (V, E)$ be graph and $\rho$ a ordering. Then is the fill-in $F_\rho$ the smallest set $F$, such that for all $v \in V$ holds:

$$\Gamma_{\rho,F}(v) \subseteq \Gamma_{\rho,F}(m_F(v)) \cup m_F(v)$$

Proof:

- Show that for $F = F_\rho$ the above equation holds.
  - Let $v$ be a node.
  - Let $w \in \Gamma_{\rho,F_\rho}(v)$ and $w \neq m_F(v) = x$.
  - Then is $m_F(v)$, $v$, $w$ a path in $G_\rho$ with $\rho(v) < \min(\rho(m_F(v)), \rho(w))$.
  - Thus $\{w, m_F(v)\} \in E \cup F_\rho$ holds.
  - And $w \in \Gamma_{\rho,F_\rho}(m_F(v))$ holds.
Proof (Let $F$ be as defined, show that $F_\rho \subseteq F$ holds)

- Show by induction over $i$:
  $\forall \{v, w\} \in F_\rho$ with $\rho(v) \leq i$: $\{v, w\} \in F$

$\Gamma_{\rho,F}(v) \subseteq \Gamma_{\rho,F}(m_F(v)) \cup m_F(v)$
Proof (Let $F$ be as defined, show that $F_\rho \subseteq F$ holds)

- Show by induction over $i$:
  $\forall \{v, w\} \in F_\rho$ with $\rho(v) \leq i$: $\{v, w\} \in F$

- Assume the above holds for $i \leq i_0$. 

\[ \Gamma_{\rho,F}(v) \subseteq \Gamma_{\rho,F}(m_F(v)) \cup m_F(v) \]
Proof (Let $F$ be as defined, show that $F_\rho \subseteq F$ holds)

- Show by induction over $i$:
  \[
  \forall \{v, w\} \in F_\rho \text{ with } \rho(v) \leq i : \{v, w\} \in F
  \]
- Assume the above holds for $i \leq i_0$.
- Let $\{v, w\} \in F_\rho$ with $\rho(v) = i_0 + 1 \leq \rho(w)$. 

\[
\Gamma_{\rho,F}(v) \subseteq \Gamma_{\rho,F}(m_F(v)) \cup m_F(v)
\]
Proof (Let $F$ be as defined, show that $F_\rho \subseteq F$ holds)

- Show by induction over $i$:
  \[
  \forall \{v, w\} \in F_\rho \text{ with } \rho(v) \leq i: \{v, w\} \in F
  \]

- Assume the above holds for $i \leq i_0$.

- Let $\{v, w\} \in F_\rho \text{ with } \rho(v) = i_0 + 1 \leq \rho(w)$.

- Thus there is a path $v = x_1 x_2 \ldots x_k = w$ in $G_\rho = (V, E \cup F_\rho)$ with:

\[
\Gamma_{\rho,F}(v) \subseteq \Gamma_{\rho,F(m_F(v))} \cup m_F(v)
\]
Proof (Let $F$ be as defined, show that $F_{\rho} \subseteq F$ holds)

- Show by induction over $i$:
  $\forall \{v, w\} \in F_{\rho}$ with $\rho(v) \leq i$: $\{v, w\} \in F$

- Assume the above holds for $i \leq i_0$.

- Let $\{v, w\} \in F_{\rho}$ with $\rho(v) = i_0 + 1 \leq \rho(w)$.

- Thus there is a path $v = x_1x_2 \ldots x_k = w$ in $G_{\rho} = (V, E \cup F_{\rho})$ with:
  - $k \geq 3$ and $\rho(x_j) < \min(\rho(v), \rho(w))$ for $j = 2, 3, \ldots k - 1$. 

\[ \Gamma_{\rho,F}(v) \subseteq \Gamma_{\rho,F}(m_F(v)) \cup m_F(v) \]
Proof (Let $F$ be as defined, show that $F_\rho \subseteq F$ holds)

- Show by induction over $i$:
  \[ \forall \{v, w\} \in F_\rho \text{ with } \rho(v) \leq i: \{v, w\} \in F \]

- Assume the above holds for $i \leq i_0$.

- Let $\{v, w\} \in F_\rho$ with $\rho(v) = i_0 + 1 \leq \rho(w)$.

- Thus there is a path $v = x_1x_2\ldots x_k = w$ in $G_\rho = (V, E \cup F_\rho)$ with:
  - $k \geq 3$ and $\rho(x_j) < \min(\rho(v), \rho(w))$ for $j = 2, 3, \ldots k - 1$.

- Let $k$ be minimal.
Proof (Let $F$ be as defined, show that $F_\rho \subseteq F$ holds)

- Show by induction over $i$:
  \[\forall \{v, w\} \in F_\rho \text{ with } \rho(v) \leq i : \{v, w\} \in F\]
- Assume the above holds for $i \leq i_0$.
- Let $\{v, w\} \in F_\rho$ with $\rho(v) = i_0 + 1 \leq \rho(w)$.
- Thus there is a path $v = x_1 x_2 \ldots x_k = w$ in $G_\rho = (V, E \cup F_\rho)$ with:
  - $k \geq 3$ and $\rho(x_j) < \min(\rho(v), \rho(w))$ for $j = 2, 3, \ldots k - 1$.
  - Let $k$ be minimal.
  - If $k > 3$ holds, the let $l \geq 2$ be with $\rho(x_l) \geq \rho(x_j)$ for $j = 2, 3, \ldots k - 1$.
Proof (Let $F$ be as defined, show that $F_\rho \subseteq F$ holds)

Show by induction over $i$:
\[
\forall \{v, w\} \in F_\rho \text{ with } \rho(v) \leq i: \{v, w\} \in F
\]

Assume the above holds for $i \leq i_0$.

Let $\{v, w\} \in F_\rho$ with $\rho(v) = i_0 + 1 \leq \rho(w)$.

Thus there is a path $v = x_1x_2 \ldots x_k = w$ in $G_\rho = (V, E \cup F_\rho)$ with:
\[
k \geq 3 \text{ and } \rho(x_j) < \min(\rho(v), \rho(w)) \text{ for } j = 2, 3, \ldots k - 1.
\]

Let $k$ be minimal.

If $k > 3$ holds, the let $l \geq 2$ be with $\rho(x_l) \geq \rho(x_j)$ for $j = 2, 3, \ldots k - 1$.

Then is $v = x_1, x_2, \ldots, x_l$ a path in $G_\rho$ with $\rho(x_j) < \min(\rho(v), \rho(w))$ for $j = 2, 3, \ldots l - 1$. 

\[
\Gamma_{\rho,F}(v) \subseteq \Gamma_{\rho,F(m_F(v))} \cup m_F(v)
\]
Proof (Let $F$ be as defined, show that $F_\rho \subseteq F$ holds)

- Show by induction over $i$:
  $\forall \{v, w\} \in F_\rho$ with $\rho(v) \leq i$: $\{v, w\} \in F$

- Assume the above holds for $i \leq i_0$.

- Let $\{v, w\} \in F_\rho$ with $\rho(v) = i_0 + 1 \leq \rho(w)$.

- Thus there is a path $v = x_1 x_2 \ldots x_k = w$ in $G_\rho = (V, E \cup F_\rho)$ with:
  $k \geq 3$ and $\rho(x_j) < \min(\rho(v), \rho(w))$ for $j = 2, 3, \ldots k - 1$.

- Let $k$ be minimal.

- If $k > 3$ holds, let $l \geq 2$ be with $\rho(x_l) \geq \rho(x_j)$ for $j = 2, 3, \ldots k - 1$.

- Then is $v = x_1, x_2, \ldots, x_l$ a path in $G_\rho$ with $\rho(x_j) < \min(\rho(v), \rho(w))$ for $j = 2, 3, \ldots l - 1$.

