Contents I

1. Bandwidth
   - Motivation
   - Definition
   - Problems
   - Complete Trees

2. Pathwidth
   - Idea
   - Definition
   - Theorems

3. Treewidth
   - Definition

4. k-Trees
   - Example
   - Definition
   - Theorems

5. Applications
   - Cactus and near-trees
   - Halin-graphs and outer-planar graphs
   - SP-Graphs
   - Minors
Motivation

- Till now: Problems are efficient solvable, if the “flow of information is not too large”.
Motivation

- Till now: Problems are efficient solvable, if the “flow of information is not too large”.
- Example: interval-graphs, permutation-graphs, trees, ...
Motivation

- Till now: Problems are efficient solvable, if the “flow of information is not too large”.
- Example: interval-graphs, permutation-graphs, trees, ...
- Idea: Try to generalize the restricted flow of information of the trees.
Motivation

- Till now: Problems are efficient solvable, if the “flow of information is not too large”.
- Example: interval-graphs, permutation-graphs, trees, ...
- Idea: Try to generalize the restricted flow of information of the trees.
- Define a generalized tree.
Motivation

- Till now: Problems are efficient solvable, if the “flow of information is not too large”.
- Example: interval-graphs, permutation-graphs, trees, ...
- Idea: Try to generalize the restricted flow of information of the trees.
- Define a generalized tree.
- Idea for this: make the nodes “fat”.
Motivation

- Till now: Problems are efficient solvable, if the “flow of information is not too large”.
- Example: interval-graphs, permutation-graphs, trees, ...
- Idea: Try to generalize the restricted flow of information of the trees.
- Define a generalized tree.
- Idea for this: make the nodes “fat”.
- We start the bandwidth problem.
Motivation

- Till now: Problems are efficient solvable, if the “flow of information is not too large”.
- Example: interval-graphs, permutation-graphs, trees, ...
- Idea: Try to generalize the restricted flow of information of the trees.
- Define a generalized tree.
- Idea for this: make the nodes “fat”.
- We start the bandwidth problem.
- After that: pathwidth, treewidth and partial $k$-trees.
Definition (Bandwidth)

Let \( G = (V, E) \) be a graph and let \( v, v' \in V \).

- A labeling of \( G \) is a function \( e : V \to \mathbb{N} \) with \( e(v) = e(v') \Rightarrow v = v' \).
Definition of Bandwidth

**Definition (Bandwidth)**

Let $G = (V, E)$ be a graph and let $v, v' \in V$.

- A labeling of $G$ is a function $e : V \rightarrow \mathbb{N}$ with $e(v) = e(v') \Rightarrow v = v'$.
- The distance between $v$ and $v'$ in the labeling $e$ is given by: $\text{dist}(e, v, v') = |e(v) - e(v')|$. 
Definition of Bandwidth

Definition (Bandwidth)

Let $G = (V, E)$ be a graph and let $v, v' \in V$.

- A **labeling** of $G$ is a function $e : V \rightarrow \mathbb{N}$ with $e(v) = e(v') \Rightarrow v = v'$.
- The distance between $v$ and $v'$ in the labeling $e$ is given by: $\text{dist}(e, v, v') = |e(v) - e(v')|$. 
- The **bandwidth** of the labeling $e$ on $G$ is $\text{bw}(e, G) = \max\{\text{dist}(e, v, v') \mid \{v, v'\} \in E\}$. 

Definition of Bandwidth

**Definition** (Bandwidth)

Let \( G = (V, E) \) be a graph and let \( v, v' \in V \).

- A **labeling** of \( G \) is a function 
  \[ e : V \rightarrow \mathbb{N} \] 
  with 
  \[ e(v) = e(v') \Rightarrow v = v' \].

- The distance between \( v \) and \( v' \) in the labeling \( e \) is given by:
  \[ \text{dist}(e, v, v') = |e(v) - e(v')| \].

- The bandwidth of the labeling \( e \) on \( G \) is
  \[ \text{bw}(e, G) = \max\{\text{dist}(e, v, v') \mid \{v, v'\} \in E\} \].

- The **bandwidth** of graph \( G \) is
  \[ \text{bw}(G) = \min_{e : V \rightarrow \mathbb{N}} \{\text{bw}(e, G)\} \].
Example

\[\Sigma = 0\]

\[
\begin{pmatrix}
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 \\
\end{pmatrix}
\]

\[\text{bandwidth} = 2\]
Example

e(v1) = 1

\text{bandwidth} = 5

\begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 
\end{bmatrix}
Example

e(v1) = 1

\[ e(v2) = 2 \]
Example

\[e(v_2) = 2\]
\[e(v_1) = 1\]
\[e(v_3) = 3\]
Example

\begin{align*}
  e(v_1) &= 1 \\
  e(v_2) &= 2 \\
  e(v_3) &= 3 \\
  e(v_4) &= 4 \\
  e(v_5) &= 6 \\
  e(v_6) &= 5 \\
  \Sigma &= 0
\end{align*}
Example

\[
e(v_2) = 2 \\
e(v_1) = 1 \\
e(v_3) = 3 \\
e(v_4) = 4 \\
e(v_5) = 5
\]
Example

Let's consider a graph with vertices $v_1, v_2, v_3, v_4, v_5, v_6$ and edges $e(v_1), e(v_2), e(v_3), e(v_4), e(v_5), e(v_6)$. The bandwidth of the graph can be calculated as follows:

- $e(v_2) = 2$
- $e(v_1) = 1$
- $e(v_3) = 3$
- $e(v_6) = 6$
- $e(v_4) = 4$
- $e(v_5) = 5$

The bandwidth of the graph is 5, as shown in the adjacency matrix below:

$$
\begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
$$

The bandwidth is equal to the maximum value of $e(v_i)$ over all vertices in the graph.
Example

\[
\begin{align*}
e(v_2) &= 2 \\
e(v_1) &= 1 \\
e(v_3) &= 3 \\
e(v_6) &= 6 \\
e(v_4) &= 4 \\
e(v_5) &= 5
\end{align*}
\]

\[
\begin{pmatrix}
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 
\end{pmatrix}
\]

\[
\begin{pmatrix}
0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 
\end{pmatrix}
\]

\[\text{bandwidth} = 5\]
Example

\[
\begin{align*}
e(v_2) &= 2 \\
e(v_1) &= 1 \\
e(v_6) &= 6 \\
e(v_3) &= 3 \\
e(v_4) &= 4 \\
e(v_5) &= 5
\end{align*}
\]

bandwidth = 5
Example

\begin{itemize}
\item \(e(v_2) = 2\)
\item \(e(v_1) = 1\)
\item \(e(v_6) = 6\)
\item \(e(v_5) = 5\)
\item \(e(v_3) = 3\)
\item \(e(v_4) = 4\)
\end{itemize}

\text{bandwidth} = 5
Example

\[ e(v_1) = 1 \]
\[ e(v_2) = 2 \]
\[ e(v_3) = 3 \]
\[ e(v_4) = 4 \]
\[ e(v_5) = 5 \]
\[ e(v_6) = 6 \]

\[ \Sigma = 0 \]

bandwidth = 5
Example

\begin{align*}
\text{bandwidth} &= 5 \\
&= 5
\end{align*}
Example

\[ e(v_1) = 1 \]
\[ e(v_2) = 2 \]
\[ e(v_3) = 3 \]
\[ e(v_4) = 4 \]
\[ e(v_5) = 5 \]
\[ e(v_6) = 6 \]

\text{bandwidth} = 5
Example

$$\begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 
\end{bmatrix}$$

bandwidth = 5
Example

\[
e(v_1) = 1, e(v_2) = 2, e(v_3) = 3, e(v_4) = 4, e(v_5) = 5, e(v_6) = 6
\]

\[
\begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 
\end{bmatrix}
\]

\[
\text{bandwidth} = 5
\]
**Example**

Consider the graph below:

- $e(v_2) = 2$
- $e(v_1) = 1$
- $e(v_6) = 6$
- $e(v_3) = 3$
- $e(v_4) = 4$
- $e(v_5) = 5$

The bandwidth is 5.

Next, consider the graph:

- $e(v_2) = 1$
- $e(v_1) = 2$
- $e(v_6) = 4$
- $e(v_3) = 3$
- $e(v_4) = 5$
- $e(v_5) = 6$

The bandwidth is 2.

Thus, the example illustrates how bandwidth can vary between different graph configurations.
Example

\[
\begin{align*}
\text{bandwidth} &= 5 \\
\begin{pmatrix}
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}
\end{align*}
\]

\[
\begin{align*}
e(v1) &= 1 \\
e(v2) &= 2 \\
e(v3) &= 3 \\
e(v4) &= 4 \\
e(v5) &= 5 \\
e(v6) &= 6
\end{align*}
\]
Example

Bandwidth

Pathwidth

Treewidth

k-Trees

Applications

Walter Unger 31.5.2016 14:35  SS2016

\[
\begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

Bandwidth = 5

\[
\begin{bmatrix}
1 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

Bandwidth = 2
Example: second view

```
\begin{align*}
e(v1) &= 1 \\
e(v2) &= 2 \\
e(v3) &= 3 \\
e(v4) &= 4 \\
e(v5) &= 5 \\
e(v6) &= 6 \\
\end{align*}
```

Bandwidth = 5

```
\begin{align*}
e(v1) &= 2 \\
e(v2) &= 1 \\
e(v3) &= 3 \\
e(v4) &= 5 \\
e(v5) &= 6 \\
e(v6) &= 4 \\
\end{align*}
```

Bandwidth = 2
Example: second view

- **Bandwidth**: The bandwidth of a graph is the minimum number of edges needed to cover all vertices such that each vertex is covered by at most one edge. In the first view, the bandwidth is 5, and in the second view, it is 2.

Graph 1:
- `v1` is connected to `v2` and `v3`.
- `v2` is connected to `v1` and `v3`.
- `v3` is connected to `v1` and `v2`.
- `v4` is connected to `v3` and `v5`.
- `v5` is connected to `v4` and `v6`.
- `v6` is connected to `v5`.

Graph 2:
- `v1` is connected to `v2` and `v3`.
- `v2` is connected to `v1` and `v3`.
- `v3` is connected to `v1` and `v2`.
- `v4` is connected to `v3` and `v5`.
- `v5` is connected to `v4` and `v6`.
- `v6` is connected to `v5`.

In both views, the bandwidth is calculated by the sum of the edge weights: `Σ = v1 + v2 + v3 + v4 + v5 + v6`.

**Bandwidth**: 5

**Bandwidth**: 2

**V**
- v1
- v2
- v3
- v4
- v5
- v6
Example: second view

\[ e(v_2) = 2 \]
\[ e(v_1) = 1 \]
\[ e(v_6) = 6 \]
\[ e(v_5) = 5 \]
\[ e(v_3) = 3 \]
\[ e(v_4) = 4 \]
Example: second view

\[ e(v_2) = 2 \]
\[ e(v_1) = 1 \]
\[ e(v_6) = 6 \]
\[ e(v_3) = 3 \]
\[ e(v_4) = 4 \]
\[ e(v_5) = 5 \]

\[ \text{bandwidth} = 5 \]

\[ e(v_2) = 1 \]
\[ e(v_1) = 2 \]
\[ e(v_6) = 4 \]
\[ e(v_3) = 3 \]
\[ e(v_4) = 5 \]
\[ e(v_5) = 6 \]

\[ \text{bandwidth} = 2 \]
Definition (Bandwidth-Problem)

The bandwidth-problem for a graph is:

- **Input:** A graph $G = (V, E)$ and a $k \in \mathbb{N}$.
- **Output:** Does $bw(G) \leq k$ hold?
Definition (Bandwidth-Problem)

The bandwidth-problem for a graph is:

- Input: A graph $G = (V, E)$ and a $k \in \mathbb{N}$.
- Output: Does $bw(G) \leq k$ hold?

Theorem

*The bandwidth-problem is NP-complete.*
Definition (Caterpillar)

A Caterpillar is a tree where all nodes of degree ≥ 3 are on a path.
Bandwidth on Caterpillars

**Definition (Caterpillar)**

A Caterpillar is a tree where all nodes of degree $\geq 3$ are on a path.

\[ \Sigma = \sum \]

**Theorem**

*The bandwidth-problem is NP-complete on caterpillars.*
**Definition (bandwidth-problem)**

The $k$-Bandwidth-problem on a graph is:

- **Input:** A graph $G = (V, E)$.
- **Output:** Does $bw(G) \leq k$ hold?
**Definition (bandwidth-problem)**

The $k$-Bandwidth-problem on a graph is:

- **Input:** A graph $G = (V, E)$.
- **Output:** Does $bw(G) \leq k$ hold?

**Theorem**

*The $k$-Bandwidth-problem can be solved in linear time.*
Definition (bandwidth-problem)

The $k$-Bandwidth-problem on a graph is:
- Input: A graph $G = (V, E)$.
- Output: Does $bw(G) \leq k$ hold?

Theorem

The $k$-Bandwidth-problem can be solved in linear time.

Theorem

Let $G = (V, E)$ be a graph with $bw(G) = k$, the following problem may be solved in linear time:
- Independent-Set, Clique, Vertex-Cover
- Colouring-problem
- Hamilton-Cycle, Hamilton-Path
Idea for this

- Let $bw(G) = k$. 
Idea for this

- Let $bw(G) = k$.
- Let the nodes be sorted by the labeling. I.e $e(v_i) = i$. 
Idea for this

- Let $bw(G) = k$.
- Let the nodes be sorted by the labeling. I.e $e(v_i) = i$.
- Consider block $B_i = \{v_i, v_{i+1}, v_{i+2}, \ldots, v_{i+k}\}$. 
Idea for this

- Let $bw(G) = k$.
- Let the nodes be sorted by the labeling. I.e $e(v_i) = i$.
- Consider block $B_i = \{v_i, v_{i+1}, v_{i+2}, \ldots, v_{i+k}\}$.
- There is no edge from a node to the left of $B_i$ to a node on the right of $B_i$. 
Idea for this

- Let $bw(G) = k$.
- Let the nodes be sorted by the labeling. I.e $e(v_i) = i$.
- Consider block $B_i = \{v_i, v_{i+1}, v_{i+2}, \ldots, v_{i+k}\}$.
- There is no edge from a node to the left of $B_i$ to a node on the right of $B_i$.
- I.e. there is no edge from a node $v_a$ to a node $v_b$ with $a < i$ and $b > i + k$. 
Idea for this

- Let $bw(G) = k$.
- Let the nodes be sorted by the labeling. I.e $e(v_i) = i$.
- Consider block $B_i = \{v_i, v_{i+1}, v_{i+2}, \ldots, v_{i+k}\}$.
- There is no edge from a node to the left of $B_i$ to a node on the right of $B_i$.
- I.e. there is no edge from a node $v_a$ to a node $v_b$ with $a < i$ and $b > i + k$.
- This means: any “information” must pass $B_i$. 
Idea for this

- Let $bw(G) = k$.
- Let the nodes be sorted by the labeling. I.e $e(v_i) = i$.
- Consider block $B_i = \{v_i, v_{i+1}, v_{i+2}, \ldots, v_{i+k}\}$.
- There is no edge from a node to the left of $B_i$ to a node on the right of $B_i$.
- I.e. there is no edge from a node $v_a$ to a node $v_b$ with $a < i$ and $b > i + k$.
- This means: any “information” must pass $B_i$.
- This calls for a solution using dynamic programming.
Idea for this

- Let $bw(G) = k$.
- Let the nodes be sorted by the labeling. I.e $e(v_i) = i$.
- Consider block $B_i = \{v_i, v_{i+1}, v_{i+2}, \ldots, v_{i+k}\}$.
- There is no edge from a node to the left of $B_i$ to a node on the right of $B_i$.
- I.e. there is no edge from a node $v_a$ to a node $v_b$ with $a < i$ and $b > i + k$.
- This means: any “information” must pass $B_i$.
- This calls for a solution using dynamic programming.
- Code on $B_i$ all possible solution for $v_1, v_2, \ldots, v_{i+k}$.
Idea for this

- Let \( bw(G) = k \).
- Let the nodes be sorted by the labeling. I.e. \( e(v_i) = i \).
- Consider block \( B_i = \{v_i, v_{i+1}, v_{i+2}, \ldots, v_{i+k}\} \).
- There is no edge from a node to the left of \( B_i \) to a node on the right of \( B_i \).
- I.e. there is no edge from a node \( v_a \) to a node \( v_b \) with \( a < i \) and \( b > i + k \).
- This means: any “information” must pass \( B_i \).
- This calls for a solution using dynamic programing.
- Code on \( B_i \) all possible solution for \( v_1, v_2, \ldots, v_{i+k} \).
- Compute all possible solutions for the nodes \( v_1, v_2, \ldots, v_{i+k+1} \) by using the data on \( B_i \) and code them on \( B_{i+1} \).
3-Colouring
3-Colouring
3-Colouring
3-Colouring
3-Colouring

\[ \Sigma = 0 \]
3-Colouring
3-Colouring
3-Colouring
3-Colouring

\[ \Sigma = 0 \]
3-Colouring
3-Colouring
3-Colouring
3-Colouring

\begin{align*}
\varepsilon &= 0
\end{align*}
3-Colouring

\[ \Sigma = 0 \]
3-Colouring
3-Colouring
3-Colouring
3-Colouring
3-Colouring
**g-Colouring**

- **Input:** $G = (V, E)$ with $bw(G) \leq k$:

  \[ V = \{v_1, v_2, \ldots, n\} \text{ and } E \subseteq \{\{v_i, v_j\} \mid i < j \leq i + k\} \]
**g-Colouring**

**Input:** $G = (V, E)$ with $bw(G) \leq k$:

$V = \{v_1, v_2, \ldots, n\}$ and $E \subset \{\{v_i, v_j\} \mid i < j \leq i + k\}$

**Data structure $C_i$ is defined by:** $(c_0, c_1, \ldots c_k) \in C_i \iff \exists g$-Colouring $c$ of $\{v_1, v_2, \ldots, v_{i+k}\}$: $\forall j\{0, \ldots, k\}: c_j = c(v_{i+j})$
g-Colouring

- Input: $G = (V, E)$ with $\text{bw}(G) \leq k$:

  $V = \{v_1, v_2, \ldots, n\}$ and $E \subseteq \{\{v_i, v_j\} \mid i < j \leq i + k\}$

- Data structure $C_i$ is defined by: $(c_0, c_1, \ldots c_k) \in C_i \iff$

  $\exists g\text{-Colouring } c \text{ of } \{v_1, v_2, \ldots, v_{i+k}\}: \forall j \{0, \ldots, k\} : c_j = c(v_{i+j})$

- Compute $C_1$ by: $(c_0, c_1, \ldots c_k) \in C_1 \iff$

  $\exists g\text{-Colouring } c \text{ of } \{v_1, v_2, \ldots, v_{1+k}\}: \forall j \in \{0, \ldots, k\} : c_j = c(v_{1+j})$
g-Colouring

- **Input:** $G = (V, E)$ with $\text{bw}(G) \leq k$:
  
  $V = \{v_1, v_2, \ldots, n\}$ and $E \subset \{\{v_i, v_j\} \mid i < j \leq i + k\}$

- Data structure $C_i$ is defined by: $(c_0, c_1, \ldots, c_k) \in C_i \iff$
  
  $\exists g$-Colouring $c$ of $\{v_1, v_2, \ldots, v_{i+k}\}$: $\forall j \in \{0, \ldots, k\}$: $c_j = c(v_{i+j})$

- Compute $C_1$ by: $(c_0, c_1, \ldots, c_k) \in C_1 \iff$
  
  $\exists g$-Colouring $c$ of $\{v_1, v_2, \ldots, v_{1+k}\}$: $\forall j \in \{0, \ldots, k\}$: $c_j = c(v_{1+j})$

- Compute $C_{i+1}$ from $C_i$ by: $(c_0, c_1, \ldots, c_k) \in C_{i+1} \iff$
  
  $\exists c' : (c', c_0, c_1, \ldots, c_{k-1}) \in C_i$
  
  $\forall j \in \{0, \ldots, k - 1\}$: $\{v_{i+j}, v_{i+k}\} \in E \Rightarrow c_i \neq c_k$
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set

\[ \Sigma = 0 \]
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set •
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set •
Independent Set
Independent Set

\[ \Sigma = 0 \]
Independent Set
Bandwidth
Pathwidth
Treewidth
k-Trees
Applications

4:11 Problems 24/55

Walter Unger 31.5.2016 14:35 SS2016

Independent Set
Independent Set
Independent Set

\[ \Sigma = 0 \]
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set •
Independent Set
Independent Set
Independent Set

- **Input**: $G = (V, E)$ with $bw(G) \leq k$:

  $$V = \{v_1, v_2, \ldots, n\} \text{ and } E \subset \{\{v_i, v_j\} \mid i < j \leq i + k\}$$
Independent Set

- **Input:** $G = (V, E)$ with $bw(G) \leq k$:

\[ V = \{v_1, v_2, \ldots, n\} \text{ and } E \subset \{\{v_i, v_j\} \mid i < j \leq i + k\} \]

- **Data structure** $C_i$ is defined by: $(I, s) \in C_i \iff$

\[ \exists S \subset \{v_1, \ldots, v_{i+k}\} : S \cap \{v_i, \ldots v_{i+k}\} = I, |S| = s, S \text{ is independent set} \]
**Independent Set**

- **Input:** $G = (V, E)$ with $\text{bw}(G) \leq k$:

  $V = \{v_1, v_2, \ldots, n\}$ and $E \subset \{\{v_i, v_j\} \mid i < j \leq i + k\}$

- **Data structure** $C_i$ is defined by: $(I, s) \in C_i \iff 

  \exists S \subset \{v_1, \ldots, v_{i+k}\} : S \cap \{v_i, \ldots, v_{i+k}\} = I, |S| = s, S$ is independent set

- **Compute** $C_1$ by: $(I, s) \in C_1 \iff 

  \exists I \subset \{v_1, \ldots, v_{1+k}\} : |I| = s, I$ is independent set
Independent Set

- **Input:** \( G = (V, E) \) with \( bw(G) \leq k \):

  \[ V = \{v_1, v_2, \ldots, n\} \text{ and } E \subset \{\{v_i, v_j\} \mid i < j \leq i + k\} \]

- **Data structure** \( C_i \) is defined by: 
  \( (I, s) \in C_i \iff \exists S \subset \{v_1, \ldots, v_{i+k}\} : S \cap \{v_i, \ldots, v_{i+k}\} = I, |S| = s, S \) is independent set

- **Compute** \( C_1 \) by: 
  \( (I, s) \in C_1 \iff \exists \exists l \subset \{v_1, \ldots, v_{1+k}\} : |I| = s, l \) is independent set

- **Compute** \( C_{i+1} \) from \( C_i \) by: 
  \( (I, s) \in C_{i+1} \iff \exists (I', s') \in C_i \)

  \[ I = I' \setminus \{v_i\}, s = s' \text{ or } \]

  \[ I = (I' \cup \{v_{i+k+1}\}) \setminus \{v_i\}, s = s' + 1, I \text{ is stable set} \]
Hamilton Cycle

\[ \sum = 0 \]
Hamilton Cycle

- **v₀** (Red) - Open Endpoint
- **v₁** (Green) - Visited Node
- **v₂** (Green) - Visited Node
- **v₃** (Red) - Visited Node
- **v₄** (Green) - Visited Node
- **v₅** (Yellow) - Nonvisited Node
- **v₆** (Green) - Visited Node
- **v₇** (Green) - Visited Node
- **v₈** (Green) - Visited Node
- **v₉** (Green) - Visited Node

**Σ = 0**
Hamilton Cycle

![Diagram of a Hamilton cycle with nodes v0, v1, v2, v3, v4, v5, v6, v7, v8, v9, showing open endpoints, visited nodes, and nonvisited nodes.](image)
Hamilton Cycle

- $v_0$, $v_1$, $v_2$, $v_3$, $v_4$, $v_5$, $v_6$, $v_7$, $v_8$, $v_9$
- Open Endpoint
- Visited Node
- Nonvisited Node

$\Sigma = 0$
Hamilton Cycle

\[ \Sigma = 0 \]
Hamilton Cycle
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node

Graph showing a Hamilton Cycle with nodes labeled from v0 to v9, indicating the cycle path.
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node
Hamilton Cycle

Σ = 0
Hamilton Cycle

Diagram showing a cycle with nodes labeled v0 to v9. Nodes are colored to indicate whether they are open endpoints, visited nodes, or nonvisited nodes.
Hamilton Cycle

- $v_0$, $v_1$, $v_2$, $v_3$, $v_4$, $v_5$, $v_6$, $v_7$, $v_8$, $v_9$

- Open Endpoint
- Visited Node
- Nonvisited Node

$\Sigma = 0$
Hamilton Cycle

$v_0$ $v_1$ $v_2$ $v_3$ $v_4$ $v_5$ $v_6$ $v_7$ $v_8$ $v_9$

- Open Endpoint
- Visited Node
- Nonvisited Node
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node
Hamilton Cycle

Graph showing a Hamiltonian cycle with nodes labeled from $v_0$ to $v_9$. The cycle visits each node exactly once and returns to the starting node. The diagram includes nodes with different colors:
- Open Endpoint: Green
- Visited Node: Red
- Nonvisited Node: Yellow

The cycle path is indicated by the connected nodes, starting at $v_0$ and ending at $v_0$ to complete the cycle.
Hamilton Cycle
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node
Hamilton Cycle

\[
\sum = 0
\]
Hamilton Cycle

$\Sigma = 0$
Hamilton Cycle
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node
Hamilton Cycle

- **v0**: Open Endpoint
- **v1**: Visited Node
- **v2**: Nonvisited Node
- **v3**: Open Endpoint
- **v4**: Visited Node
- **v5**: Nonvisited Node
- **v6**: Open Endpoint
- **v7**: Visited Node
- **v8**: Nonvisited Node
- **v9**: Open Endpoint
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node
Hamilton Cycle
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node

Graph showing a Hamiltonian cycle with nodes v0 to v9 and connections indicating the cycle. The diagram uses colors to differentiate between open endpoints, visited nodes, and nonvisited nodes.
Hamilton Cycle

- **Open Endpoint**: Green
- **Visited Node**: Red
- **Nonvisited Node**: Yellow

Diagram showing a cycle through nodes labeled $v_0$ to $v_9$.
Hamilton Cycle

- $v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9$
- Open Endpoint
- Visited Node
- Nonvisited Node

$\Sigma = 0$
Hamilton Cycle
Hamilton Cycle
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node
Hamilton Cycle
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node
Hamilton Cycle

- **v0**: Open Endpoint
- **v8**, **v9**: Nonvisited Node

Legend:
- Green: Open Endpoint
- Red: Visited Node
- Yellow: Nonvisited Node
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node

$\Sigma = 0$
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node
Hamilton Cycle

\[ \Sigma = 0 \]
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node
Hamilton-Cycle

- **Input:** $G = (V, E)$ with $bw(G) \leq k$:

  \[ V = \{v_1, v_2, \ldots, n\} \text{ and } E \subset \{\{v_i, v_j\} \mid i < j \leq i+k\} \]
Hamilton-Cycle

- **Input:** $G = (V, E)$ with $bw(G) \leq k$:
  
  $V = \{v_1, v_2, \ldots, n\}$ and $E \subset \{\{v_i, v_j\} \mid i < j \leq i + k\}$

- **Data structure** $C_i$ describes $[0,2]$-factors in $\{v_1, v_2, \ldots, v_{i+k}\}$:
Hamilton-Cycle

- **Input**: $G = (V, E)$ with $\text{bw}(G) \leq k$:
  
  $V = \{v_1, v_2, \ldots, n\}$ and $E \subset \{\{v_i, v_j\} \mid i < j \leq i + k\}$

- **Data structure $C_i$** describes $[0,2]$-factors in $\{v_1, v_2, \ldots, v_{i+k}\}$:
  
  $H = (\{v_1, v_2, \ldots, v_{i+k}\}, F)$ is a $[0,2]$-factor in $\{v_1, v_2, \ldots, v_{i+k}\}$.
Hamilton-Cycle

- **Input:** \( G = (V, E) \) with \( \text{bw}(G) \leq k \):
  \[
  V = \{v_1, v_2, \ldots, n\} \text{ and } E \subset \{\{v_i, v_j\} \mid i < j \leq i + k\}
  \]

- Data structure \( C_i \) describes \([0,2]\)-factors in \( \{v_1, v_2, \ldots, v_{i+k}\} \):
  - \( H = (\{v_1, v_2, \ldots, v_{i+k}\}, F) \) is a \([0,2]\)-factor in \( \{v_1, v_2, \ldots, v_{i+k}\} \).
  - \( \delta_H(v_j) = 2 \) for all \( j \in \{1, 2, \ldots, i - 1\} \).
Hamilton-Cycle

- **Input:** $G = (V, E)$ with $\text{bw}(G) \leq k$:
  
  $$V = \{v_1, v_2, \ldots, n\} \text{ and } E \subset \{\{v_i, v_j\} | i < j \leq i + k\}$$

- **Data structure $C_i$** describes $[0,2]$-factors in $\{v_1, v_2, \ldots, v_{i+k}\}$:
  
  - $H = (\{v_1, v_2, \ldots, v_{i+k}\}, F)$ is a $[0,2]$-factor in $\{v_1, v_2, \ldots, v_{i+k}\}$.
  - $\delta_H(v_j) = 2$ for all $j \in \{1, 2, \ldots, i - 1\}$.
  - I.e. each component in $H$ is path.
Hamilton-Cycle

- **Input:** $G = (V, E)$ with $\text{bw}(G) \leq k$: 
  
  $V = \{v_1, v_2, \ldots, n\}$ and $E \subset \{\{v_i, v_j\} \mid i < j \leq i + k\}$

- **Data structure $C_i$ describes [0,2]-factors in $\{v_1, v_2, \ldots, v_{i+k}\}$:**
  - $H = (\{v_1, v_2, \ldots, v_{i+k}\}, F)$ is a [0,2]-factor in $\{v_1, v_2, \ldots, v_{i+k}\}$.
  - $\delta_H(v_j) = 2$ for all $j \in \{1, 2, \ldots, i-1\}$.
  - I.e. each component in $H$ is path.
  - For each component $C \ \exists j, i \leq j \leq i + k : \delta_H(v_j) = 1$ und $v_j \in C$. 