- Thus $\{v, x_l\} \in F_\rho$ holds.
Proof (Let $F$ be as defined, show that $F_\rho \subseteq F$ holds)

- Show by induction over $i$:
  $\forall \{v, w\} \in F_\rho \text{ with } \rho(v) \leq i : \{v, w\} \in F$
- Assume the above holds for $i \leq i_0$.
- Let $\{v, w\} \in F_\rho$ with $\rho(v) = i_0 + 1 \leq \rho(w)$.
- Thus there is a path $v = x_1x_2 \ldots x_k = w$ in $G_\rho = (V, E \cup F_\rho)$ with:
  - $k \geq 3$ and $\rho(x_j) < \min(\rho(v), \rho(w))$ for $j = 2, 3, \ldots k - 1$.
  - Let $k$ be minimal.
  - If $k > 3$ holds, let $l \geq 2$ be with $\rho(x_l) \geq \rho(x_j)$ for $j = 2, 3, \ldots k - 1$.
  - Then is $v = x_1, x_2, \ldots, x_l$ a path in $G_\rho$ with $\rho(x_j) < \min(\rho(v), \rho(w))$ for $j = 2, 3, \ldots l - 1$.
  - Thus $\{v, x_l\} \in F_\rho$ holds.
- This is a contradiction to the minimality of the path.
Proof (Let $F$ be a set satisfying the above equation, show that $F_\rho \subseteq F$ holds)

Let $k = 3$ and $u = x_2$ with: $v, w \in \Gamma_{\rho,F_{\rho}}(u)$.

\[ \Gamma_{\rho,F}(v) \subseteq \Gamma_{\rho,F}(m_F(v)) \cup m_F(v) \]
\[ v = x_1x_2x_3 = w \]
\[ \rho(x_2) < \min(\rho(v), \rho(w)) \]

\[ \Sigma = 0 \]
Proof (Let $F$ be a set satisfying the above equation, show that $F_\rho \subseteq F$ holds)

- Let $k = 3$ and $u = x_2$ with: $v, w \in \Gamma_{\rho,F}(u)$.
- Choose $u$ such that $\rho(u)$ is maximal.

\[
\Gamma_{\rho,F}(v) \subseteq \Gamma_{\rho,F}(m_F(v)) \cup m_F(v)
\]

$v = x_1x_2x_3 = w$

$\rho(x_2) < \min(\rho(v), \rho(w))$
Proof (Let $F$ be a set satisfying the above equation, show that $F_\rho \subseteq F$ holds)

- Let $k = 3$ and $u = x_2$ with: $v, w \in \Gamma_{\rho,F_\rho}(u)$.
- Choose $u$ such that $\rho(u)$ is maximal.
- By induction and $\rho(u) < \rho(v)$ does $v, w \in \Gamma_{\rho,F}(u)$ hold.
Proof (Let $F$ be a set satisfying the above equation, show that $F_{\rho} \subseteq F$ holds)

- Let $k = 3$ and $u = x_2$ with: $v, w \in \Gamma_{\rho,F_{\rho}}(u)$.
- Choose $u$ such that $\rho(u)$ is maximal.
- By induction and $\rho(u) < \rho(v)$ does $v, w \in \Gamma_{\rho,F}(u)$ hold.
- If $v \neq m_F(u)$ then we would get $v, w \in \Gamma_{\rho,F}(m_F(u))$. 

$\Gamma_{\rho,F}(v) \subseteq \Gamma_{\rho,F}(m_F(v)) \cup m_F(v)$

$v = x_1x_2x_3 = w$

$\rho(x_2) < \min(\rho(v), \rho(w))$
Proof (Let $F$ be a set satisfying the above equation, show that $F_{\rho} \subseteq F$ holds)

- Let $k = 3$ and $u = x_2$ with: $v, w \in \Gamma_{\rho,F_{\rho}}(u)$.
- Choose $u$ such that $\rho(u)$ is maximal.
- By induction and $\rho(u) < \rho(v)$ does $v, w \in \Gamma_{\rho,F}(u)$ hold.
- If $v \neq m_{F}(u)$ then we would get $v, w \in \Gamma_{\rho,F}(m_{F}(u))$.
- But this is a contradiction to the maximality of $\rho(u)$.

$$\Gamma_{\rho,F}(v) \subseteq \Gamma_{\rho,F}(m_{F}(v)) \cup m_{F}(v)$$

$v = x_1x_2x_3 = w$

$\rho(x_2) < \min(\rho(v), \rho(w))$
Proof (Let $F$ be a set satisfying the above equation, show that $F_\rho \subseteq F$ holds)

- Let $k = 3$ and $u = x_2$ with: $v, w \in \Gamma_{\rho,F}(u)$.
- Choose $u$ such that $\rho(u)$ is maximal.
- By induction and $\rho(u) < \rho(v)$ does $v, w \in \Gamma_{\rho,F}(u)$ hold.
- If $v \neq m_F(u)$ then we would get $v, w \in \Gamma_{\rho,F}(m_F(u))$.
- But this is a contradiction to the maximality of $\rho(u)$.
- Thus we have $v = m_F(u)$.

\[
\Gamma_{\rho,F}(v) \subseteq \Gamma_{\rho,F}(m_F(v)) \cup m_F(v)
\]

$v = x_1 x_2 x_3 = w$

\[\rho(x_2) < \min(\rho(v), \rho(w))\]
Proof (Let $F$ be a set satisfying the above equation, show that $F_\rho \subseteq F$ holds)

- Let $k = 3$ and $u = x_2$ with: $v, w \in \Gamma_{\rho,F}(u)$.
- Choose $u$ such that $\rho(u)$ is maximal.
- By induction and $\rho(u) < \rho(v)$ does $v, w \in \Gamma_{\rho,F}(u)$ hold.
- If $v \neq m_F(u)$ then we would get $v, w \in \Gamma_{\rho,F}(m_F(u))$.
- But this is a contradiction to the maximality of $\rho(u)$.
- Thus we have $v = m_F(u)$.
- But then is $w \in \Gamma_{\rho,F}(m_F(u))$. 

\[
\Gamma_{\rho,F}(v) \subseteq \Gamma_{\rho,F}(m_F(v)) \cup m_F(v)
\]

\[
v = x_1x_2x_3 = w
\]

\[
\rho(x_2) < \min(\rho(v), \rho(w))
\]
Proof (Let $F$ be a set satisfying the above equation, show that $F_\rho \subseteq F$ holds)

- Let $k = 3$ and $u = x_2$ with: $v, w \in \Gamma_{\rho,F}(u)$.
- Choose $u$ such that $\rho(u)$ is maximal.
- By induction and $\rho(u) < \rho(v)$ does $v, w \in \Gamma_{\rho,F}(u)$ hold.
- If $v \neq m_F(u)$ then we would get $v, w \in \Gamma_{\rho,F}(m_F(u))$.
- But this is a contradiction to the maximality of $\rho(u)$.
- Thus we have $v = m_F(u)$.
- But then is $w \in \Gamma_{\rho,F}(m_F(u))$.
- And also $\{v, w\} = \{m_F(u), w\} \in F$. 

\[
\Gamma_{\rho,F}(v) \subseteq \Gamma_{\rho,F}(m_F(v)) \cup m_F(v)
\]
\[
v = x_1 x_2 x_3 = w,
\rho(x_2) < \min(\rho(v), \rho(w))
\]
Proof (Let $F$ be a set satisfying the above equation, show that $F_\rho \subseteq F$ holds)

- Let $k = 3$ and $u = x_2$ with: $v, w \in \Gamma_{\rho,F}(u)$.
- Choose $u$ such that $\rho(u)$ is maximal.
- By induction and $\rho(u) < \rho(v)$ does $v, w \in \Gamma_{\rho,F}(u)$ hold.
- If $v \neq m_F(u)$ then we would get $v, w \in \Gamma_{\rho,F}(m_F(u))$.
- But this is a contradiction to the maximality of $\rho(u)$.
- Thus we have $v = m_F(u)$.
- But then is $w \in \Gamma_{\rho,F}(m_F(u))$.
- And also $\{v, w\} = \{m_F(u), w\} \in F$.
- Thus we get by induction: $F_\rho \subseteq F$.

\[
\begin{align*}
\Gamma_{\rho,F}(v) &\subseteq \Gamma_{\rho,F}(m_F(v)) \cup m_F(v) \\
v = x_1 x_2 x_3 = w \\
\rho(x_2) &< \min(\rho(v), \rho(w))
\end{align*}
\]
Lemma

For a graph $G$ and a ordering $\rho$ is the fill-in computable in time $O(n + m + |F_\rho|)$.