Data structure $C_i$ describes [0,2]-factors in $\{v_1, v_2, \ldots, v_{i+k}\}$:
Hamilton-Cycle

- **Input:** \( G = (V, E) \) with \( \text{bw}(G) \leq k \):
  \[
  V = \{v_1, v_2, \ldots, n\} \quad \text{and} \quad E \subset \{\{v_i, v_j\} \mid i < j \leq i + k\}
  \]

- **Data structure** \( C_i \) describes \([0,2]\)-factors in \( \{v_1, v_2, \ldots, v_{i+k}\} \):
  - \( H = (\{v_1, v_2, \ldots, v_{i+k}\}, F) \) is a \([0,2]\)-factor in \( \{v_1, v_2, \ldots, v_{i+k}\} \).
  - \( \delta_H(v_j) = 2 \) for all \( j \in \{1, 2, \ldots, i-1\} \).
  - I.e. each component in \( H \) is path.
  - For each component \( C \) \( \exists j, i \leq j \leq i + k : \delta_H(v_j) = 1 \) und \( v_j \in C \).
  - I.e. each component in \( H \) is a path with endpoints in \( \{v_i, v_{i+1}, \ldots, v_{i+k}\} \).
Hamilton-Cycle

- **Input:** $G = (V, E)$ with $bw(G) \leq k$:
  \[
  V = \{v_1, v_2, \ldots, n\} \text{ and } E \subset \{\{v_i, v_j\} \mid i < j \leq i + k\}
  \]

- **Data structure $C_i$ describes $[0,2]$-factors in $\{v_1, v_2, \ldots, v_{i+k}\}$:**
  - $H = (\{v_1, v_2, \ldots, v_{i+k}\}, F)$ is a $[0,2]$-factor in $\{v_1, v_2, \ldots, v_{i+k}\}$.
  - $\delta_H(v_j) = 2$ for all $j \in \{1, 2, \ldots, i-1\}$.
  - I.e. each component in $H$ is path.
  - For each component $C \exists j, i \leq j \leq i + k : \delta_H(v_j) = 1$ und $v_j \in C$.
  - I.e. each component in $H$ is a path with endpoints in $\{v_i, v_{i+1}, \ldots, v_{i+k}\}$.

- **Problem has solution, if**
Hamilton-Cycle

- **Input:** \( G = (V, E) \) with \( \text{bw}(G) \leq k \):
  \[
  V = \{v_1, v_2, \ldots, n\} \text{ and } E \subset \{\{v_i, v_j\} \mid i < j \leq i + k\}
  \]

- **Data structure** \( C_i \) describes \([0,2]\)-factors in \( \{v_1, v_2, \ldots, v_{i+k}\} \):
  - \( H = (\{v_1, v_2, \ldots, v_{i+k}\}, F) \) is a \([0,2]\)-factor in \( \{v_1, v_2, \ldots, v_{i+k}\} \).
  - \( \delta_H(v_j) = 2 \) for all \( j \in \{1, 2, \ldots, i - 1\} \).
  - I.e. each component in \( H \) is path.
  - For each component \( C \) \( \exists j, i \leq j \leq i + k : \delta_H(v_j) = 1 \) und \( v_j \in C \).
  - I.e. each component in \( H \) is a path with endpoints in \( \{v_i, v_{i+1}, \ldots, v_{i+k}\} \).

- **Problem has solution, if**
  - \( H = (\{v_1, v_2, \ldots, v_{i+k}\}, F) \) is \([1,2]\)-Factor in \( \{v_1, v_2, \ldots, v_n\} \).
Hamilton-Cycle

- **Input:** $G = (V, E)$ with $\text{bw}(G) \leq k$:
  
  $$V = \{v_1, v_2, \ldots, n\} \text{ and } E \subseteq \{\{v_i, v_j\} \mid i < j \leq i + k\}$$

- **Data structure $C_i$ describes [0,2]-factors in $\{v_1, v_2, \ldots, v_{i+k}\}$:**
  - $H = (\{v_1, v_2, \ldots, v_{i+k}\}, F)$ is a [0,2]-factor in $\{v_1, v_2, \ldots, v_{i+k}\}$.
  - $\delta_H(v_j) = 2$ for all $j \in \{1, 2, \ldots, i - 1\}$.
  - I.e. each component in $H$ is path.
  - For each component $C \exists j, i \leq j \leq i + k : \delta_H(v_j) = 1$ und $v_j \in C$.
  - I.e. each component in $H$ is a path with endpoints in $\{v_i, v_{i+1}, \ldots, v_{i+k}\}$.

- **Problem has solution, if**
  - $H = (\{v_1, v_2, \ldots, v_{i+k}\}, F)$ is [1,2]-Factor in $\{v_1, v_2, \ldots, v_n\}$.
  - $\exists a, b : n - k \leq a, b \leq n$: 
Hamilton-Cycle

- **Input:** \( G = (V, E) \) with \( \text{bw}(G) \leq k \):
  
  \[
  V = \{v_1, v_2, \ldots, n\} \text{ and } E \subset \{\{v_i, v_j\} \mid i < j \leq i + k\}
  \]

- **Data structure** \( C_i \) describes \([0,2]\)-factors in \( \{v_1, v_2, \ldots, v_{i+k}\} \):
  
  - \( H = (\{v_1, v_2, \ldots, v_{i+k}\}, F) \) is a \([0,2]\)-factor in \( \{v_1, v_2, \ldots, v_{i+k}\} \).
  - \( \delta_H(v_j) = 2 \) for all \( j \in \{1, 2, \ldots, i - 1\} \).
  - I.e. each component in \( H \) is path.
  - For each component \( C \exists j, i \leq j \leq i + k : \delta_H(v_j) = 1 \) und \( v_j \in C \).
  - I.e. each component in \( H \) is a path with endpoints in \( \{v_i, v_{i+1}, \ldots, v_{i+k}\} \).

- **Problem has solution, if**
  
  - \( H = (\{v_1, v_2, \ldots, v_{i+k}\}, F) \) is \([1,2]\)-Factor in \( \{v_1, v_2, \ldots, v_n\} \).
  - \( \exists a, b : n - k \leq a, b \leq n \):
    
    \( \forall j \in \{1, 2, \ldots, n\} \setminus \{a, b\} : \delta_H(v_{a}) = 2. \)
Hamilton-Cycle

- **Input:** $G = (V, E)$ with $\text{bw}(G) \leq k$:
  
  $V = \{v_1, v_2, \ldots, n\}$ and $E \subseteq \{\{v_i, v_j\} \mid i < j \leq i + k\}$

- **Data structure** $C_i$ describes $[0,2]$-factors in $\{v_1, v_2, \ldots, v_{i+k}\}$:
  - $H = (\{v_1, v_2, \ldots, v_{i+k}\}, F)$ is a $[0,2]$-factor in $\{v_1, v_2, \ldots, v_{i+k}\}$.
  - $\delta_H(v_j) = 2$ for all $j \in \{1, 2, \ldots, i-1\}$.
  - I.e. each component in $H$ is a path.
  - For each component $C$ $\exists j, i \leq j \leq i + k : \delta_H(v_j) = 1$ und $v_j \in C$.
  - I.e. each component in $H$ is a path with endpoints in $\{v_i, v_{i+1}, \ldots, v_{i+k}\}$.

- **Problem has solution, if**
  - $H = (\{v_1, v_2, \ldots, v_{i+k}\}, F)$ is $[1,2]$-Factor in $\{v_1, v_2, \ldots, v_n\}$.
  - $\exists a, b : n - k \leq a, b \leq n$:
    - $\forall j \in \{1, 2, \ldots, n\} \setminus \{a, b\} : \delta_H(v_a) = 2$.
    - $\delta_H(v_a) = \delta_H(v_b) = 1$. 

Hamilton-Cycle

- Input: $G = (V, E)$ with $bw(G) \leq k$:
  
  $$V = \{v_1, v_2, \ldots, n\} \text{ and } E \subset \{\{v_i, v_j\} \mid i < j \leq i + k\}$$

- Data structure $C_i$ describes $[0,2]$-factors in $\{v_1, v_2, \ldots, v_{i+k}\}$:
  - $H = (\{v_1, v_2, \ldots, v_{i+k}\}, F)$ is a $[0,2]$-factor in $\{v_1, v_2, \ldots, v_{i+k}\}$.
  - $\delta_H(v_j) = 2$ for all $j \in \{1, 2, \ldots, i-1\}$.
  - I.e. each component in $H$ is path.
  - For each component $C \exists j, i \leq j \leq i + k : \delta_H(v_j) = 1$ und $v_j \in C$.
  - I.e. each component in $H$ is a path with endpoints in $\{v_i, v_{i+1}, \ldots, v_{i+k}\}$.

- Problem has solution, if
  - $H = (\{v_1, v_2, \ldots, v_{i+k}\}, F)$ is $[1,2]$-Factor in $\{v_1, v_2, \ldots, v_n\}$.
  - $\exists a, b : n - k \leq a, b \leq n$
    - $\forall j \in \{1, 2, \ldots, n\} \setminus \{a, b\} : \delta_H(v_a) = 2$.
    - $\delta_H(v_a) = \delta_H(v_b) = 1$.
    - $\{v_a, v_b\} \in E$.  

---

**Bandwidth**

**Pathwidth**

**Treewidth**

**k-Trees**

**Applications**

Walter Unger 31.5.2016 14:35  
SS2016  
RWTH
Lower Bound on Bandwidth

**Definition (Diameter and Radius)**

- The diameter of $G = (V, E)$ is:
  \[
  \text{diam}(G) = \max \{ \text{dist}(v, w) | v, w \in V \}.
  \]
Lower Bound on Bandwidth

Definition (Diameter and Radius)

- The diameter of $G = (V, E)$ is:
  \[\text{diam}(G) = \max\{\text{dist}(v, w) \mid v, w \in V\}\.\]

- The radius of a node $v \in V$ is:
  \[\text{rad}(v, G) = \max\{\text{dist}(v, x) \mid x \in V\}\/\]
Lower Bound on Bandwidth

Definition (Diameter and Radius)

- The diameter of \( G = (V, E) \) is:
  \[
  \text{diam}(G) = \max\{\text{dist}(v, w) \mid v, w \in V\}.
  \]

- The radius of a node \( v \in V \) is:
  \[
  \text{rad}(v, G) = \max\{\text{dist}(v, x) \mid x \in V\}.
  \]

- The radius of \( G \) is:
  \[
  \text{rad}(G) = \min\{\text{rad}(v, G) \mid v \in V\}.
  \]
Lower Bound for Bandwidth

**Theorem (Lower Bound for Bandwidth)**

Let $G = (V, E)$ be a graph with $n = |V|$ nodes. Then the following hold:

$$bw(G) \geq \left\lceil \frac{n - 1}{\text{diam}(G)} \right\rceil$$
Lower Bound for Bandwidth

Theorem (Lower Bound for Bandwidth)

Let $G = (V, E)$ be a graph with $n = |V|$ nodes. Then the following hold:

$$bw(G) \geq \left\lceil \frac{n - 1}{\text{diam}(G)} \right\rceil$$

Theorem (Lower Bound for Bandwidth of a Complete Tree)

Let $T = (V, E)$ be a complete tree with depth $k$. Then the following hold:

$$bw(G) \geq \left\lceil \frac{2^k - 1}{k} \right\rceil$$
Lower Bound for Bandwidth

Theorem (Lower Bound for Bandwidth)

Let $G = (V, E)$ be a graph with $n = |V|$ nodes. Then the following hold:

\[
\text{bw}(G) \geq \left\lceil \frac{n - 1}{\text{diam}(G)} \right\rceil
\]

Theorem (Lower Bound for Bandwidth of a Complete Tree)

Let $T = (V, E)$ be a complete tree with depth $k$. Then the following hold:

\[
\text{bw}(G) \geq \left\lceil \frac{2^k - 1}{k} \right\rceil = \left\lceil \frac{2^{k+1} - 2}{2k} \right\rceil.
\]
Theorem (Upper Bound for Bandwidth of the Complete Binary Tree)

Let $T = (V, E)$ be a complete binary tree with depth $k$, then the following hold:

$$bw(T) = \left\lceil \frac{2^k - 1}{k} \right\rceil.$$
Hardness of the Bandwidth Problem

**Theorem**

For $\varepsilon > 0$ it is not possible to approximate the bandwidth-problem by a factor of $2 - \varepsilon$, under the assumption $\mathcal{P} \neq \mathcal{NP}$. 
Hardness of the Bandwidth Problem

Theorem

For $\varepsilon > 0$ it is not possible to approximate the bandwidth problem by a factor of $2 - \varepsilon$, under the assumption $\mathcal{P} \neq \mathcal{NP}$.

Theorem

It is not possible to approximate the bandwidth problem by a constant factor of $k$, under the assumption $\mathcal{P} \neq \mathcal{NP}$.
Hardness of the Bandwidth Problem

Theorem

For $\varepsilon > 0$ it is not possible to approximate the bandwidth-problem by a factor of $2 - \varepsilon$, under the assumption $P \neq NP$.

Theorem

It is not possible to approximate the bandwidth-problem by a constant factor of $k$, under the assumption $P \neq NP$.

Theorem

It is not possible to approximate the bandwidth-problem for caterpillars by a constant factor of $k$, under the assumption $P \neq NP$. 
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwith (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwith (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwith (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwidth (Idea)
Hardness of Bandwith (Idea)
Hardness of Bandwith (Idea)
Hardness of Bandwidth (Idea)
Idea of Pathwidth

Σ = 0
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth

Σ = 0
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth

Σ = 0
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth

Σ = 0
Idea of Pathwidth
Idea of Pathwidth
Pathwidth

Definition

A graph $G = (V, E)$ has pathwidth $k$, iff there is a path $P = (V_p, E_p)$ and a mapping $f : V_p \rightarrow \mathcal{P}(V)$ with:

- $\forall (a, b) \in E : \exists x \in V_p : a, b \in f(x)$
A graph $G = (V, E)$ has pathwidth $k$, iff there is a path $P = (V_p, E_p)$ and a mapping $f : V_p \rightarrow \mathcal{P}(V)$ with:

- $\forall (a, b) \in E : \exists x \in V_p : a, b \in f(x)$
- If $c$ is on the path from $a$ to $b$ on $P$, then does $f(b) \cap f(a) \subset f(c)$ hold.
Pathwidth

Definition

A graph \( G = (V, E) \) has pathwidth \( k \), iff there is a path \( P = (V_p, E_p) \) and a mapping \( f : V_p \rightarrow \mathcal{P}(V) \) with:

- \( \forall (a, b) \in E : \exists x \in V_p : a, b \in f(x) \)
- If \( c \) is on the path from \( a \) to \( b \) on \( P \), then does \( f(b) \cap f(a) \subset f(c) \) hold.
- \( \forall x \in V_p : |f(x)| \leq k + 1. \)
### Pathwidth

**Definition**

A graph \( G = (V, E) \) has pathwidth \( k \), iff there is a path \( P = (V_p, E_p) \) and a mapping \( f : V_p \rightarrow \mathcal{P}(V) \) with:

1. \( \forall (a, b) \in E : \exists x \in V_p : a, b \in f(x) \)
2. If \( c \) is on the path from \( a \) to \( b \) on \( P \), then does \( f(b) \cap f(a) \subseteq f(c) \) hold.
3. \( \forall x \in V_p : |f(x)| \leq k + 1. \)
Pathwidth

### Definition

A graph $G = (V, E)$ has pathwidth $k$, iff there is a path $P = (V_p, E_p)$ and a mapping $f : V_p \rightarrow \mathcal{P}(V)$ with:

- $\forall (a, b) \in E : \exists x \in V_p : a, b \in f(x)$
- If $c$ is on the path from $a$ to $b$ on $P$, then does $f(b) \cap f(a) \subset f(c)$ hold.
- $\forall x \in V_p : |f(x)| \leq k + 1$.

and for $k - 1$ exists no such function $f$ and path $P$.

Notation: $\text{pw}(G) = k$. 
**Definition**

A graph $G = (V, E)$ has pathwidth $k$, iff there is a path $P = (V_p, E_p)$ and a mapping $f : V_p \rightarrow \mathcal{P}(V)$ with:

- $\forall (a, b) \in E : \exists x \in V_p : a, b \in f(x)$
- If $c$ is on the path from $a$ to $b$ on $P$, then does $f(b) \cap f(a) \subset f(c)$ hold.
- $\forall x \in V_p : |f(x)| \leq k + 1$.

and for $k - 1$ exists no such function $f$ and path $P$.

Notation: $\text{pw}(G) = k$.

**Theorem**

Let $G = (V, E)$ be a graph. Then holds: $\text{bw}(G) \geq \text{pw}(G)$. 
Pathwidth

Definition

A graph $G = (V, E)$ has pathwidth $k$, iff there is a path $P = (V_p, E_p)$ and a mapping $f : V_p \rightarrow \mathcal{P}(V)$ with:

- $\forall (a, b) \in E : \exists x \in V_p : a, b \in f(x)$
- If $c$ is on the path from $a$ to $b$ on $P$, then does $f(b) \cap f(a) \subset f(c)$ hold.
- $\forall x \in V_p : |f(x)| \leq k + 1$.

and for $k - 1$ exists no such function $f$ and path $P$.
Notation: $\text{pw}(G) = k$.

Theorem

Let $G = (V, E)$ be a graph. Then holds: $\text{bw}(G) \geq \text{pw}(G)$.

Note: $\text{bw}(K_{1,2n+1}) = n$ but $\text{pw}(K_{1,2n+1}) = 1$
Theorem

Let $G = (V, E)$ be a graph. Then does $bw(G) \geq pw(G)$ holds.
Theorem

Let $G = (V, E)$ be a graph. Then does $\text{bw}(G) \geq \text{pw}(G)$ holds.

- Let $G = (V, E)$ be a graph with $\text{bw}(G) = k$ and $|V| = n$. 
Theorems I

Theorem

Let $G = (V, E)$ be a graph. Then does $bw(G) \geq pw(G)$ holds.

- Let $G = (V, E)$ be a graph with $bw(G) = k$ and $|V| = n$.
- Show $pw(G) \leq k$. 
Theorems I

Theorem

Let $G = (V, E)$ be a graph. Then does $\text{bw}(G) \geq \text{pw}(G)$ holds.

- Let $G = (V, E)$ be a graph with $\text{bw}(G) = k$ and $|V| = n$.
- Show $\text{pw}(G) \leq k$.
- Let $e$ be an embedding function with $\text{bw}(e, G) = k$. 
Theorem

Let $G = (V, E)$ be a graph. Then does $bw(G) \geq pw(G)$ holds.

- Let $G = (V, E)$ be a graph with $bw(G) = k$ and $|V| = n$.
- Show $pw(G) \leq k$.
- Let $e$ be an embedding function with $bw(e, G) = k$.
- Let $e(v_i) = i$. 
Let $G = (V, E)$ be a graph. Then does $bw(G) \geq pw(G)$ holds.

- Let $G = (V, E)$ be a graph with $bw(G) = k$ and $|V| = n$.
- Show $pw(G) \leq k$.
- Let $e$ be an embedding function with $bw(e, G) = k$.
- Let $e(v_i) = i$.
- Let $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{p_j, p_{j+1}\} | 1 \leq j \leq n-k)$ be a path with $n-k+1$ nodes.
Theorems I

Theorem

Let $G = (V, E)$ be a graph. Then does $bw(G) \geq pw(G)$ holds.

- Let $G = (V, E)$ be a graph with $bw(G) = k$ and $|V| = n$.
- Show $pw(G) \leq k$.
- Let $e$ be an embedding function with $bw(e, G) = k$.
- Let $e(v_i) = i$.
- Let $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} | 1 \leq j \leq n - k\})$ be a path with $n - k + 1$ nodes.
- Define function $f : \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto \mathcal{P}(V)$:
Theorems I

**Theorem**

Let $G = (V, E)$ be a graph. Then does $bw(G) \geq pw(G)$ holds.

- Let $G = (V, E)$ be a graph with $bw(G) = k$ and $|V| = n$.
- Show $pw(G) \leq k$.
- Let $e$ be an embedding function with $bw(e, G) = k$.
- Let $e(v_i) = i$.
- Let $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} | 1 \leq j \leq n-k\})$ be a path with $n-k+1$ nodes.
- Define function $f : \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto P(V)$:
  - $f(p_i) = \{v_i, v_{i+1}, \ldots, v_{i+k}\}$. 
Let $G = (V, E)$ be a graph. Then does $bw(G) \geq pw(G)$ holds.

- Let $G = (V, E)$ be a graph with $bw(G) = k$ and $|V| = n$.
- Show $pw(G) \leq k$.
- Let $e$ be an embedding function with $bw(e, G) = k$.
- Let $e(v_i) = i$.
- Let $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} \mid 1 \leq j \leq n-k\})$ be a path with $n-k+1$ nodes.
- Define function $f : \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto \mathcal{P}(V)$:
- $f(p_i) = \{v_i, v_{i+1}, \ldots, v_{i+k}\}$.
- Then the following holds: $|f(p_i)| = k + 1$. 
Theorem

Let $G = (V, E)$ be a graph. Then does $bw(G) \geq pw(G)$ holds.

- Let $G = (V, E)$ be a graph with $bw(G) = k$ and $|V| = n$.
- Show $pw(G) \leq k$.
- Let $e$ be an embedding function with $bw(e, G) = k$.
- Let $e(v_i) = i$.
- Let $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} \mid 1 \leq j \leq n-k\})$ be a path with $n - k + 1$ nodes.
- Define function $f : \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto \mathcal{P}(V)$:
  \[ f(p_i) = \{v_i, v_{i+1}, \ldots, v_{i+k}\}. \]
- Then the following holds: $|f(p_i)| = k + 1$.
- And if $\{v_i, v_{i+d}\} \in E$ hold, then $\{v_i, v_{i+d}\} \subset f(p_i)$ follows.
Theorems I

Theorem

Let $G = (V, E)$ be a graph. Then does $\text{bw}(G) \geq \text{pw}(G)$ holds.

- Let $G = (V, E)$ be a graph with $\text{bw}(G) = k$ and $|V| = n$.
- Show $\text{pw}(G) \leq k$.
- Let $e$ be an embedding function with $\text{bw}(e, G) = k$.
- Let $e(v_i) = i$.
- Let $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} | 1 \leq j \leq n - k\})$ be a path with $n - k + 1$ nodes.
- Define function $f : \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto \mathcal{P}(V)$:
  
  $f(p_i) = \{v_i, v_{i+1}, \ldots, v_{i+k}\}$.
- Then the following holds: $|f(p_i)| = k + 1$.
- And if $\{v_i, v_{i+d}\} \in E$ hold, then $\{v_i, v_{i+d}\} \subset f(p_i)$ follows.
- Thus we have: $\text{pw}(G) \leq k$
Theorem

Let $G = (V, E)$ be a graph. Then the following hold:

- The problem, to compute the pathwidth of a graph, is NP-complete.
Theorem II

Let $G = (V, E)$ be a graph. Then the following hold:

- The problem, to compute the pathwidth of a graph, is NP-complete.
- For a fixed $k$ it is possible to check in linear time $O(n + m)$, if a graph has pathwidth $k$. 
Theorem

Let $G = (V, E)$ be a graph. Then the following hold:

- The problem, to compute the pathwidth of a graph, is NP-complete.
- For a fixed $k$ it is possible to check in linear time $O(n + m)$, if a graph has pathwidth $k$. 
Theorems II

Theorem

Let $G = (V, E)$ be a graph. Then the following hold:

- The problem, to compute the pathwidth of a graph, is NP-complete.
- For a fixed $k$ it is possible to check in linear time $O(n + m)$, if a graph has pathwidth $k$.

Theorem

Let $G = (V, E)$ be a graphs with $\text{pw}(G) = k$. The following problem may be solved in linear time:

- Independent-Set, Clique, Vertex-Cover
- Colouring-problem
- Hamilton-Cycle, Hamilton-Path
Example (Independent Set)

- Let $G = (V, E)$ be a graph with $\text{pw}(G) = k$ and $|V| = n$. 
Example (Independent Set)

- Let $G = (V, E)$ be a graph with $pw(G) = k$ and $|V| = n$.
- Let $P_{n-k} = \{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} | 1 \leq j \leq n-k\}$ be a path with $n - k + 1$ nodes.
Example (Independent Set)

- Let $G = (V, E)$ be a graph with $\text{pw}(G) = k$ and $|V| = n$.
- Let $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} \mid 1 \leq j \leq n - k\})$ be a path with $n - k + 1$ nodes.
- Let $f : \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto \mathcal{P}(V)$ the embedding function with pathwidth $k$. 
Example (Independent Set)

- Let $G = (V, E)$ be a graph with $\text{pw}(G) = k$ and $|V| = n$.
- Let $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} | 1 \leq j \leq n - k\})$ be a path with $n - k + 1$ nodes.
- Let $f : \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto \mathcal{P}(V)$ the embedding function with pathwidth $k$.
- For each subset $I_j \subseteq f(p_i)$ store:
Example (Independent Set)

- Let $G = (V, E)$ be a graph with $\text{pw}(G) = k$ and $|V| = n$.
- Let $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} \mid 1 \leq j \leq n - k\})$ be a path with $n - k + 1$ nodes.
- Let $f : \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto 2^V$ the embedding function with pathwidth $k$.
- For each subset $I_i \subset f(p_i)$ store:
  - $I_i$ and $w_i$ with
Example (Independent Set)

- Let $G = (V, E)$ be a graph with $\text{pw}(G) = k$ and $|V| = n$.
- Let $P_{n-k} = \{\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} \mid 1 \leq j \leq n-k\}$ be a path with $n-k+1$ nodes.
- Let $f : \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto \mathcal{P}(V)$ be the embedding function with pathwidth $k$.
- For each subset $I_i \subset f(p_i)$ store:
  - $I_i$ and $w_i$ with
  - $w_i = |I|$ the size of the largest independent set $I$ on $\bigcup_{j=1}^{i} f(p_j)$ with $I_i \subset I$. 

Note: There is no fun in this ugly task. We have to simplify.
Example (Independent Set)

- Let $G = (V, E)$ be a graph with $\text{pw}(G) = k$ and $|V| = n$.
- Let $P_{n-k} = \{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} | 1 \leq j \leq n-k\}$ be a path with $n - k + 1$ nodes.
- Let $f: \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto \mathcal{P}(V)$ the embedding function with pathwidth $k$.
- For each subset $I^j_i \subset f(p_i)$ store:
  - $I^j_i$ and $w^j_i$ with
  - $w^j_i = |I|$ the size of the largest independent set $I$ on $\bigcup_{j=1}^i f(p_j)$ with $I^j_i \subset I$.
- Iteration step on $f(p_{i+1})$:
Example (Independent Set)

- Let $G = (V, E)$ be a graph with $\text{pw}(G) = k$ and $|V| = n$.

- Let $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} | 1 \leq j \leq n - k\})$ be a path with $n - k + 1$ nodes.

- Let $f: \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto \mathcal{P}(V)$ the embedding function with pathwidth $k$.

- For each subset $I_i^j \subset f(p_i)$ store:
  - $I_i^j$ and $w_i^j$ with
  - $w_i^j = |I|$ the size of the largest independent set $I$ on $\bigcup_{j=1}^{i} f(p_j)$ with $I_i^j \subset I$.

- Iteration step on $f(p_{i+1})$:

- For each subset $I_i^{j+1}$ compute the above value:
Example (Independent Set)

- Let $G = (V, E)$ be a graph with $\text{pw}(G) = k$ and $|V| = n$.
- Let $P_{n-k} = \{\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} \mid 1 \leq j \leq n-k\}$) be a path with $n-k+1$ nodes.
- Let $f : \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto \mathcal{P}(V)$ the embedding function with pathwidth $k$.
- For each subset $I_i^j \subset f(p_i)$ store:
  - $I_i^j$ and $w_i^j$ with
  - $w_i^j = |I|$ the size of the largest independent set $I$ on $\bigcup_{j=1}^{i} f(p_j)$ with $I_i^j \subset I$.
- Iteration step on $f(p_{i+1})$:
- For each subset $I_{i+1}^j$ compute the above value:
- **No** there is no fun in this ugly task
Example (Independent Set)

- Let $G = (V, E)$ be a graph with $\text{pw}(G) = k$ and $|V| = n$.
- Let $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} \mid 1 \leq j \leq n - k\})$ be a path with $n - k + 1$ nodes.
- Let $f : \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto \mathcal{P}(V)$ the embedding function with pathwidth $k$.
- For each subset $I_i \subset f(p_i)$ store:
  - $I_i$ and $w_i$ with
  - $w_i = |I|$ the size of the largest independent set $I$ on $\bigcup_{j=1}^{i} f(p_j)$ with $I_i \subset I$.
- Iteration step on $f(p_{i+1})$:
  - For each subset $I_{i+1}$ compute the above value:
- **No** there is no fun in this ugly task
- **We have to simplify.**
Example (Independent Set)

- Let $G = (V, E)$ be a graph with $\text{pw}(G) = k$ and $|V| = n$.
- Let $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} \mid 1 \leq j \leq n - k\})$ be a path with $n - k + 1$ nodes.
- Let $f : \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto \mathcal{P}(V)$ the embedding function with pathwidth $k$ with:
  - $|f(p_i) \mathbin{\uplus} f(p_{i+1})| = 1$,
  - $f(p_i) \mathbin{\cup} \{x\} = f(p_{i+1})$ or
Example (Independent Set)

- Let $G = (V, E)$ be a graph with $\text{pw}(G) = k$ and $|V| = n$.
- Let $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} \mid 1 \leq j \leq n-k\})$ be a path with $n - k + 1$ nodes.
- Let $f : \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto \mathcal{P}(V)$ the embedding function with pathwidth $k$ with:
  - $|f(p_i) \oplus f(p_{i+1})| = 1$,
    - $f(p_i) \cup \{x\} = f(p_{i+1})$ or
    - $f(p_i) = f(p_{i+1}) \cup \{x\}$
Example (Independent Set)