Algorithm $Fill\_In(G, \rho)$

- For all $v \in V$ do:
**Lemma**

For a graph $G$ and a ordering $\rho$ is the fill-in computable in time $O(n + m + |F_\rho|)$.

Algorithm $Fill\_ln(G, \rho)$

- For all $v \in V$ do:
  - $A(v) := \Gamma_{\rho,\emptyset}(v) = \{w \in \Gamma(V) \mid \rho(w) > \rho(v)\}$
Theorems

Lemma

For a graph $G$ and a ordering $\rho$ is the fill-in computable in time $O(n + m + |F_{\rho}|)$.

Algorithm $Fill\_In(G, \rho)$

- For all $v \in V$ do:
  - $A(v) := \Gamma_{\rho,\emptyset}(v) = \{ w \in \Gamma(V) \mid \rho(w) > \rho(v) \}$
- For $i := 1$ bis $n - 1$ do:
Theorems

Lemma

For a graph $G$ and a ordering $\rho$ is the fill-in computable in time $O(n + m + |F_\rho|)$.

Algorithm $Fill\_In(G, \rho)$

1. For all $v \in V$ do:
   - $A(v) := \Gamma_{\rho,\emptyset}(v) = \{w \in \Gamma(V) | \rho(w) > \rho(v)\}$
2. For $i := 1$ bis $n - 1$ do:
   - $v := \rho^{-1}(i)$
Lemma

For a graph $G$ and a ordering $\rho$ is the fill-in computable in time $O(n + m + |F_\rho|)$.

Algorithm $Fill\_In(G, \rho)$

- For all $v \in V$ do:
  - $A(v) := \Gamma_{\rho,\emptyset}(v) = \{w \in \Gamma(V) | \rho(w) > \rho(v)\}$
- For $i := 1$ bis $n - 1$ do:
  - $v := \rho^{-1}(i)$
  - $m(v) := \rho^{-1}(\min\{\rho(u) | u \in A(v)\})$
**Lemma**

*For a graph $G$ and a ordering $\rho$ is the fill-in computable in time $O(n + m + |F_\rho|)$.*

**Algorithm $\text{Fill\_In}(G, \rho)$**

- For all $v \in V$ do:
  
  - $A(v) := \Gamma_{\rho,\emptyset}(v) = \{w \in \Gamma(V) \mid \rho(w) > \rho(v)\}$

- For $i := 1$ bis $n - 1$ do:
  
  - $v := \rho^{-1}(i)$
  - $m(v) := \rho^{-1}(\min\{\rho(u) \mid u \in A(v)\})$
  - $A(m(v)) := A(m(v)) \cup \{w \in A(v) \mid w \neq m(v)\}$
Theorems

**Lemma**

For a graph $G$ and a ordering $\rho$ is the fill-in computable in time $O(n + m + |F_\rho|)$.

**Algorithm Fill\_In($G, \rho$)**

- For all $v \in V$ do:
  - $A(v) := \Gamma_{\rho, \emptyset}(v) = \{w \in \Gamma(V) | \rho(w) > \rho(v)\}$

- For $i := 1$ bis $n - 1$ do:
  - $v := \rho^{-1}(i)$
  - $m(v) := \rho^{-1}(\min\{\rho(u) | u \in A(v)\})$
  - $A(m(v)) := A(m(v)) \cup \{w \in A(v) | w \neq m(v)\}$

- $F_\rho = \emptyset$
Lemma

For a graph $G$ and a ordering $\rho$ is the fill-in computable in time $O(n + m + |F_\rho|)$.

Algorithm $Fill\_In(G, \rho)$

1. For all $v \in V$ do:
   - $A(v) := \Gamma_{\rho, \emptyset}(v) = \{w \in \Gamma(V) \mid \rho(w) > \rho(v)\}$

2. For $i := 1$ bis $n - 1$ do:
   - $v := \rho^{-1}(i)$
   - $m(v) := \rho^{-1}(\min\{\rho(u) \mid u \in A(v)\})$
   - $A(m(v)) := A(m(v)) \cup \{w \in A(v) \mid w \neq m(v)\}$

3. $F_\rho = \emptyset$

4. For all $v \in V$ and $w \in A(v) \setminus \Gamma(v)$ do:
Lemma

For a graph $G$ and a ordering $\rho$ is the fill-in computable in time $O(n + m + |F_\rho|)$.

Algorithm $\text{Fill\_In}(G, \rho)$

- For all $v \in V$ do:
  - $A(v) := \Gamma_{\rho, \emptyset}(v) = \{w \in \Gamma(V) \mid \rho(w) > \rho(v)\}$

- For $i := 1$ bis $n - 1$ do:
  - $v := \rho^{-1}(i)$
  - $m(v) := \rho^{-1}(\min\{\rho(u) \mid u \in A(v)\})$
  - $A(m(v)) := A(m(v)) \cup \{w \in A(v) \mid w \neq m(v)\}$

- $F_\rho = \emptyset$

- For all $v \in V$ and $w \in A(v) \setminus \Gamma(v)$ do:
  - $F_\rho = F_\rho \cup \{v, w\}$
Definition

An ordering $\rho$ for $G = (V, E)$ is called minimal elimination schema (MES), iff the Fill-in $F_{\rho}$ is minimal, i.e. $\nexists \rho' : F_{\rho'} \subset F_{\rho}$.

- Aim: clique-separator for $G$ should also be clique-separator for $G_\rho$, if $\rho$ is a MES.
Definition

An ordering $\rho$ for $G = (V, E)$ is called minimal elimination schema (MES), iff the Fill-in $F_\rho$ is minimal, i.e. $\not\exists \rho' : F_{\rho'} \subset F_\rho$.

- **Aim:** clique-separator for $G$ should also be clique-separator for $G_\rho$, if $\rho$ is a MES.
- **Note:** to find the smallest MES is in NPC.
**Definition**

An ordering $\rho$ for $G = (V, E)$ is called minimal elimination schema (MES), iff the Fill-in $F_\rho$ is minimal, i.e. $\nexists \rho' : F_\rho \subset F_\rho$.

- **Aim:** clique-separator for $G$ should also be clique-separator for $G_\rho$, if $\rho$ is a MES.
- **Note:** to find the smallest MES is in NPC.
- **But here we only need a MES.**
An ordering $\rho$ for $G = (V, E)$ is called minimal elimination schema (MES), iff the Fill-in $F_\rho$ is minimal, i.e. $\forall \rho' : F_{\rho'} \subset F_\rho$.

- Aim: clique-separator for $G$ should also be clique-separator for $G_\rho$, if $\rho$ is a MES.
- Note: to find the smallest MES is in NPC.
- But here we only need a MES.
- This is possible in polynomial time:
Definition

An ordering $\rho$ for $G = (V, E)$ is called minimal elimination schema (MES), iff the Fill-in $F_\rho$ is minimal, i.e. $\not\exists \rho' : F_{\rho'} \subset F_\rho$.

- Aim: clique-separator for $G$ should also be clique-separator for $G_\rho$, if $\rho$ is a MES.
- Note: to find the smallest MES is in NPC.
- But here we only need a MES.
- This is possible in polynomial time:
  - Lexicographical breath-first-search
An ordering $\rho$ for $G = (V, E)$ is called minimal elimination schema (MES), iff the Fill-in $F_\rho$ is minimal, i.e. $\not\exists \rho' : F_\rho' \subset F_\rho$.

Aim: clique-separator for $G$ should also be clique-separator for $G_\rho$, if $\rho$ is a MES.

Note: to find the smallest MES is in NPC.

But here we only need a MES.

This is possible in polynomial time:
- Lexicographical breath-first-search
- Comparing sets by their lexicographical order:
Definition

An ordering $\rho$ for $G = (V, E)$ is called minimal elimination schema (MES), iff the Fill-in $F_\rho$ is minimal, i.e. $\not\exists \rho' : F_{\rho'} \subset F_\rho$.