- Let $G = (V, E)$ be a graph with $\text{pw}(G) = k$ and $|V| = n$.
- Let $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} \mid 1 \leq j \leq n-k\})$ be a path with $n - k + 1$ nodes.
- Let $f : \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto \mathcal{P}(V)$ the embedding function with pathwidth $k$ with:
  - $|f(p_i) \cup f(p_{i+1})| = 1$,
    - $f(p_i) \cup \{x\} = f(p_{i+1})$ or
    - $f(p_i) = f(p_{i+1}) \cup \{x\}$
- Notations:
Example (Independent Set)

- Let $G = (V, E)$ be a graph with $\text{pw}(G) = k$ and $|V| = n$.
- Let $P_{n-k} = \{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} \mid 1 \leq j \leq n - k\}$ be a path with $n - k + 1$ nodes.
- Let $f : \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto \mathcal{P}(V)$ the embedding function with pathwidth $k$ with:
  - $|f(p_i) \oplus f(p_{i+1})| = 1$,
    - $f(p_i) \cup \{x\} = f(p_{i+1})$ or
    - $f(p_i) = f(p_{i+1}) \cup \{x\}$
- Notations:
  - $f(p_{i+1}) = \text{add}(f(p_i), x)$ and
Example (Independent Set)

- Let $G = (V, E)$ be a graph with $\text{pw}(G) = k$ and $|V| = n$.
- Let $P_{n-k} = \{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} | 1 \leq j \leq n-k\}$ be a path with $n-k+1$ nodes.
- Let $f : \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto \mathcal{P}(V)$ the embedding function with pathwidth $k$ with:
  - $|f(p_i) \oplus f(p_{i+1})| = 1$,
    - $f(p_i) \cup \{x\} = f(p_{i+1})$ or
    - $f(p_i) = f(p_{i+1}) \cup \{x\}$
- Notations:
  - $f(p_{i+1}) = \text{add}(f(p_i), x)$ and
  - $f(p_{i+1}) = \text{del}(f(p_i), x)$
Example (Independent Set)

Thus we only have to define the following:

- What we store for $f(p_i)$: $D(f(p_i))$
Example (Independent Set)

Thus we only have to define the following:

- What we store for $f(p_i)$: $D(f(p_i))$
- The procedure $D(f(p_1)) := \text{Init}(f(p_1))$, to compute the initial values
Example (Independent Set)

Thus we only have to define the following:

- What we store for $f(p_i)$: $D(f(p_i))$
- The procedure $D(f(p_1)) := \text{Init}(f(p_1))$, to compute the initial values
- The procedure $D(f(p_{i+1})) := \text{Add}(D(f(p_i)), x)$, to compute the values for $f(p_{i+1})$. 
Thus we only have to define the following:

- What we store for $f(p_i)$: $D(f(p_i))$
- The procedure $D(f(p_1)) := Init(f(p_1))$, to compute the initial values
- The procedure $D(f(p_{i+1})) := Add(D(f(p_i)), x)$, to compute the values for $f(p_{i+1})$.
- The procedure $D(f(p_{i+1})) := Del(D(f(p_i)), x)$, to compute the values for $f(p_{i+1})$. 
Example (Independent Set)

\[ D(f(p_i)) = \{(I, w) \mid I \text{ is IS on } f(p_i) \land w = \text{Wert}(I, i)\} \text{ with:} \]

\[ \text{Wert}(I, i) = \max\{|I'| \mid I' \subset \bigcup_{j=1}^{i} f(p_j) \land I' \text{ is IS} \land I \subset I'| \]
Example (Independent Set)

\[ D(f(p_i)) = \{(l, w) \mid l \text{ is IS on } f(p_i) \land w = \text{Wert}(l, i)\} \text{ with:} \]
\[ \text{Wert}(l, i) = \max\{|l'| \mid l' \subset \bigcup_{j=1}^{i} f(p_j) \land l' \text{ ist IS} \land l \subset l'\} \]

\[ \text{Init}(f(p_1)) = \{(l, w) \mid l \text{ is IS on } f(p_1) \land w = |l|\}, \]
compute all IS \( l \subset f(p_1) \) and set \( w = |l| \).
Example (Independent Set)

- \( D(f(p_i)) = \{(I, w) \mid I \text{ is IS on } f(p_i) \land w = \text{Wert}(I, i)\} \) with:
  \[ \text{Wert}(I, i) = \max\{|I'| \mid I' \subseteq \bigcup_{j=1}^{i} f(p_j) \land I' \text{ is IS} \land I \subseteq I' \} \]

- \( \text{Init}(f(p_1)) = \{(I, w) \mid I \text{ is IS on } f(p_1) \land w = |I|\}, \) compute all IS \( I \subseteq f(p_1) \) and set \( w = |I| \).

- \( (I, w) \in \text{Del}(D(f(p_i)), x) \) iff:
Example (Independent Set)

\[ D(f(p_i)) = \{(I, w) \mid I \text{ is IS on } f(p_i) \land w = \text{Wert}(I, i)\} \text{ with:} \]
\[ \text{Wert}(I, i) = \max\{|I'| \mid I' \subset \bigcup_{j=1}^{i} f(p_j) \land I' \text{ is IS } \land I \subset I'\} \]

\[ \text{Init}(f(p_1)) = \{(I, w) \mid I \text{ is IS on } f(p_1) \land w = |I|\}, \]
compute all IS \( I \subset f(p_1) \) and set \( w = |I| \).

\[ (I, w) \in \text{Del}(D(f(p_i)), x) \text{ iff:} \]
\[ (I \cup \{x\}, w') \in D(f(p_i)) \text{ or } (I, w'') \in D(f(p_i)) \text{ and} \]
Example (Independent Set)

- $D(f(p_i)) = \{(I, w) \mid I \text{ is IS on } f(p_i) \land w = \text{Wert}(I, i)\}$ with:
  \[
  \text{Wert}(I, i) = \max\{|I'| \mid I' \subseteq \bigcup_{j=1}^{i} f(p_j) \land I' \text{ is IS} \land I \subseteq I'\}
  \]

- $Init(f(p_1)) = \{(I, w) \mid I \text{ is IS on } f(p_1) \land w = |I|\}$,
  compute all IS $I \subseteq f(p_1)$ and set $w = |I|$.

- $(I, w) \in \text{Del}(D(f(p_i)), x)$ iff:
  - $(I \cup \{x\}, w') \in D(f(p_i))$ or $(I, w'') \in D(f(p_i))$ and
  - $w = \max\{w' \mid (I \cup \{x\}, w') \in D(f(p_i)) \lor (I, w') \in D(f(p_i))\}$. 
Example (Independent Set)

- $D(f(p_i)) = \{(l, w) \mid l \text{ is IS on } f(p_i) \land w = \text{Wert}(l, i)\}$ with:
  \[
  \text{Wert}(l, i) = \max\{|l'| \mid l' \subset \bigcup_{j=1}^{i} f(p_j) \land l' \text{ is IS} \land l \subset l'\}
  \]

- $\text{Init}(f(p_1)) = \{(l, w) \mid l \text{ is IS on } f(p_1) \land w = |l|\}$, compute all IS $l \subset f(p_1)$ and set $w = |l|$.

- $(l, w) \in \text{Del}(D(f(p_i)), x)$ iff:
  - $(l \cup \{x\}, w') \in D(f(p_i))$ or $(l, w'') \in D(f(p_i))$ and
  - $w = \max\{w' \mid (l \cup \{x\}, w') \in D(f(p_i)) \lor (l, w') \in D(f(p_i))\}$.

- $(l, w) \in \text{Add}(D(f(p_i)), x)$ iff:
Example (Independent Set)

- \( D(f(p_i)) = \{(l, w) \mid l \text{ is IS on } f(p_i) \land w = \text{Wert}(l, i)\} \) with:
  \[ \text{Wert}(l, i) = \max\{|l'| \mid l' \subset \bigcup_{j=1}^{i} f(p_j) \land l' \text{ ist IS } \land l \subset l'\} \]

- \( \text{Init}(f(p_1)) = \{(l, w) \mid l \text{ is IS on } f(p_1) \land w = |l|\}, \) compute all IS \( l \subset f(p_1) \) and set \( w = |l|. \)

- \((l, w) \in \text{Del}(D(f(p_i)), x)\) iff:
  - \((l \cup \{x\}, w') \in D(f(p_i))\) or \((l, w'') \in D(f(p_i))\) and
  - \(w = \max\{w' \mid (l \cup \{x\}, w') \in D(f(p_i))\) or \((l, w') \in D(f(p_i))\}\).

- \((l, w) \in \text{Add}(D(f(p_i)), x)\) iff:
  - \((l, w) \in D(f(p_i))\) or
**Example (Independent Set)**

- $D(f(p_i)) = \{(l, w) \mid l \text{ is IS on } f(p_i) \land w = \text{Wert}(l, i)\}$ with:
  
  $$\text{Wert}(l, i) = \max \{|l'| \mid l' \subset \bigcup_{j=1}^{i} f(p_j) \land l' \text{ ist IS} \land l \subset l'\}$$

- $\text{Init}(f(p_1)) = \{(l, w) \mid l \text{ is IS on } f(p_1) \land w = |l|\}$,
  compute all IS $l \subset f(p_1)$ and set $w = |l|$.

- $(l, w) \in \text{Del}(D(f(p_i)), x)$ iff:
  
  - $(l \cup \{x\}, w') \in D(f(p_i))$ or $(l, w'') \in D(f(p_i))$ and
  
  - $w = \max\{w' \mid (l \cup \{x\}, w') \in D(f(p_i)) \lor (l, w') \in D(f(p_i))\}$.

- $(l, w) \in \text{Add}(D(f(p_i)), x)$ iff:
  
  - $(l, w) \in D(f(p_i))$ or
  
  - $(l \setminus \{x\}, w - 1) \in D(f(p_i))$ and $l$ is IS.
A graph $G = (V, E)$ has treewidth $k$, iff there is a tree $T = (V_T, E_T)$ and a mapping $f : V_T \to \mathcal{P}(V)$ with:

- $\forall (v, u) \in E: \exists x \in V_T: v, u \in f(x)$
**Definition**

A graph $G = (V, E)$ has treewidth $k$, iff there is a tree $T = (V_T, E_T)$ and a mapping $f : V_T \rightarrow \mathcal{P}(V)$ with:

- $\forall (v, u) \in E : \exists x \in V_T : v, u \in f(x)$
- If $c$ is on the path from $a$ to $b$ on $T$, then does $f(b) \cap f(a) \subseteq f(c)$ hold.
A graph $G = (V, E)$ has treewidth $k$, iff there is a tree $T = (V_T, E_T)$ and a mapping $f : V_T \rightarrow \mathcal{P}(V)$ with:

- $\forall (v, u) \in E : \exists x \in V_T : v, u \in f(x)$
- If $c$ is on the path from $a$ to $b$ on $T$, then does $f(b) \cap f(a) \subset f(c)$ hold.
- $\forall x \in V_T : |f(x)| \leq k + 1.$
**Treewidth**

**Definition**

A graph $G = (V, E)$ has treewidth $k$, iff there is a tree $T = (V_T, E_T)$ and a mapping $f : V_T \rightarrow \mathcal{P}(V)$ with:

- $\forall (v, u) \in E : \exists x \in V_T : v, u \in f(x)$
- If $c$ is on the path from $a$ to $b$ on $T$, then does $f(b) \cap f(a) \subseteq f(c)$ hold.
- $\forall x \in V_T : |f(x)| \leq k + 1$. 

Notation: $pw(G) = k$.

Note: $T, f$ is called tree decomposition of width $k$. 
Treewidth

**Definition**

A graph $G = (V, E)$ has treewidth $k$, iff there is a tree $T = (V_T, E_T)$ and a mapping $f : V_T \rightarrow \mathcal{P}(V)$ with:

- $\forall (v, u) \in E : \exists x \in V_T : v, u \in f(x)$
- If $c$ is on the path from $a$ to $b$ on $T$, then does $f(b) \cap f(a) \subseteq f(c)$ hold.
- $\forall x \in V_T : |f(x)| \leq k + 1$.

and for $k - 1$ exists no such function $f$ and tree $T$.

Notation: $\text{pw}(G) = k$.

Note: $T, f$ is called tree decomposition of width $k$. 
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)

\[ \Sigma = \textcolor{red}{0} \]

\[ \text{Pathwidth} \]

\[ \text{Treewidth} \]

\[ \text{k-Trees} \]

\[ \text{Applications} \]

Walter Unger 31.5.2016 14:35 SS2016
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)

Σ = 0
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Theorem

Let $G = (V, E)$ be a graph. Then the following hold:

- The problem, to compute the treewidth of a graph, is NP-complete.
Let $G = (V, E)$ be a graph. Then the following hold:

- The problem, to compute the treewidth of a graph, is NP-complete.
- For a fixed $k$ it is possible to check in linear time $O(n + m)$, if a graph has treewidth $k$. 
Let $G = (V, E)$ be a graph. Then the following hold:

- The problem, to compute the treewidth of a graph, is NP-complete.
- For a fixed $k$ it is possible to check in linear time $O(n + m)$, if a graph has treewidth $k$. 
Theorem

Let $G = (V, E)$ be a graph. Then the following hold:

- The problem, to compute the treewidth of a graph, is NP-complete.
- For a fixed $k$ it is possible to check in linear time $O(n + m)$, if a graph has treewidth $k$.