- Aim: clique-separator for $G$ should also be clique-separator for $G_\rho$, if $\rho$ is a MES.
- Note: to find the smallest MES is in NPC.
- But here we only need a MES.
- This is possible in polynomial time:
  - Lexicographical breath-first-search
  - Comparing sets by their lexicographical order:
  - Thus $\{2, 5\} < \{2, 4, 5\}$
Definition

An ordering \( \rho \) for \( G = (V, E) \) is called minimal elimination schema (MES), iff the Fill-in \( F_\rho \) is minimal, i.e.
\[
\forall \rho' : F_{\rho'} \subset F_\rho.
\]

- Aim: clique-separator for \( G \) should also be clique-separator for \( G_\rho \), if \( \rho \) is a MES.
- Note: to find the smallest MES is in NPC.
- But here we only need a MES.
- This is possible in polynomial time:
  - Lexicographical breath-first-search
  - Comparing sets by their lexicographical order:
    - Thus \( \{2, 5\} < \{2, 4, 5\} \)
    - And \( \emptyset < \{2\} \)
Algorithm

For all $v \in V$ do:
Algorithm

- For all $v \in V$ do:
  - $pr(v) := \emptyset$

Proof of correctness is complicated.
Algorithm

For all \( v \in V \) do:

- \( pr(v) := \emptyset \)
- \( \rho(v) := 0 \)
Algorithm

For all $v \in V$ do:

- $pr(v) := \emptyset$
- $\rho(v) := 0$

For $i := n$ down to 1 do:
Algorithm

- For all $v \in V$ do:
  - $pr(v) := \emptyset$
  - $\rho(v) := 0$

- For $i := n$ down to 1 do:
  - Choose node $v$ with $pr(v)$ maximal and $\rho(v) = 0$
Algorithm

- For all $v \in V$ do:
  - $pr(v) := \emptyset$
  - $\rho(v) := 0$

- For $i := n$ down to 1 do:
  - Choose node $v$ with $pr(v)$ maximal and $\rho(v) = 0$
  - $\rho(v) := i$

Proof of correctness is complicated.

Running-time $O(n(m + n))$.
Algorithm

For all $v \in V$ do:

- $pr(v) := \emptyset$
- $\rho(v) := 0$

For $i := n$ down to 1 do:

- Choose node $v$ with $pr(v)$ maximal and $\rho(v) = 0$
- $\rho(v) := i$
- For all $w$ with $\rho(w) = 0$ do
Algorithm

- For all \( v \in V \) do:
  - \( pr(v) := \emptyset \)
  - \( \rho(v) := 0 \)

- For \( i := n \) down to 1 do:
  - Choose node \( v \) with \( pr(v) \) maximal and \( \rho(v) = 0 \)
  - \( \rho(v) := i \)
  - For all \( w \) with \( \rho(w) = 0 \) do
    - If there is a path \( v = v_1, v_2, \ldots, v_k = w \) with:
      - \( \rho(v_i) = 0 \) and \( pr(v_j) < pr(v_w) \)
      - for \( j = 2, 3, \ldots, k - 1 \), do:
Algorithm

- For all $v \in V$ do:
  - $pr(v) := \emptyset$
  - $\rho(v) := 0$

- For $i := n$ down to 1 do:
  - Choose node $v$ with $pr(v)$ maximal and $\rho(v) = 0$
  - $\rho(v) := i$
  - For all $w$ with $\rho(w) = 0$ do
    - If there is a path $v = v_1, v_2, \ldots, v_k = w$ with:
      - $\rho(v_i) = 0$ and $pr(v_j) < pr(v_w)$ for $j = 2, 3, \ldots, k - 1$, do:
      - $pr(w) := pr(w) \cup \{i\}$

Proof of correctness is complicated.

Running-time $O(n(m + n))$.
Algorithm

- For all \( v \in V \) do:
  - \( pr(v) := \emptyset \)
  - \( \rho(v) := 0 \)

- For \( i := n \) down to 1 do:
  - Choose node \( v \) with \( pr(v) \) maximal and \( \rho(v) = 0 \)
  - \( \rho(v) := i \)
  - For all \( w \) with \( \rho(w) = 0 \) do
    - If there is a path \( v = v_1, v_2, \ldots, v_k = w \) with:
      - \( \rho(v_i) = 0 \) and \( pr(v_j) < pr(v_w) \)
      - for \( j = 2, 3, \ldots, k - 1 \), do:
        - \( pr(w) := pr(w) \cup \{i\} \)
  - Proof of correctness is complicated.
Algorithm

- For all $v \in V$ do:
  - $pr(v) := \emptyset$
  - $\rho(v) := 0$

- For $i := n$ down to 1 do:
  - Choose node $v$ with $pr(v)$ maximal and $\rho(v) = 0$
  - $\rho(v) := i$
  - For all $w$ with $\rho(w) = 0$ do
    - If there is a path $v = v_1, v_2, \ldots, v_k = w$ with:
      - $\rho(v_i) = 0$ and $pr(v_j) < pr(v_w)$
      - for $j = 2, 3, \ldots, k - 1$, do:
        - $pr(w) := pr(w) \cup \{i\}$

- Proof of correctness is complicated.

- Running-time $O(n(m + n))$
Statements

**Theorem**

Let $\rho$ be a MES for $G = (V, E)$. Then a clique-separator for $G$ is also a clique-separator for $G_\rho$.

- Let $V_1, \ldots, V_k$ be the node sets of the components from $G[V \setminus C]$. 
Statements

Theorem

Let $\rho$ be a MES for $G = (V, E)$. Then a clique-separator for $G$ is also a clique-separator for $G_\rho$.

- Let $V_1, \ldots, V_k$ be the node sets of the components from $G[V \setminus C]$.
- Delete from $F_\rho$ all edges, which connect two components.
Statements

Theorem

Let $\rho$ be a MES for $G = (V, E)$. Then a clique-separator for $G$ is also a clique-separator for $G_\rho$.

- Let $V_1, \ldots, V_k$ be the node sets of the components from $G[V \setminus C]$.
- Delete from $F_\rho$ all edges, which connect two components.
- Call this new edge set $F$, $F \subset F_\rho$. 
Theorem

Let \( \rho \) be a MES for \( G = (V, E) \). Then a clique-separator for \( G \) is also a clique-separator for \( G_\rho \).

- Let \( V_1, \ldots, V_k \) be the node sets of the components from \( G[V \setminus C] \).
- Delete from \( F_\rho \) all edges, which connect two components.
- Call this new edge set \( F, F \subset F_\rho \).
- Show: \( G' = (V, E \cup F) \) is chordal.
Statements

Theorem

Let $\rho$ be a MES for $G = (V, E)$. Then a clique-separator for $G$ is also a clique-separator for $G_\rho$.

- Let $V_1, \ldots, V_k$ be the node sets of the components from $G[V \setminus C]$.
- Delete from $F_\rho$ all edges, which connect two components.
- Call this new edge set $F$, $F \subseteq F_\rho$.
- Show: $G' = (V, E \cup F)$ is chordal.
  - Let $K$ be a cycle in $G'$ of length $\geq 4$. 
Statements

**Theorem**

Let \( \rho \) be a MES for \( G = (V, E) \). Then a clique-separator for \( G \) is also a clique-separator for \( G_\rho \).

- Let \( V_1, \ldots, V_k \) be the node sets of the components from \( G[V \setminus C] \).
- Delete from \( F_\rho \) all edges, which connect two components.
- Call this new edge set \( F, F \subset F_\rho \).
- Show: \( G' = (V, E \cup F) \) is chordal.
  - Let \( K \) be a cycle in \( G' \) of length \( \geq 4 \).
  - If \( K \subset G[V_i \cup C] \), then has \( K \) a chord in \( F_\rho \), because \( G_\rho \) is chordal.
Theorem

Let $\rho$ be a MES for $G = (V, E)$. Then a clique-separator for $G$ is also a clique-separator for $G_\rho$.

- Let $V_1, \ldots, V_k$ be the node sets of the components from $G[V \setminus C]$.
- Delete from $F_\rho$ all edges, which connect two components.
- Call this new edge set $F, F \subset F_\rho$.
- Show: $G' = (V, E \cup F)$ is chordal.
  - Let $K$ be a cycle in $G'$ of length $\geq 4$.
  - If $K \subset G[V_i \cup C]$, then has $K$ a chord in $F_\rho$, because $G_\rho$ is chordal.
  - This chord is in $E \cup F$. 
Let \( \rho \) be a MES for \( G = (V, E) \). Then a clique-separator for \( G \) is also a clique-separator for \( G_\rho \).