Theorem

Let $G = (V, E)$ be a graph with $tw(G) = k$. The following problem may be solved in linear time:

- Independent-Set, Clique, Vertex-Cover, $k$-Dominating Set,
- Colouring-problem, Edge-Colouring,
- Hamilton-Cycle, Hamilton-Path,
- Graph-Isomorphism, Is-A-Disk-Graph-Problem,
Simplifications

Let $t$ be a successor of $s$ in the tree. Then we may assume w.l.o.g.:

- $s$ has at most two successors
Simplifications

Let $t$ be a successor of $s$ in the tree. Then we may assume w.l.o.g.:

- $s$ has at most two successors
- $f(t) = f(s)$ holds if there is a second successor of $s$. 
Let $t$ be a successor of $s$ in the tree. Then we may assume w.l.o.g.:

- $s$ has at most two successors
- $f(t) = f(s)$ holds if there is a second successor of $s$.
- $|f(t) \oplus f(s)| = 1$ if there is no second successor of $s$: 

Simplifications

Let \( t \) be a successor of \( s \) in the tree. Then we may assume w.l.o.g.:

- \( s \) has at most two successors
- \( f(t) = f(s) \) holds if there is a second successor of \( s \).
- \(|f(t) \oplus f(s)| = 1\) if there is no second successor of \( s \):
  - \( f(s) = \text{add}(f(t), x) \) and
Let $t$ be a successor of $s$ in the tree. Then we may assume w.l.o.g.:

- $s$ has at most two successors
- $f(t) = f(s)$ holds if there is a second successor of $s$.
- $|f(t) \oplus f(s)| = 1$ if there is no second successor of $s$:
  - $f(s) = \text{add}(f(t), x)$ and
  - $f(s) = \text{del}(f(t), x)$
Example (Independent Set)

Thus we only have to define the following:

- What we store for $f(p_i)$: $D(f(p_i))$. 
Example (Independent Set)

Thus we only have to define the following:

- What we store for $f(p_i)$: $D(f(p_i))$.
- The procedure $D(f(p_1)) := \text{Init}(f(p_1))$, to compute the initial values.
Example (Independet Set)

Thus we only have to define the following:

- What we store for $f(p_i)$: $D(f(p_i))$.
- The procedure $D(f(p_1)) := \text{Init}(f(p_1))$, to compute the initial values.
- The procedure $D(f(p_{i+1})) := \text{Add}(D(f(p_i)), x)$, to compute the values for $f(p_{i+1})$. 
Thus we only have to define the following:

- What we store for $f(p_i)$: $D(f(p_i))$.
- The procedure $D(f(p_1)) := \text{Init}(f(p_1))$, to compute the initial values.
- The procedure $D(f(p_{i+1})) := \text{Add}(D(f(p_i)), x)$, to compute the values for $f(p_{i+1})$.
- The procedure $D(f(p_{i+1})) := \text{Del}(D(f(p_i)), x)$, to compute the values for $f(p_{i+1})$. 
Example (Independent Set)

Thus we only have to define the following:

- What we store for $f(p_i)$: $D(f(p_i))$.
- The procedure $D(f(p_1)) := \text{Init}(f(p_1))$, to compute the initial values.
- The procedure $D(f(p_{i+1})) := \text{Add}(D(f(p_i)), x)$, to compute the values for $f(p_{i+1})$.
- The procedure $D(f(p_{i+1})) := \text{Del}(D(f(p_i)), x)$, to compute the values for $f(p_{i+1})$.
- The procedure $D(f(s)) := \text{Join}(D(f(t)), D(f(t')))$. 
Example (Independent Set)

- \( D(f(s)) = \{(l, w) \mid l \text{ is IS on } f(s) \wedge w = \text{value}(l, s)\} \) with:
  - \( \text{value}(l, s) = \max\{|l'| \mid l' \subset \bigcup_{t \in V(T_s)} f(t) \wedge l' \text{ is IS } \wedge l \subset l'\} \) and
  - \( T_s \) is the subtree with root \( s \).

- \( \text{Init}(f(t)) = \{(l, w) \mid l \text{ is IS on } f(t) \wedge w = |l|\} \),
  compute all IS \( l \subset f(t) \) and set \( w = |l| \).

- \((l, w) \in \text{Del}(D(f(t)), x)\) iff:
  - \((l \cup \{x\}, w') \in D(f(p_i)) \) or \((l, w'') \in D(f(p_i))\) and
  - \( w = \max\{w' \mid (l \cup \{x\}, w') \in D(f(p_i)) \) or \((l, w') \in D(f(p_i))\}\).

- \((l, w) \in \text{Add}(D(f(t)), x)\) iff:
  - \((l, w) \in D(f(t))\) or
  - \((l \setminus \{x\}, w - 1) \in D(f(t)) \) and \( l \) is IS.

- \((l, w) \in \text{Join}(D(f(t)), D(f(t')))\) iff:
  - \((l, w') \in D(f(t))\) and
  - \((l, w'') \in D(f(t'))\) and
  - \( w = w' + w'' - |l| \).
Vertex Cover and Treewidth

Definition (Vertex Cover)

Let $G = (V, E)$ be a graph. The size of the minimal vertex cover is:

$$vc(G) = \min_{C \subseteq V: \forall e \in E: e \cap C \neq \emptyset} |C|$$
Vertex Cover and Treewidth

**Definition (Vertex Cover)**

Let $G = (V, E)$ be a graph. The size of the minimal vertex cover is:

$$vc(G) = \min_{C \subseteq V : \forall e \in E : e \cap C \neq \emptyset} |C|$$

**Theorem**

Let $G = (V, E)$ be a graph. Then $tw(G) \leq vc(G)$ holds.
Vertex Cover and Treewidth

**Definition (Vertex Cover)**

Let $G = (V, E)$ be a graph. The size of the minimal vertex cover is:

$$vc(G) = \min_{C \subseteq V : \forall e \in E : e \cap C \neq \emptyset} |C|$$

**Theorem**

Let $G = (V, E)$ be a graph. Then $tw(G) \leq vc(G)$ holds.

**Theorem**

Let $G = (V, E)$ be a graph. Then $pw(G) \leq vc(G)$ hold.
Vertex Cover and Treewidth

Theorem

Let $G = (V, E)$ be a graph. Then $\text{pw}(G) \leq \text{vc}(G)$ hold.

Proof:

- Let $C \subseteq V$ with: $\forall e \in E : e \cap C \neq \emptyset$ and $|C| = k = \text{vc}(G)$. 
Vertex Cover and Treewidth

**Theorem**

Let $G = (V, E)$ be a graph. Then $pw(G) \leq vc(G)$ hold.

**Proof:**

- Let $C \subset V$ with: $\forall e \in E : e \cap C \neq \emptyset$ and $|C| = k = vc(G)$.
- Let w.l.o.g. $C = \{v_1, v_2, \ldots, v_k\}$ and $V = \{v_1, v_2, \ldots, v_n\}$. 

Vertex Cover and Treewidth

Theorem

Let $G = (V, E)$ be a graph. Then $pw(G) \leq vc(G)$ hold.

Proof:

- Let $C \subset V$ with: $\forall e \in E : e \cap C \neq \emptyset$ and $|C| = k = vc(G)$.
- Let w.l.o.g. $C = \{v_1, v_2, \ldots, v_k\}$ and $V = \{v_1, v_2, \ldots, v_n\}$.
- Furthermore let $P = (\{p_{k+1}, p_{k+2}, \ldots, p_n\}, \{\{p_j, p_{j+1}\} \mid k + 1 \leq j < n\})$ be a path with $n - k$ nodes.
Vertex Cover and Treewidth

Theorem

Let \(G = (V, E)\) be a graph. Then \(\text{pw}(G) \leq \text{vc}(G)\) hold.

Proof:

- Let \(C \subseteq V\) with: \(\forall e \in E : e \cap C \neq \emptyset\) and \(|C| = k = \text{vc}(G)\).
- Let w.l.o.g. \(C = \{v_1, v_2, \ldots, v_k\}\) and \(V = \{v_1, v_2, \ldots, v_n\}\).
- Furthermore let \(P = (\{p_{k+1}, p_{k+2}, \ldots, p_n\}, \{\{p_j, p_{j+1}\} \mid k + 1 \leq j < n\})\) be a path with \(n - k\) nodes.
- Define \(f(p_j) = C \cup \{v_j\}\) for \(k + 1 \leq j \leq n\).
Vertex Cover and Treewidth

**Theorem**

Let $G = (V, E)$ be a graph. Then $pw(G) \leq vc(G)$ hold.

**Proof:**

- Let $C \subseteq V$ with: $\forall e \in E : e \cap C \neq \emptyset$ and $|C| = k = vc(G)$.
- Let w.l.o.g. $C = \{v_1, v_2, \ldots, v_k\}$ and $V = \{v_1, v_2, \ldots, v_n\}$.
- Furthermore let $P = (\{p_{k+1}, p_{k+2}, \ldots, p_n\}, \{\{p_j, p_{j+1}\} \mid k + 1 \leq j < n\})$ be a path with $n - k$ nodes.
- Define $f(p_j) = C \cup \{v_j\}$ for $k + 1 \leq j \leq n$.
- Then we have:
Vertex Cover and Treewidth

**Theorem**

Let $G = (V, E)$ be a graph. Then $pw(G) \leq vc(G)$ holds.

**Proof:**

- Let $C \subset V$ with: $\forall e \in E : e \cap C \neq \emptyset$ and $|C| = k = vc(G)$.
- Let w.l.o.g. $C = \{v_1, v_2, \ldots, v_k\}$ and $V = \{v_1, v_2, \ldots, v_n\}$.
- Furthermore let $P = (\{p_{k+1}, p_{k+2}, \ldots, p_n\}, \{\{p_j, p_{j+1}\} \mid k + 1 \leq j < n\})$ be a path with $n - k$ nodes.
- Define $f(p_j) = C \cup \{v_j\}$ for $k + 1 \leq j \leq n$.
- Then we have:
  - $|f(p_j)| \leq vc(G) + 1$ for $k + 1 \leq j \leq n$. 
Vertex Cover and Treewidth

**Theorem**

Let $G = (V, E)$ be a graph. Then $\text{pw}(G) \leq \text{vc}(G)$ hold.

**Proof:**

- Let $C \subset V$ with: $\forall e \in E : e \cap C \neq \emptyset$ and $|C| = k = \text{vc}(G)$.
- Let w.l.o.g. $C = \{v_1, v_2, \ldots, v_k\}$ and $V = \{v_1, v_2, \ldots, v_n\}$.
- Furthermore let $P = (\{p_{k+1}, p_{k+2}, \ldots, p_n\}, \{\{p_j, p_{j+1}\} | k + 1 \leq j < n\})$ be a path with $n - k$ nodes.
- Define $f(p_j) = C \cup \{v_j\}$ for $k + 1 \leq j \leq n$.
- Then we have:
  - $|f(p_j)| \leq \text{vc}(G) + 1$ for $k + 1 \leq j \leq n$.
  - $\{v_c, v_j\} \in E$ and $\{v_c, v_j\} \subset C$, then we get $\{v_c, v_j\} \subset f(p_n)$. 


Vertex Cover and Treewidth

**Theorem**

Let $G = (V, E)$ be a graph. Then $pw(G) \leq vc(G)$ hold.

**Proof:**

- Let $C \subset V$ with: $\forall e \in E : e \cap C \neq \emptyset$ and $|C| = k = vc(G)$.
- Let w.l.o.g. $C = \{v_1, v_2, \ldots, v_k\}$ and $V = \{v_1, v_2, \ldots, v_n\}$.
- Furthermore let $P = (\{p_{k+1}, p_{k+2}, \ldots, p_n\}, \\{\{p_j, p_{j+1}\} | k + 1 \leq j < n\})$ be a path with $n - k$ nodes.
- Define $f(p_j) = C \cup \{v_j\}$ for $k + 1 \leq j \leq n$.
- Then we have:
  - $|f(p_j)| \leq vc(G) + 1$ for $k + 1 \leq j \leq n$.
  - $\{v_c, v_j\} \in E$ and $\{v_c, v_j\} \subset C$, then we get $\{v_c, v_j\} \subset f(p_n)$.
  - $\{v_c, v_j\} \in E$ and $\{v_c, v_j\} \cap C = v_c$, then we get $\{v_c, v_j\} \subset f(p_j)$. 
Vertex Cover and Treewidth

Theorem

Let $G = (V, E)$ be a graph. Then $\text{pw}(G) \leq \text{vc}(G)$ hold.

Proof:

- Let $C \subset V$ with: $\forall e \in E : e \cap C \neq \emptyset$ and $|C| = k = \text{vc}(G)$.
- Let w.l.o.g. $C = \{v_1, v_2, \ldots, v_k\}$ and $V = \{v_1, v_2, \ldots, v_n\}$.
- Furthermore let $P = (\{p_{k+1}, p_{k+2}, \ldots, p_n\}, \{\{p_j, p_{j+1}\} | k + 1 \leq j < n\})$ be a path with $n - k$ nodes.
- Define $f(p_j) = C \cup \{v_j\}$ for $k + 1 \leq j \leq n$.
- Then we have:
  - $|f(p_j)| \leq \text{vc}(G) + 1$ for $k + 1 \leq j \leq n$.
  - $\{v_c, v_j\} \in E$ and $\{v_c, v_j\} \subset C$, then we get $\{v_c, v_j\} \subset f(p_n)$.
  - $\{v_c, v_j\} \in E$ and $\{v_c, v_j\} \cap C = v_c$, then we get $\{v_c, v_j\} \subset f(p_j)$.
- Thus $\text{pw}(G) \leq \text{vc}(G)$ holds.
Idea (1-Tree)
Idea (1-Tree)
Idea (1-Tree)
Idea (1-Tree)
Idea (1-Tree)
Idea (1-Tree)
Idea (1-Tree)
Idea (1-Tree)

\[ \Sigma = 0 \]

\[ \Sigma = 0 \]
Idea (1-Tree)
Idea (1-Tree)
Idea (1-Tree)
Idea (1-Tree)
Idea (1-Tree)
Idea (2-Tree)
Idea (2-Tree)

\[ v_1, v_2, v_3 \]
Idea (2-Tree)
Idea (2-Tree)

\[ \Sigma = v_1v_2v_3 \]

\[ \Sigma = v_1v_2v_4 \]

\[ \Sigma = v_1v_3v_5 \]
Idea (2-Tree)
Idea (2-Tree)
Idea (2-Tree)
Idea (2-Tree)
Idea (2-Tree)
Idea (2-Tree)

\[ \Sigma = 0 \]
Idea (2-Tree)
Idea (2-Tree)
Idea (3-Tree)
Idea (3-Tree)
Idea (3-Tree)
Idea (3-Tree)

\[ \Sigma = v_1v_2v_3v_4 \]

\[ \Sigma = v_1v_2v_3v_5 \]

\[ \Sigma = v_1v_2v_5v_6 \]
Idea (3-Tree)
Idea (3-Tree)

\[ \Sigma = \emptyset \]
Idea (3-Tree)
Idea (3-Tree)
Idea (3-Tree)
Idea (3-Tree)
Idea (3-Tree)
**Definition (k-tree (Rose 1974))**

A *k*-tree is as follows recursively defined:

- $K_{k+1}$ is a *k*-tree.
- Note: One may also start with $K_k$.
- If $T = (V, E)$ is a *k*-tree and $C = \{c_1, c_2, \ldots, c_k\}$ is a clique in $T$, then is $T = (V \cup \{v\}, E \cup \{(v, c_i); 1 \leq i \leq k\})$ also a *k*-tree.
- There are no further $k$-trees.

Let $T = (V, E)$ be a *k*-tree. Then is $G = (V, F)$ with $F \subset E$ called a partial $k$-tree.
Theorems I

- A 1-tree is a tree.
- Let $G$ be a $k$-tree. Then $\omega(G) = k + 1$ holds if $G$ has more than $k$ nodes (otherwise $\omega(G) = k$).
- $\omega(G) = \max\{|C| \mid C \subset V(G) \land C \text{ ist Clique}\}$
Theorems I

**Theorem**

- A 1-tree is a tree.
- Let $G$ be a $k$-tree. Then $\omega(G) = k + 1$ holds if $G$ has more than $k$ nodes (otherwise $\omega(G) = k$).
- $\omega(G) = \max\{|C| \mid C \subset V(G) \land C \text{ ist Clique}\}$

**Lemma**

A $k$-tree could be constructed by starting from any clique.
Theorems I

**Theorem**

- A 1-tree is a tree.
- Let $G$ be a $k$-tree. Then $\omega(G) = k + 1$ holds if $G$ has more than $k$ nodes (otherwise $\omega(G) = k$).
- $\omega(G) = \max\{|C| \mid C \subseteq V(G) \land C \text{ ist Clique}\}$

**Lemma**

A $k$-tree could be constructed by starting from any clique.

**Note**

A $k$-tree is chordal and perfect.
Theorem

A graph \( G = (V, E) \) is a \( k \)-tree, iff \( tw(G) = k \) and \( G \) is maximal.
Theorem

A graph $G = (V, E)$ is a $k$-tree, iff $\text{tw}(G) = k$ and $G$ is maximal.

Theorem

A graph $G = (V, E)$ is a partial $k$-tree, iff $\text{tw}(G) \leq k$. 
Finding the Treewidth of a Graph

- It is hard to find the tree.
Finding the Treewidth of a Graph

- It is hard to find the tree.
- One may use the following model:
Finding the Treewidth of a Graph

- It is hard to find the tree.
- One may use the following model:
- **Modify the search-number.**
Finding the Treewidth of a Graph

- It is hard to find the tree.
- One may use the following model:
  - Modify the search-number.
- How many policeman are needed to find a person in a graph.
Finding the Treewidth of a Graph

- It is hard to find the tree.
- One may use the following model:
  - Modify the search-number.
- How many policeman are needed to find a person in a graph.
- The person may be arbitrary fast and may use nodes and edges.
Finding the Treewidth of a Graph

- It is hard to find the tree.
- One may use the following model:
  - Modify the search-number.
- How many policeman are needed to find a person in a graph.
- The person may be arbitrary fast and may use nodes and edges.
- **Policeman are only allowed to use the nodes, but may jump.**
Finding the Treewidth of a Graph

- It is hard to find the tree.
- One may use the following model:
  - Modify the search-number.
- How many policeman are needed to find a person in a graph.
- The person may be arbitrary fast and may use nodes and edges.
- Policeman are only allowed to use the nodes, but may jump.
- The person may not pass a node where a policeman is.
Finding the Treewidth of a Graph

- It is hard to find the tree.
- One may use the following model:
  - Modify the search-number.
  - How many policeman are needed to find a person in a graph.
  - The person may be arbitrary fast and may use nodes and edges.
  - Policeman are only allowed to use the nodes, but may jump.
  - The person may not pass a node where a policeman is.
- The policeman know the position of the person.
Finding the Treewidth of a Graph

- It is hard to find the tree.
- One may use the following model:
  - Modify the search-number.
  - How many policeman are needed to find a person in a graph.
  - The person may be arbitrary fast and may use nodes and edges.
  - Policeman are only allowed to use the nodes, but may jump.
  - The person may not pass a node where a policeman is.
  - The policeman know the position of the person.
- An edge is called free (searched) if there is a policeman on both the incident nodes.
Finding the Treewidth of a Graph

- It is hard to find the tree.
- One may use the following model:
- Modify the search-number.
- How many policeman are needed to find a person in a graph.
- The person may be arbitrary fast and may use nodes and edges.
- Policeman are only allowed to use the nodes, but may jump.
- The person may not pass a node where a policeman is.
- The policeman know the position of the person.
- An edge is called free (searched) if there is a policeman on both the incident nodes.
- Modified search-number corresponds to treewidth of a graph.
Definition
A graph $G = (V, E)$ is called cactus, iff each 2-connected component is a cycle.

Theorem
For a cactus $G = (V, E)$ holds: $\text{tw}(G) \leq 2$
Theorems III

Definition
A graph \( G = (V, E) \) is called cactus, iff each 2-connected component is a cycle.

Theorem
For a cactus \( G = (V, E) \) holds: \( \text{tw}(G) \leq 2 \)

Definition
A graph \( G = (V, E) \) is called near-tree\( (k) \), iff each 2-connected component with \( x \) nodes has at most \( x + k - 1 \) edges.

Theorem
For a near-tree\( (k) \) \( G = (V, E) \) holds: \( \text{tw}(G) \leq k + 1 \)
Idea Cactus
Idea Cactus

\[ \Sigma = \]

\[ \begin{align*}
\text{a}_0 & \quad \text{c}_0 & \quad \text{e}_0 \\
\text{a}_1 & \quad \text{c}_1 & \quad \text{e}_1 \\
\text{r}_1 & \quad \text{r}_2
\end{align*} \]
Idea Cactus
Idea Cactus
Idea Cactus

\[ \Sigma = \]
Idea Cactus
Idea Cactus

\[ \Sigma = \]

\[
\begin{array}{c}
a_0 \\
c_0 \\
e_0 \\
\end{array}
\]

\[
\begin{array}{c}
a_1 \\
c_1 \\
e_1 \\
\end{array}
\]

\[
\begin{array}{c}
r_1 \\
r_2 \\
\end{array}
\]

\[
\begin{array}{c}
a_0 a_1 c_0 c_1 e_0 e_1 r_1 c_2 e_3 \\
\end{array}
\]

\[
\begin{array}{c}
a_1 c_2 \\
a_1 c_1 c_2 e_1 e_0 e_1 c_0 c_1 e_3 r_2 r_1 c_2 \\
\end{array}
\]

\[
\begin{array}{c}
e_3 \\
c_2 e_3 r_2 e_3 \\
c_2 \\
e_1 c_2 r_1 c_2 \\
\end{array}
\]

\[
\begin{array}{c}
a_0 a_1 a_0 a_1 c_0 c_1 c_0 c_1 e_0 e_0 e_1 e_0 e_1 e_3 r_2 r_1 c_2 \\
\end{array}
\]
Idea Cactus
Idea Cactus
Idea Cactus
Idea Cactus
Idea: Cactus

\[ \Sigma = \]

Diagram of a Cactus graph and a tree-like structure.
Idea Cactus
Idea Cactus

Diagram of a cactus graph with nodes labeled as follows:
- **c2**: Central node
- **e3**: Edge node
- **r2**: Root node
- **e1**, **c1**, **a1**, **e0**, **c0**, **a0**: Various nodes

Diagram showing the connectivity and structure of the cactus graph.
Proof (Cactus)

- Let $G = (V, E)$ be a cactus with $V = \{v_1, v_2, v_3, \ldots, v_n\}$. 


Proof (Cactus)

- Let $G = (V, E)$ be a cactus with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all cycles in $G$. 

\[ \text{Proof (Cactus)} \]

\[ \text{Let } G = (V, E) \text{ be a cactus with } V = \{v_1, v_2, v_3, \ldots, v_n\}. \]

\[ \text{Let } C_1, C_2, \ldots, C_d \text{ be all cycles in } G. \]
Proof (Cactus)

- Let $G = (V, E)$ be a cactus with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all cycles in $G$.
- Delete from each cycle $C_i$ one edge $e_i$. 
Proof (Cactus)

- Let $G = (V, E)$ be a cactus with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all cycles in $G$.
- Delete from each cycle $C_i$ one edge $e_i$.
- Then $T = (V, E \setminus \{e_1, e_2, \ldots, e_d\})$ is a tree with root $v_1$. 

Proof (Cactus)

- Let $G = (V, E)$ be a cactus with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all cycles in $G$.
- Delete from each cycle $C_i$ one edge $e_i$.
- Then $T = (V, E \setminus \{e_1, e_2, \ldots, e_d\})$ is a tree with root $v_1$.
- Modify now $T$ as follows:
Proof (Cactus)

Let $G = (V, E)$ be a cactus with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.

Let $C_1, C_2, \ldots, C_d$ be all cycles in $G$.

Delete from each cycle $C_i$ one edge $e_i$.

Then $T = (V, E \setminus \{e_1, e_2, \ldots, e_d\})$ is a tree with root $v_1$.

Modify now $T$ as follows:

- For each node $v$ define $f(v) = \{v\}$. 

Proof (Cactus)

- Let $G = (V, E)$ be a cactus with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all cycles in $G$.
- Delete from each cycle $C_i$ one edge $e_i$.
- Then $T = (V, E \setminus \{e_1, e_2, \ldots, e_d\})$ is a tree with root $v_1$.
- Modify now $T$ as follows:
  - For each node $v$ define $f(v) = \{v\}$.
  - For each edge $\{a, b\} \in E(T)$ generate a new node $v_{a,b}$ and define $f(v_{a,b}) = \{a, b\}$. 
Proof (Cactus)

Let $G = (V, E)$ be a cactus with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
Let $C_1, C_2, \ldots, C_d$ be all cycles in $G$.
Delete from each cycle $C_i$ one edge $e_i$.
Then $T = (V, E \setminus \{e_1, e_2, \ldots, e_d\})$ is a tree with root $v_1$.

Modify now $T$ as follows:

- For each node $v$ define $f(v) = \{v\}$.
- For each edge $\{a, b\} \in E(T)$ generate a new node $v_{a,b}$ and define $f(v_{a,b}) = \{a, b\}$.
- Replace each edge $\{a, b\} \in E(T)$ by $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
Proof (Cactus)

- Let $G = (V, E)$ be a cactus with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all cycles in $G$.
- Delete from each cycle $C_i$ one edge $e_i$.
- Then $T = (V, E \setminus \{e_1, e_2, \ldots, e_d\})$ is a tree with root $v_1$.
- Modify now $T$ as follows:
  - For each node $v$ define $f(v) = \{v\}$.
  - For each edge $\{a, b\} \in E(T)$ generate a new node $v_{a,b}$ and define $f(v_{a,b}) = \{a, b\}$.
  - Replace each edge $\{a, b\} \in E(T)$ by $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
  - Replace each node $v$ with $\deg(v) > 3$ by a tree of degree 3.
Proof (Cactus)

- Let $G = (V, E)$ be a cactus with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all cycles in $G$.
- Delete from each cycle $C_i$ one edge $e_i$.
- Then $T = (V, E \setminus \{e_1, e_2, \ldots, e_d\})$ is a tree with root $v_1$.
- Modify now $T$ as follows:
  - For each node $v$ define $f(v) = \{v\}$.
  - For each edge $\{a, b\} \in E(T)$ generate a new node $v_{a,b}$ and define $f(v_{a,b}) = \{a, b\}$.
  - Replace each edge $\{a, b\} \in E(T)$ by $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
  - Replace each node $v$ with $\deg(v) > 3$ by a tree of degree 3.
  - For all nodes $x$ which replace $v$ define $f(x) = \{v\}$. 


Proof (Cactus)

- Let $G = (V, E)$ be a cactus with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all cycles in $G$.
- Delete from each cycle $C_i$ one edge $e_i$.
- Then $T = (V, E \setminus \{e_1, e_2, \ldots, e_d\})$ is a tree with root $v_1$.
- Modify now $T$ as follows:
  - For each node $v$ define $f(v) = \{v\}$.
  - For each edge $\{a, b\} \in E(T)$ generate a new node $v_{a,b}$ and define $f(v_{a,b}) = \{a, b\}$.
  - Replace each edge $\{a, b\} \in E(T)$ by $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
  - Replace each node $v$ with $\deg(v) > 3$ by a tree of degree 3.
  - For all nodes $x$ which replace $v$ define $f(x) = \{v\}$.
  - For each edge $e_i = \{a, b\}$ let $x$ be the smallest common predecessor.
Proof (Cactus)

- Let $G = (V, E)$ be a cactus with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all cycles in $G$.
- Delete from each cycle $C_i$ one edge $e_i$.
- Then $T = (V, E \setminus \{e_1, e_2, \ldots, e_d\})$ is a tree with root $v_1$.
- Modify now $T$ as follows:
  - For each node $v$ define $f(v) = \{v\}$.
  - For each edge $\{a, b\} \in E(T)$ generate a new node $v_{a,b}$ and define $f(v_{a,b}) = \{a, b\}$.
  - Replace each edge $\{a, b\} \in E(T)$ by $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
  - Replace each node $v$ with $\deg(v) > 3$ by a tree of degree 3.
  - For all nodes $x$ which replace $v$ define $f(x) = \{v\}$.
  - For each edge $e_i = \{a, b\}$ let $x$ be the smallest common predecessor.
  - For each node $y$ on the path from $a$ to $x$ define $f(y) = f(y) \cup \{a\}$. 

\[\text{Proof (Cactus)}\]
Proof (Cactus)

- Let $G = (V, E)$ be a cactus with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all cycles in $G$.
- Delete from each cycle $C_i$ one edge $e_i$.
- Then $T = (V, E \setminus \{e_1, e_2, \ldots, e_d\})$ is a tree with root $v_1$.
- Modify now $T$ as follows:
  - For each node $v$ define $f(v) = \{v\}$.
  - For each edge $\{a, b\} \in E(T)$ generate a new node $v_{a,b}$ and define $f(v_{a,b}) = \{a, b\}$.
  - Replace each edge $\{a, b\} \in E(T)$ by $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
  - Replace each node $v$ with $\deg(v) > 3$ by a tree of degree 3.
  - For all nodes $x$ which replace $v$ define $f(x) = \{v\}$.
  - For each edge $e_i = \{a, b\}$ let $x$ be the smallest common predecessor.
  - For each node $y$ on the path from $a$ to $x$ define $f(y) = f(y) \cup \{a\}$.
  - For each node $y$ on the path from $b$ to $x$ define $f(y) = f(y) \cup \{b\}$.
Proof (Cactus)

Let $G = (V, E)$ be a cactus with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.

Let $C_1, C_2, \ldots, C_d$ be all cycles in $G$.

Delete from each cycle $C_i$ one edge $e_i$.

Then $T = (V, E \setminus \{e_1, e_2, \ldots, e_d\})$ is a tree with root $v_1$.

Modify now $T$ as follows:

- For each node $v$ define $f(v) = \{v\}$.
- For each edge $\{a, b\} \in E(T)$ generate a new node $v_{a,b}$ and define $f(v_{a,b}) = \{a, b\}$.
- Replace each edge $\{a, b\} \in E(T)$ by $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
- Replace each node $v$ with $\deg(v) > 3$ by a tree of degree 3.
- For all nodes $x$ which replace $v$ define $f(x) = \{v\}$.
- For each edge $e_i = \{a, b\}$ let $x$ be the smallest common predecessor.
- For each node $y$ on the path from $a$ to $x$ define $f(y) = f(y) \cup \{a\}$.
- For each node $y$ on the path from $b$ to $x$ define $f(y) = f(y) \cup \{b\}$.

We have for each node $z$: $|f(z)| \leq 3$. 
Idea near-tree
Idea near-tree

\[ \Sigma = \]

\begin{align*}
  &a_0 \quad c_0 \quad e_0 \\
  &a_1 \quad c_1 \quad e_1 \\
  &c_2 \quad r_2 \\
\end{align*}
Idea near-tree
Idea near-tree
Idea near-tree

- Diagram showing a near-tree structure with nodes labeled a, c, e, r, and their relationships.
Idea near-tree
Idea near-tree

\[ \Sigma = 0 \]
Idea near-tree
Idea near-tree

\[ \Sigma = \]

\[ \begin{array}{c}
  \text{a0} \\
  \text{c0} \\
  \text{e0} \\
  \text{a1} \\
  \text{c1} \\
  \text{e1} \\
  \text{r1} \\
  \text{c2} \\
  \text{r2} \\
  \text{e3}
\end{array} \]
Idea near-tree
Idea near-tree
Idea near-tree
Idea near-tree
Idea near-tree
Proof (near-tree)

Let $G = (V, E)$ be a near-tree($k$) with $V = \{v_1, v_2, v_3, \ldots, v_n\}$. 
Proof (near-tree)

- Let $G = (V, E)$ be a near-tree($k$) with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all 2-connected components in $G$. 
Proof (near-tree)

- Let $G = (V, E)$ be a near-tree($k$) with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all 2-connected components in $G$.
- Delete from each component $C_i$ up to $d_i \leq k$ edges $e_i^j$. 