- Let \( V_1, \ldots, V_k \) be the node sets of the components from \( G[V \setminus C] \).
- Delete from \( F_\rho \) all edges, which connect two components.
- Call this new edge set \( F, F \subset F_\rho \).
- Show: \( G' = (V, E \cup F) \) is chordal.
  - Let \( K \) be a cycle in \( G' \) of length \( \geq 4 \).
  - If \( K \subset G[V_i \cup C] \), then has \( K \) a chord in \( F_\rho \), because \( G_\rho \) is chordal.
  - This chord is in \( E \cup F \).
  - If \( K \) goes through different \( V_i \), then has \( K \) two nodes in \( C \), which are not connected in \( C \).
**Theorem**

Let $\rho$ be a MES for $G = (V, E)$. Then a clique-separator for $G$ is also a clique-separator for $G_\rho$.

- Let $V_1, \ldots, V_k$ be the node sets of the components from $G[V \setminus C]$.
- Delete from $F_\rho$ all edges, which connect two components.
- Call this new edge set $F$, $F \subset F_\rho$.
- Show: $G' = (V, E \cup F)$ is chordal.
  - Let $K$ be a cycle in $G'$ of length $\geq 4$.
  - If $K \subset G[V_i \cup C]$, then has $K$ a chord in $F_\rho$, because $G_\rho$ is chordal.
  - This chord is in $E \cup F$.
  - If $K$ goes through different $V_i$, then has $K$ two nodes in $C$, which are not connected in $C$.
  - Thus $K$ has a chord in $G'$. 
**Theorem**

Let $\rho$ be a MES for $G = (V, E)$. Then a clique-separator for $G$ is also a clique-separator for $G_\rho$.

- Let $V_1, \ldots, V_k$ be the node sets of the components from $G[V \setminus C]$.
- Delete from $F_\rho$ all edges, which connects two components.
- Call this new edge set $F$, $F \subseteq F_\rho$.
- Shown on the last slide: $G' = (V, E \cup F)$ is chordal
Theorem

Let $\rho$ be a MES for $G = (V, E)$. Then a clique-separator for $G$ is also a clique-separator for $G_\rho$.

- Let $V_1, \ldots, V_k$ be the node sets of the components from $G[V \setminus C]$.
- Delete from $F_\rho$ all edges, which connects two components.
- Call this new edge set $F$, $F \subset F_\rho$.
- Shown on the last slide: $G' = (V, E \cup F)$ is chordal
- Thus $G'$ is chordal and has PES $\rho'$ with $F_{\rho'} = F$. 
Theorem

Let \( \rho \) be a MES for \( G = (V, E) \). Then a clique-separator for \( G \) is also a clique-separator for \( G_\rho \).

- Let \( V_1, \ldots, V_k \) be the node sets of the components from \( G[V \setminus C] \).
- Delete from \( F_\rho \) all edges, which connects two components.
- Call this new edge set \( F, F \subseteq F_\rho \).
- Shown on the last slide: \( G' = (V, E \cup F) \) is chordal
- Thus \( G' \) is chordal and has PES \( \rho' \) with \( F_{\rho'} = F \).
- \( \rho \) is a MES, thus: \( F_{\rho'} = F_\rho = F \).
Theorem

Let \( \rho \) be a MES for \( G = (V, E) \). Then a clique-separator for \( G \) is also a clique-separator for \( G_\rho \).

- Let \( V_1, \ldots, V_k \) be the node sets of the components from \( G[V \setminus C] \).
- Delete from \( F_\rho \) all edges, which connects two components.
- Call this new edge set \( F, F \subset F_\rho \).
- Shown on the last slide: \( G' = (V, E \cup F) \) is chordal
- Thus \( G' \) is chordal and has PES \( \rho' \) with \( F_{\rho'} = F \).
- \( \rho \) is a MES, thus: \( F_{\rho'} = F_\rho = F \).
- This ends the proof.
Clique-Separator-Tree Algorithm

\[ \rho := \text{LexBFS}(G) \]
Clique-Separator-Tree Algorithm

- $\rho := \text{LexBFS}(G)$
- $F_\rho := \text{Fill}_\text{In}(G, \rho)$
Clique-Separator-Tree Algorithm

- $\rho := \text{LexBFS}(G)$
- $F_\rho := \text{Fill\_ln}(G, \rho)$
- For all $v \in V$ do:
Clique-Separator-Tree Algorithm

- $\rho := \text{LexBFS}(G)$
- $F_\rho := \text{Fill\_In}(G, \rho)$
- For all $v \in V$ do:
  - $C(v) := \emptyset$
Clique-Separator-Tree Algorithm

- $\rho := \text{LexBFS}(G)$
- $F_\rho := \text{Fill\_In}(G, \rho)$

For all $v \in V$ do:

- $C(v) := \emptyset$
- For all $w \in V$ do:
Clique-Separator-Tree Algorithm

- $\rho := \text{LexBFS}(G)$
- $F_\rho := \text{Fill\_In}(G, \rho)$
- For all $v \in V$ do:
  - $C(v) := \emptyset$
  - For all $w \in V$ do:
    - If $\rho(w) > \rho(v)$ and $\{v, w\} \in E \cup F_\rho$ holds, then do:
Clique-Separator-Tree Algorithm

- $\rho := \text{LexBFS}(G)$
- $F_\rho := \text{Fill}_\text{In}(G, \rho)$
- For all $v \in V$ do:
  - $C(v) := \emptyset$
  - For all $w \in V$ do:
    - If $\rho(w) > \rho(v)$ and $\{v, w\} \in E \cup F_\rho$ holds, then do:
      - $C(v) := C(v) \cup \{w\}$
Clique-Separator-Tree Algorithm

- $\rho := \text{LexBFS}(G)$
- $F_\rho := \text{Fill}_\rho(G, \rho)$
- For all $v \in V$ do:
  - $C(v) := \emptyset$
  - For all $w \in V$ do:
    - If $\rho(w) > \rho(v)$ and $\{v, w\} \in E \cup F_\rho$ holds, then do:
      - $C(v) := C(v) \cup \{w\}$
- $k := 1$
Clique-Separator-Tree Algorithm

\begin{itemize}
\item \( \rho := \text{LexBFS}(G) \)
\item \( F_\rho := \text{Fill}_\text{ln}(G, \rho) \)
\item For all \( v \in V \) do:
  \begin{itemize}
  \item \( C(v) := \emptyset \)
  \item For all \( w \in V \) do:
    \begin{itemize}
    \item If \( \rho(w) > \rho(v) \) and \( \{v, w\} \in E \cup F_\rho \) holds, then do:
    \begin{itemize}
    \item \( C(v) := C(v) \cup \{w\} \)
    \end{itemize}
    \end{itemize}
  \end{itemize}
\item \( k := 1 \)
\item For all \( i := 1 \) bis \( n - 1 \) do:
\end{itemize}
Clique-Separator-Tree Algorithm

- \( \rho := \text{LexBFS}(G) \)
- \( F_\rho := \text{Fill}_\text{In}(G, \rho) \)
- For all \( v \in V \) do:
  - \( C(v) := \emptyset \)
  - For all \( w \in V \) do:
    - If \( \rho(w) > \rho(v) \) and \( \{v, w\} \in E \cup F_\rho \) holds, then do:
      - \( C(v) := C(v) \cup \{w\} \)
- \( k := 1 \)
- For all \( i := 1 \) bis \( n - 1 \) do:
  - \( v := \rho^{-1}(i) \)
Clique-Separator-Tree Algorithm

\[ \rho := \text{LexBFS}(G) \]
\[ F_\rho := \text{Fill}_\text{In}(G, \rho) \]

For all \( v \in V \) do:
- \( C(v) := \emptyset \)
- For all \( w \in V \) do:
  - If \( \rho(w) > \rho(v) \) and \( \{v, w\} \in E \cup F_\rho \) holds, then do:
    - \( C(v) := C(v) \cup \{w\} \)

\[ k := 1 \]

For all \( i := 1 \) bis \( n - 1 \) do:
- \( v := \rho^{-1}(i) \)
- Let \( A \) be a component in \( G[V \setminus C(v)] \) which contains \( v \).
Clique-Separator-Tree Algorithm

- $\rho := \text{LexBFS}(G)$
- $F_\rho := \text{Fill}_\text{In}(G, \rho)$
- For all $v \in V$ do:
  - $C(v) := \emptyset$
  - For all $w \in V$ do:
    - If $\rho(w) > \rho(v)$ and $\{v, w\} \in E \cup F_\rho$ holds, then do:
      - $C(v) := C(v) \cup \{w\}$
- $k := 1$
- For all $i := 1$ bis $n - 1$ do:
  - $v := \rho^{-1}(i)$
  - Let $A$ be a component in $G[V \setminus C(v)]$ which contains $v$.
  - Let $B = V \setminus (A \cup C(v))$
Clique-Separator-Tree Algorithm

- \( \rho := \text{LexBFS}(G) \)
- \( F_\rho := \text{Fill}_\text{In}(G, \rho) \)
- For all \( v \in V \) do:
  - \( C(v) := \emptyset \)
  - For all \( w \in V \) do:
    - If \( \rho(w) > \rho(v) \) and \( \{v, w\} \in E \cup F_\rho \) holds, then do:
      - \( C(v) := C(v) \cup \{w\} \)
- \( k := 1 \)
- For all \( i := 1 \) bis \( n - 1 \) do:
  - \( v := \rho^{-1}(i) \)
  - Let \( A \) be a component in \( G[V \setminus C(v)] \) which contains \( v \).
  - Let \( B = V \setminus (A \cup C(v)) \)
  - If \( B \neq \emptyset \) and \( C(v) \) is a clique:
Clique-Separator-Tree Algorithm

- $\rho := \text{LexBFS}(G)$
- $F_\rho := \text{Fill\_In}(G, \rho)$
- For all $v \in V$ do:
  - $C(v) := \emptyset$
  - For all $w \in V$ do:
    - If $\rho(w) > \rho(v)$ and $\{v, w\} \in E \cup F_\rho$ holds, then do:
      - $C(v) := C(v) \cup \{w\}$
- $k := 1$
- For all $i := 1$ bis $n - 1$ do:
  - $v := \rho^{-1}(i)$
  - Let $A$ be a component in $G[V \setminus C(v)]$ which contains $v$.
  - Let $B = V \setminus (A \cup C(v))$
  - If $B \neq \emptyset$ and $C(v)$ is a clique:
    - $\text{Atoms}(k) := A$
Clique-Separator-Tree Algorithm

- $\rho := \text{LexBFS}(G)$
- $F_\rho := \text{Fill}_\text{In}(G, \rho)$

For all $v \in V$ do:
  - $C(v) := \emptyset$
  - For all $w \in V$ do:
    - If $\rho(w) > \rho(v)$ and $\{v, w\} \in E \cup F_\rho$ holds, then do:
      - $C(v) := C(v) \cup \{w\}$

- $k := 1$

For all $i := 1$ bis $n - 1$ do:
  - $v := \rho^{-1}(i)$
  - Let $A$ be a component in $G[V \setminus C(v)]$ which contains $v$.
  - Let $B = V \setminus (A \cup C(v))$
  - If $B \neq \emptyset$ and $C(v)$ is a clique:
    - $\text{Atoms}(k) := A$
    - $k := k + 1$
Clique-Separator-Tree Algorithm

1. \( \rho := \text{LexBFS}(G) \)
2. \( F_\rho := \text{Fill}_\rho \text{In}(G, \rho) \)
3. For all \( v \in V \) do:
   - \( C(v) := \emptyset \)
   - For all \( w \in V \) do:
     - If \( \rho(w) > \rho(v) \) and \( \{v, w\} \in E \cup F_\rho \) holds, then do:
       - \( C(v) := C(v) \cup \{w\} \)
4. \( k := 1 \)
5. For all \( i := 1 \) bis \( n - 1 \) do:
   - \( v := \rho^{-1}(i) \)
   - Let \( A \) be a component in \( G[V \setminus C(v)] \) which contains \( v \).
   - Let \( B = V \setminus (A \cup C(v)) \)
   - If \( B \neq \emptyset \) and \( C(v) \) is a clique:
     - \( \text{Atoms}(k) := A \)
     - \( k := k + 1 \)
     - \( G := G[B \cup C(v)] \)
Clique-Separator-Tree Algorithm

\[ \rho := \text{LexBFS}(G) \]
\[ F_\rho := \text{Fill\_In}(G, \rho) \]

For all \( v \in V \) do:
\[ C(v) := \emptyset \]

For all \( w \in V \) do:
\[ \text{If } \rho(w) > \rho(v) \text{ and } \{v, w\} \in E \cup F_\rho \text{ holds, then do:} \]
\[ C(v) := C(v) \cup \{w\} \]

\[ k := 1 \]

For all \( i := 1 \) bis \( n - 1 \) do:
\[ v := \rho^{-1}(i) \]

Let \( A \) be a component in \( G[V \setminus C(v)] \) which contains \( v \).

Let \( B = V \setminus (A \cup C(v)) \)

If \( B \neq \emptyset \) and \( C(v) \) is a clique:
\[ \text{Atoms}(k) := A \]
\[ k := k + 1 \]
\[ G := G[B \cup C(v)] \]
\[ \text{Atoms}(k) := V(G) \]
Correctness

**Theorem**

*If $G$ has a clique-separator. Then is this separator $C(v)$ for some node $v$.***

- Let $\rho$ a MES as computed by the above slides.
Correctness

Theorem

If $G$ has a clique-separator. Then is this separator $C(v)$ for some node $v$.

- Let $\rho$ a MES as computed by the above slides.
- Let $C$ be a inclusion minimal clique-separator.
Correctness

**Theorem**

*If $G$ has a clique-separator. Then is this separator $C(v)$ for some node $v$.***

- Let $\rho$ a MES as computed by the above slides.
- Let $C$ be a inclusion minimal clique-separator.
- Let $A, B$ be two components from $G[V \setminus C]$. 
Correctness

Theorem

*If G has a clique-separator. Then is this separator C(v) for some node v.*

- Let ρ a MES as computed by the above slides.
- Let C be a inclusion minimal cliqueseparator.
- Let A, B be two components from G[V \ C].
- Thus each node from C has a neighbour in A and B.
Correctness

Theorem

If $G$ has a clique-separator. Then is this separator $C(v)$ for some node $v$.

- Let $\rho$ a MES as computed by the above slides.
- Let $C$ be a inclusion minimal clique-separator.
- Let $A, B$ be two components from $G[V \setminus C]$.
- Thus each node from $C$ has a neighbour in $A$ and $B$.
- Let $x, y$ be nodes with the largest $\rho$ values in $A$ and $B$. 

By contradiction on the next slide.
Correctness

**Theorem**

*If G has a clique-separator. Then is this separator C(v) for some node v.*

- Let \( \rho \) a MES as computed by the above slides.
- Let \( C \) be a inclusion minimal clique-separator.
- Let \( A, B \) be two components from \( G[V \setminus C] \).
- Thus each node from \( C \) has a neighbour in \( A \) and \( B \).
- Let \( x, y \) be nodes with the largest \( \rho \) values in \( A \) and \( B \).
- Show now: there is no node \( z \in C \) with: \( \rho(z) \leq \min\{\rho(x), \rho(y)\} \).
Correctness

**Theorem**

*If G has a clique-separator. Then is this separator C(v) for some node v.*

- Let \( \rho \) a MES as computed by the above slides.
- Let \( C \) be a inclusion minimal clique-separator.
- Let \( A, B \) be two components from \( G[V \setminus C] \).
- Thus each node from \( C \) has a neighbour in \( A \) and \( B \).
- Let \( x, y \) be nodes with the largest \( \rho \) values in \( A \) and \( B \).
- Show now: there is no node \( z \in C \) with: \( \rho(z) \leq \min\{\rho(x), \rho(y)\} \).
  - By contradiction
Correctness

**Theorem**

*If $G$ has a clique-separator. Then is this separator $C(v)$ for some node $v$."

- Let $\rho$ a MES as computed by the above slides.
- Let $C$ be a inclusion minimal clique-separator.
- Let $A, B$ be two components from $G[V \setminus C]$.
- Thus each node from $C$ has a neighbour in $A$ and $B$.
- Let $x, y$ be nodes with the largest $\rho$ values in $A$ and $B$.
- Show now: there is no node $z \in C$ with: $\rho(z) \leq \min\{\rho(x), \rho(y)\}$.
  - By contradiction
  - on the next slide.
Correctness (intermediate step)

If $G$ has a clique-separator, then is it $C(v)$ for some node $v$.

Assume: There is a node $z \in C$ with: $\rho(z) \leq \min\{\rho(x), \rho(y)\}$.

1. Let $x = x_1, x_2, \ldots, x_{j-1}, x_j = z$ be the shortest path in $G_\rho$ with $x_1x_2 \ldots x_{j-1} \in A$. 

![Diagram showing nodes and edges in a graph](image)
Correctness (intermediate step)

If $G$ has a clique-separator, then is it $C(v)$ for some node $v$.

Assume: There is a node $z \in C$ with: $\rho(z) \leq \min\{\rho(x), \rho(y)\}$.

1. Let $x = x_1, x_2, \ldots, x_{j-1}, x_j = z$ be the shortest path in $G_\rho$ with $x_1x_2 \ldots x_{j-1} \in A$.

2. If there is an $i$ with $i \leq j - 1$ and $\rho(x_i) \leq \rho(x_{j-1})$, then choose such $i$ maximal.
Correctness (intermediate step)

If \( G \) has a clique-separator, then is it \( C(v) \) for some node \( v \).

Assume: There is a node \( z \in C \) with: \( \rho(z) \leq \min\{\rho(x), \rho(y)\} \).

- Let \( x = x_1, x_2, \ldots, x_{j-1}, x_j = z \) be the shortest path in \( G_\rho \) with \( x_1x_2\ldots x_{j-1} \in A \).
- If there is an \( i \) with \( i \leq j - 1 \) and \( \rho(x_i) \leq \rho(x_{j-1}) \), then choose such \( i \) maximal.
- Thus we have \( i \geq 2 \) (Note: \( \rho(z) \leq \min\{\rho(x), \rho(y)\} \))
Correctness (intermediate step)

If $G$ has a clique-separator, then is it $C(v)$ for some node $v$.

Assume: There is a node $z \in C$ with: $\rho(z) \leq \min\{\rho(x), \rho(y)\}$.

- Let $x = x_1, x_2, \ldots, x_{j-1}, x_j = z$ be the shortest path in $G_\rho$ with $x_1x_2 \ldots x_{j-1} \in A$.
- If there is an $i$ with $i \leq j - 1$ and $\rho(x_i) \leq \rho(x_{j-1})$, then choose such $i$ maximal.
- Thus we have $i \geq 2$ (Note: $\rho(z) \leq \min\{\rho(x), \rho(y)\}$)
- And $\{x_{i-1}, x_{i+1}\} \in F_\rho$ holds, because of $\rho(x_i) \leq \min\{\rho(x_{i-1}), \rho(x_{i+1})\}$ and the definition of Fill-In
Correctness (intermediate step)

If $G$ has a clique-separator, then is it $C(v)$ for some node $v$.

Assume: There is a node $z \in C$ with: $\rho(z) \leq \min\{\rho(x), \rho(y)\}$.

- Let $x = x_1, x_2, \ldots, x_{j-1}, x_j = z$ be the shortest path in $G_\rho$ with $x_1x_2 \ldots x_{j-1} \in A$.
- If there is an $i$ with $i \leq j - 1$ and $\rho(x_i) \leq \rho(x_{j-1})$, then choose such $i$ maximal.
- Thus we have $i \geq 2$ (Note: $\rho(z) \leq \min\{\rho(x), \rho(y)\}$)
- And $\{x_{i-1}, x_{i+1}\} \in F_\rho$ holds, because of $\rho(x_i) \leq \min\{\rho(x_{i-1}), \rho(x_{i+1})\}$ and the definition of Fill-In
- This is a contradiction to the minimality of the path.
Correctness (intermediate step)

If $G$ has a clique-separator, then it is $C(v)$ for some node $v$.

Assume: There is a node $z \in C$ with: $\rho(z) \leq \min\{\rho(x), \rho(y)\}$.

Thus there is a path $x = x_1x_2 \ldots x_{j-1}x_j = z$ in $G_\rho$ with $\rho(x_i) > \rho(x_{i+1})$ for $i = 1, 2, \ldots, j - 1$. 
Correctness (intermediate step)

If $G$ has a clique-separator, then it is $C(v)$ for some node $v$.

Assume: There is a node $z \in C$ with: $\rho(z) \leq \min\{\rho(x), \rho(y)\}$.

- Thus there is a path $x = x_1x_2 \ldots x_{j-1}x_j = z$ in $G_\rho$ with $\rho(x_i) > \rho(x_{i+1})$ for $i = 1, 2, \ldots, j - 1$.

- Thus there a path $y = y_1y_2 \ldots y_{l-1}x_l = z$ in $G_\rho$ with $\rho(y_i) > \rho(y_{i+1})$ for $i = 1, 2, \ldots, l - 1$. 
Correctness (intermediate step)

If \( G \) has a clique-separator, then is it \( C(v) \) for some node \( v \).

Assume: There is a node \( z \in C \) with: \( \rho(z) \leq \min\{\rho(x), \rho(y)\} \).

- Thus there is a path \( x = x_1x_2\ldots x_{j-1}x_j = z \) in \( G_\rho \) with \( \rho(x_i) > \rho(x_{i+1}) \) for \( i = 1, 2, \ldots, j-1 \).

- Thus there a path \( y = y_1y_2\ldots y_{l-1}x_l = z \) in \( G_\rho \) with \( \rho(y_i) > \rho(y_{i+1}) \) for \( i = 1, 2, \ldots, l-1 \).

- Thus \( \{x, y\} \in F_\rho \) holds, which is a contradiction.
Correctness (Continuation)

If $G$ has a clique-separator, then is it $C(v)$ for some node $v$.

- W.l.o.g. let now be $\rho(x) < \rho(y)$. 
Correctness (Continuation)

If $G$ has a clique-separator, then is it $C(v)$ for some node $v$.

- W.l.o.g. let now be $\rho(x) < \rho(y)$.
- Then holds: $\max\{\rho(v) \mid v \in A\} = \rho(x) < \rho(z)$ for all $z \in C$. 
Correctness (Continuation)

If $G$ has a clique-separator, then is it $C(v)$ for some node $v$.

1. W.l.o.g. let now be $\rho(x) < \rho(y)$.
2. Then holds: $\max\{\rho(v) \mid v \in A\} = \rho(x) < \rho(z)$ for all $z \in C$.
3. Show now $C(x) = C$.
Correctness (Continuation)

If $G$ has a clique-separator, then is it $C(v)$ for some node $v$.

- W.l.o.g. let now be $\rho(x) < \rho(y)$.
- Then holds: $\max\{\rho(v) \mid v \in A\} = \rho(x) < \rho(z)$ for all $z \in C$.
- Show now $C(x) = C$
- I.e. show: $\forall z \in C : \{x, z\} \in E \cup F_{\rho}$. 
Correctness (Continuation)

If $G$ has a clique-separator, then is it $C(v)$ for some node $v$.

- W.l.o.g. let now be $\rho(x) < \rho(y)$.
- Then holds: $\max\{\rho(v) \mid v \in A\} = \rho(x) < \rho(z)$ for all $z \in C$.
- Show now $C(x) = C$
- I.e. show: $\forall z \in C : \{x, z\} \in E \cup F_\rho$.
- Let $x = x_1x_2 \ldots x_{j-1}x_j = z$ be the shortest path in $G_\rho$ with $x_1, x_2, \ldots, x_{j-1} \in A$. 
Correctness (Continuation)

If $G$ has a clique-separator, then is it $C(v)$ for some node $v$.

- W.l.o.g. let now be $\rho(x) < \rho(y)$.
- Then holds: $\max\{\rho(v) \mid v \in A\} = \rho(x) < \rho(z)$ for all $z \in C$.
- Show now $C(x) = C$
- i.e. show: $\forall z \in C : \{x, z\} \in E \cup F_\rho$.
- Let $x = x_1 x_2 \ldots x_{j-1} x_j = z$ be the shortest path in $G_\rho$ with $x_1, x_2, \ldots, x_{j-1} \in A$.
- If $j \geq 3$ holds, then we have $\rho(x_i) > \rho(x_{i+1})$ for $i = 1, 2, \ldots, j - 1$. 
Correctness (Continuation)

If $G$ has a clique-separator, then it is $C(v)$ for some node $v$.

- W.l.o.g. let now be $\rho(x) < \rho(y)$.
- Then holds: $\max\{\rho(v) \mid v \in A\} = \rho(x) < \rho(z)$ for all $z \in C$.
- Show now $C(x) = C$
- I.e. show: $\forall z \in C : \{x, z\} \in E \cup F_\rho$.
- Let $x = x_1x_2 \ldots x_{j-1}x_j = z$ be the shortest path in $G_\rho$ with $x_1, x_2, \ldots, x_{j-1} \in A$.
- If $j \geq 3$ holds, then we have $\rho(x_i) > \rho(x_{i+1})$ for $i = 1, 2, \ldots, j - 1$.
- This is contradiction to $\rho(z) > \rho(x)$. 
Correctness (Continuation)

If $G$ has a clique-separator, then is it $C(v)$ for some node $v$.

- W.l.o.g. let now be $\rho(x) < \rho(y)$.
- Then holds: $\max\{\rho(v) \mid v \in A\} = \rho(x) < \rho(z)$ for all $z \in C$.
- Show now $C(x) = C$
- I.e. show: $\forall z \in C : \{x, z\} \in E \cup F_\rho$.
- Let $x = x_1x_2 \ldots x_{j-1}x_j = z$ be the shortest path in $G_\rho$ with $x_1, x_2, \ldots, x_{j-1} \in A$.
- If $j \geq 3$ holds, then we have $\rho(x_i) > \rho(x_{i+1})$ for $i = 1, 2, \ldots, j - 1$.
- This is contradiction to $\rho(z) > \rho(x)$.
- Thus $j = 2$ and $\{x, z\} \in E \cup F_\rho$. 
The above algorithm has running-time $O(n(n + m))$ for computing the clique-separator-tree.
The above algorithm has running-time $O(n(n + m))$ for computing the clique-separator-tree.

By using the clique-separator-tree are the following problems are reduced to the atoms:

- Clique-Problem
**Theorem**

The above algorithm has running-time $O(n(n + m))$ for computing the clique-separator-tree.

**Theorem**

By using the clique-separator-tree are the following problems are reduced to the atoms:

- Clique-Problem
- Independent-Set Problem
**Theorem**

The above algorithm has running-time $O(n(n + m))$ for computing the clique-separator-tree.

**Theorem**

By using the clique-separator-tree are the following problems are reduced to the atoms:

- **Clique-Problem**
- **Independent-Set Problem**
- **Colouring-Problem**
A graph $G = (V, E)$ is of type $T_1$, iff:

- $V$ could be partitioned in $V_1, V_2$. 
A graph $G = (V, E)$ is of type $T_1$, iff:

- $V$ could be partitioned in $V_1, V_2$.
- $G[V_1]$ is a bipartite graph.
Clique-Separable

**Definition**

A graph $G = (V, E)$ is of type $T_1$, iff:

- $V$ could be partitioned in $V_1, V_2$.
- $G[V_1]$ is a bipartite graph.
- $G[V_2]$ is a clique.
Definition

A graph \( G = (V, E) \) is of type \( T_1 \), iff:

- \( V \) could be partitioned in \( V_1, V_2 \).
- \( G[V_1] \) is a bipartite graph.
- \( G[V_2] \) is a clique.
- Between \( V_1 \) and \( V_2 \) exist all possible edges.
Clique-Separable

Definition

A graph \( G = (V, E) \) is of type \( T_1 \), iff:

- \( V \) could be partitioned in \( V_1, V_2 \).
- \( G[V_1] \) is a bipartite graph.
- \( G[V_2] \) is a clique.
- Between \( V_1 \) and \( V_2 \) exist all possible edges.
Clique-Separable

Definition

A graph \( G = (V, E) \) is of type \( T_1 \), iff:

- \( V \) could be partitioned in \( V_1, V_2 \).
- \( G[V_1] \) is a bipartite graph.
- \( G[V_2] \) is a clique.
- Between \( V_1 \) and \( V_2 \) exist all possible edges.

Definition

A graph \( G = (V, E) \) is of type \( T_2 \), iff it is complete \( k \)-partite.
**Clique-Separable**

**Definition**

A graph $G = (V, E)$ is clique-separable, iff all Atoms are of Type $T_1$ or $T_2$. 

**Theorem**

Clique-separable graphs could be recognized in time $O(n^4)$.

The Clique-Problem, Independent-Set Problem and Colouring-Problem are solvable in polynomial time on clique-separable graphs.
Clique-Separable

**Definition**
A graph $G = (V, E)$ is clique-separable, iff all Atoms are of Type $T_1$ or $T_2$.

**Theorem**
Clique-separable graphs could be recognized in time $O(n^4)$. The Clique-Problem, Independent-Set Problem and Colouring-Problem are solvable in polynomial time on clique-separable graphs.
Questions

- What is a perfect graph?
Questions

- What is a perfect graph?
- Which graph classes are perfect?
Questions

- What is a perfect graph?
- Which graph classes are perfect?
- How hard is the recognition of perfect graphs?
Questions

- What is a perfect graph?
- Which graph classes are perfect?
- How hard is the recognition of perfect graphs?
- How hard is the colouring on perfect graphs?
Questions

- What is a perfect graph?
- Which graph classes are perfect?
- How hard is the recognition of perfect graphs?
- How hard is the colouring on perfect graphs?
- What is a minimal imperfect graph?
Questions

- What is a perfect graph?
- Which graph classes are perfect?
- How hard is the recognition of perfect graphs?
- How hard is the colouring on perfect graphs?
- What is a minimal imperfect graph?
- Which graphs are minimal imperfect?
Questions

- What is a perfect graph?
- Which graph classes are perfect?
- How hard is the recognition of perfect graphs?
- How hard is the colouring on perfect graphs?
- What is a minimal imperfect graph?
- Which graphs are minimal imperfect?
- What is a chordal graph?
Questions

- What is a perfect graph?
- Which graph classes are perfect?
- How hard is the recognition of perfect graphs?
- How hard is the colouring on perfect graphs?
- What is a minimal imperfect graph?
- Which graphs are minimal imperfect?
- What is a chordal graph?
- What is known about chordal graph?
Questions

- What is a perfect graph?
- Which graph classes are perfect?
- How hard is the recognition of perfect graphs?
- How hard is the colouring on perfect graphs?
- What is a minimal imperfect graph?
- Which graphs are minimal imperfect?
- What is a chordal graph?
- What is known about chordal graph?
- Why are chordal graphs not perfect?
Questions

- How hard is the recognition of chordal graphs?
Questions

1. How hard is the recognition of chordal graphs?
2. What is a PES?
Questions

- How hard is the recognition of chordal graphs?
- What is a PES?
- Which problems are easy on chordal graphs?
Questions

- How hard is the recognition of chordal graphs?
- What is a PES?
- Which problems are easy on chordal graphs?
- Give an alternative representation for chordal graphs?
Questions

- How hard is the recognition of chordal graphs?
- What is a PES?
- Which problems are easy on chordal graphs?
- Give an alternative representation for chordal graphs?
- What are comparability graphs?
Questions

- How hard is the recognition of chordal graphs?
- What is a PES?
- Which problems are easy on chordal graphs?
- Give an alternative representation for chordal graphs?
- What are comparability graphs?
- What is known about comparability graphs and interval graphs?
Questions

- How hard is the recognition of chordal graphs?
- What is a PES?
- Which problems are easy on chordal graphs?
- Give an alternative representation for chordal graphs?
- What are comparability graphs?
- What is known about comparability graphs and interval graphs?
- What is the idea of the proof to show that perfect graphs are closes under complement?