Proof (near-tree)

- Let $G = (V, E)$ be a near-tree($k$) with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all 2-connected components in $G$.
- Delete from each component $C_i$ up to $d_i \leq k$ edges $e^i_j$.
- Then is $T = (V, E \setminus \{e^i_j | 1 \leq i \leq d \land 1 \leq j \leq d_i\})$ a tree with root $v_1$. 
Proof (near-tree)

- Let $G = (V, E)$ be a near-tree($k$) with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all 2-connected components in $G$.
- Delete from each component $C_i$ up to $d_i \leq k$ edges $e^i_j$.
- Then is $T = (V, E \setminus \{e^i_j \mid 1 \leq i \leq d \land 1 \leq j \leq d_i\})$ a tree with root $v_1$.
- Modify now $T$ as follows:
Proof (near-tree)

- Let $G = (V, E)$ be a near-tree($k$) with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all 2-connected components in $G$.
- Delete from each component $C_i$ up to $d_i \leq k$ edges $e^i_j$.
- Then is $T = (V, E \setminus \{e^i_j \mid 1 \leq i \leq d \land 1 \leq j \leq d_i\})$ a tree with root $v_1$.
- Modify now $T$ as follows:
  - For each node $v$ define $f(v) = \{v\}$. 
Proof (near-tree)

- Let $G = (V, E)$ be a near-tree($k$) with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all 2-connected components in $G$.
- Delete from each component $C_i$ up to $d_i \leq k$ edges $e_i^j$.
- Then is $T = (V, E \setminus \{e_i^j \mid 1 \leq i \leq d \land 1 \leq j \leq d_i\})$ a tree with root $v_1$.
- Modify now $T$ as follows:
  - For each node $v$ define $f(v) = \{v\}$.
  - Replace any node $v$ which is several 2-connected components by a star.
Proof (near-tree)

- Let $G = (V, E)$ be a near-tree($k$) with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all 2-connected components in $G$.
- Delete from each component $C_i$ up to $d_i \leq k$ edges $e^i_j$.
- Then is $T = (V, E \setminus \{e^i_j | 1 \leq i \leq d \land 1 \leq j \leq d_i\})$ a tree with root $v_1$.
- Modify now $T$ as follows:
  - For each node $v$ define $f(v) = \{v\}$.
  - Replace any node $v$ which is several 2-connected components by a star.
  - For all nodes $x$ which replace $v$ define $f(x) = \{v\}$. 

Proof (near-tree)

- Let $G = (V, E)$ be a near-tree($k$) with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all 2-connected components in $G$.
- Delete from each component $C_i$ up to $d_i \leq k$ edges $e_i^j$.
- Then is $T = (V, E \setminus \{e_i^j \mid 1 \leq i \leq d \land 1 \leq j \leq d_i\})$ a tree with root $v_1$.
- Modify now $T$ as follows:
  - For each node $v$ define $f(v) = \{v\}$.
  - Replace any node $v$ which is several 2-connected components by a star.
  - For all nodes $x$ which replace $v$ define $f(x) = \{v\}$.
  - For each edge $\{a, b\} \in E(T)$ generate a new node $v_{a,b}$ and define $f(v_{a,b}) = \{a, b\}$. 

Proof (near-tree)

Let $G = (V, E)$ be a near-tree($k$) with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
Let $C_1, C_2, \ldots, C_d$ be all 2-connected components in $G$.
Delete from each component $C_i$ up to $d_i \leq k$ edges $e^i_j$.
Then is $T = (V, E \setminus \{e^i_j \mid 1 \leq i \leq d \land 1 \leq j \leq d_i\})$ a tree with root $v_1$.

Modify now $T$ as follows:
- For each node $v$ define $f(v) = \{v\}$.
- Replace any node $v$ which is several 2-connected components by a star.
- For all nodes $x$ which replace $v$ define $f(x) = \{v\}$.
- For each edge $\{a, b\} \in E(T)$ generate a new node $v_{a,b}$ and define $f(v_{a,b}) = \{a, b\}$.
- Replace each edge $\{a, b\} \in E(T)$ by $\{a, v_{a,b}\}, \{v_{a,b}, b\}$. 
Proof (near-tree)

- Let $G = (V, E)$ be a near-tree($k$) with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all 2-connected components in $G$.
- Delete from each component $C_i$ up to $d_i \leq k$ edges $e^i_j$.
- Then is $T = (V, E \setminus \{e^i_j | 1 \leq i \leq d \land 1 \leq j \leq d_i\})$ a tree with root $v_1$.
- Modify now $T$ as follows:
  - For each node $v$ define $f(v) = \{v\}$.
  - Replace any node $v$ which is several 2-connected components by a star.
  - For all nodes $x$ which replace $v$ define $f(x) = \{v\}$.
  - For each edge $\{a, b\} \in E(T)$ generate a new node $v_{a,b}$ and define $f(v_{a,b}) = \{a, b\}$.
  - Replace each edge $\{a, b\} \in E(T)$ by $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
  - For each edge $e^i_j = \{a, b\}$ and for each node $y$ on the path from $a$ to $b$ define $f(y) = f(y) \cup \{a\}$. 
Proof (near-tree)

- Let $G = (V, E)$ be a near-tree($k$) with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $C_1, C_2, \ldots, C_d$ be all 2-connected components in $G$.
- Delete from each component $C_i$ up to $d_i \leq k$ edges $e_i^j$.
- Then is $T = (V, E \setminus \{e_i^j | 1 \leq i \leq d \land 1 \leq j \leq d_i\})$ a tree with root $v_1$.
- Modify now $T$ as follows:
  - For each node $v$ define $f(v) = \{v\}$.
  - Replace any node $v$ which is several 2-connected components by a star.
  - For all nodes $x$ which replace $v$ define $f(x) = \{v\}$.
  - For each edge $\{a, b\} \in E(T)$ generate a new node $v_{a,b}$ and define $f(v_{a,b}) = \{a, b\}$.
  - Replace each edge $\{a, b\} \in E(T)$ by $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
  - For each edge $e_i^j = \{a, b\}$ and for each node $y$ on the path from $a$ to $b$ define $f(y) = f(y) \cup \{a\}$.
- We have for each node $z$: $|f(z)| \leq 2 + k$. 
Theorems IV

Definition
A graph $G = (V, E)$ is called Halin-graph, iff $G$ is a planar embedded tree where the leaves are connected by the cycle.

Theorem
For a Halin-Graph $G = (V, E)$ holds: $\text{tw}(G) \leq 3$
Theorems IV

**Definition**

A graph $G = (V, E)$ is called Halin-graph, iff $G$ is a planar embedded tree where the leave are connected by the cycle.

**Theorem**

For a Halin-Graph $G = (V, E)$ holds: $\text{tw}(G) \leq 3$

**Definition**

A planar graph $G = (V, E)$ is called outer-planar, iff it could be drawn in the plane, such that no two edges cross and all nodes are on the outer window.

**Theorem**

For a outer-planar graph $G = (V, E)$ holds: $\text{tw}(G) \leq 2$
Idea Halin.
Idea Halin
Idea Halin
Idea Halin
Idea Halin

\[ \Sigma = a_0 c_0 e_0 a_1 c_1 e_1 r_2 c_2 e_3 \]
Idea Halin
Idea Halin
Idea Halin
Idea Halin

- 

\[ \Sigma = 0 \]

- 

\[ \Sigma = 0 \]
Idea Halin

\[
\text{Σ} = \begin{array}{cccc}
a_0 & c_0 & e_0 \\
\end{array}
\]

\[
\begin{array}{cc}
a_1 & c_1 \\
\end{array}
\]

\[
\begin{array}{c}
e_1 \\
r_1 \\
\end{array}
\]

\[
\begin{array}{c}
c_2 \\
r_2 \\
\end{array}
\]

\[
\begin{array}{c}
e_3 \\
\end{array}
\]

\[
\begin{array}{c}
\text{e}3 = \begin{array}{c}
a_0c_0e_0 \\
\end{array}
\]

\[
\begin{array}{c}
a_1c_2a_0 \\
\end{array}
\]

\[
\begin{array}{c}
\text{c}_{2}e_{3} \\
r_{2}e_{3} \\
\end{array}
\]

\[
\begin{array}{c}
\text{c}_{2} \\
r_{2} \\
\end{array}
\]

\[
\begin{array}{c}
a_{1}a_{0} \\
\end{array}
\]

\[
\begin{array}{c}
\text{c}_{1} \\
e_{1} \\
\end{array}
\]

\[
\begin{array}{c}
a_{0}a_{1} \\
\end{array}
\]

\[
\begin{array}{c}
\text{c}_{0}c_{1} \\
e_{0}e_{1} \\
\end{array}
\]

\[
\begin{array}{c}
a_{0} \\
\end{array}
\]

\[
\begin{array}{c}
c_{0} \\
e_{0} \\
\end{array}
\]
Idea Halin
Idea Halin

\[ \Sigma = \]
Idea Halin

\[ \Sigma = \]

\[ a_0 \quad c_0 \quad e_0 \]

\[ a_1 \quad c_1 \quad e_1 \]

\[ r_1 \quad c_2 \quad e_3 \]

\[ a_{01} \quad c_{10} \quad e_{10} \]

\[ a_{00} \quad c_{01} \quad e_{01} \]

\[ a_0 \quad c_0 \quad e_0 \]

\[ a_0 \quad c_0 \quad e_0 \]

\[ a_{11} \quad c_{11} \quad e_{11} \]

\[ a_{01} \quad c_{01} \quad e_{01} \]

\[ a_0 \quad c_0 \quad e_0 \]

\[ a_0 \quad c_0 \quad e_0 \]

\[ a_{10} \quad c_{10} \quad e_{10} \]

\[ a_{00} \quad c_{00} \quad e_{00} \]

\[ a_0 \quad c_0 \quad e_0 \]

\[ a_0 \quad c_0 \quad e_0 \]

\[ a_{01} \quad c_{01} \quad e_{01} \]

\[ a_{00} \quad c_{00} \quad e_{00} \]

\[ a_0 \quad c_0 \quad e_0 \]

\[ a_0 \quad c_0 \quad e_0 \]
Idea Halin
Idea Halin
Idea Halin
Idea Halin

\[ \Sigma = \]

\[ a_0 \]

\[ c_0 \]

\[ e_0 \]

\[ a_1 \]

\[ c_1 \]

\[ e_1 \]

\[ r_1 \]

\[ c_2 \]

\[ r_2 \]

\[ e_3 \]
Idea Halin
Idea Halin
Idea Halin
Idea Halin
Proof I (Halin Graph)

Let $G = (V, E)$ be a Halin-graph with $V = \{v_1, v_2, v_3, \ldots, v_n\}$. 

Proof I (Halin Graph)

- Let $G = (V, E)$ be a Halin-graph with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $T = (V, E')$ be the tree of $G$ with root $v_1$. 
Proof I (Halin Graph)

- Let $G = (V, E)$ be a Halin-graph with $V = \{v_1, v_2, \ldots, v_n\}$.
- Let $T = (V, E')$ be the tree of $G$ with root $v_1$.
- Let $(a_1, a_2, \ldots, a_k)$ be the cycle connecting the leaves.
Proof I (Halin Graph)

- Let $G = (V, E)$ be a Halin-graph with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $T = (V, E')$ be the tree of $G$ with root $v_1$.
- Let $(a_1, a_2, \ldots, a_k)$ be the cycle connecting the leaves.
- Modify now $T$ as follows:
Proof I (Halin Graph)

1. Let $G = (V, E)$ be a Halin-graph with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
2. Let $T = (V, E')$ be the tree of $G$ with root $v_1$.
3. Let $(a_1, a_2, \ldots, a_k)$ be the cycle connecting the leaves.
4. Modify now $T$ as follows:
   - For each node $v$ define $f(v) = \{v\}$. 

Proof I (Halin Graph)

Let $G = (V, E)$ be a Halin-graph with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
Let $T = (V, E')$ be the tree of $G$ with root $v_1$.
Let $(a_1, a_2, \ldots, a_k)$ be the cycle connecting the leaves.
Modify now $T$ as follows:

- For each node $v$ define $f(v) = \{v\}$.
- For each edge $\{a, b\} \in E(T)$ generate a new node $v_{a,b}$ and define $f(v_{a,b}) = \{a, b\}$. 
Proof I (Halin Graph)

- Let $G = (V, E)$ be a Halin-graph with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $T = (V, E')$ be the tree of $G$ with root $v_1$.
- Let $(a_1, a_2, \ldots, a_k)$ be the cycle connecting the leaves.
- Modify now $T$ as follows:
  - For each node $v$ define $f(v) = \{v\}$.
  - For each edge $\{a, b\} \in E(T)$ generate a new node $v_{a,b}$ and define $f(v_{a,b}) = \{a, b\}$.
  - Replace each edge $\{a, b\} \in E(T)$ by $\{a, v_{a,b}\}, \{v_{a,b}, b\}$. 
Proof I (Halin Graph)

- Let $G = (V, E)$ be a Halin-graph with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $T = (V, E')$ be the tree of $G$ with root $v_1$.
- Let $(a_1, a_2, \ldots, a_k)$ be the cycle connecting the leaves.
- Modify now $T$ as follows:
  - For each node $v$ define $f(v) = \{v\}$.
  - For each edge $\{a, b\} \in E(T)$ generate a new node $v_{a,b}$ and define $f(v_{a,b}) = \{a, b\}$.
  - Replace each edge $\{a, b\} \in E(T)$ by $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
  - Replace each node $v$ with $\deg(v) > 3$ by a tree of degree 3.
Proof I (Halin Graph)

Let \( G = (V, E) \) be a Halin-graph with \( V = \{v_1, v_2, v_3, \ldots, v_n\} \).

Let \( T = (V, E') \) be the tree of \( G \) with root \( v_1 \).

Let \((a_1, a_2, \ldots, a_k)\) be the cycle connecting the leaves.

Modify now \( T \) as follows:

- For each node \( v \) define \( f(v) = \{v\} \).
- For each edge \( \{a, b\} \in E(T) \) generate a new node \( v_{a,b} \) and define \( f(v_{a,b}) = \{a, b\} \).
- Replace each edge \( \{a, b\} \in E(T) \) by \( \{a, v_{a,b}\}, \{v_{a,b}, b\} \).
- Replace each node \( v \) with \( \deg(v) > 3 \) by a tree of degree 3.
- For all nodes \( x \) which replace \( v \) define \( f(x) = \{v\} \).
Proof I (Halin Graph)

- Let $G = (V, E)$ be a Halin-graph with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $T = (V, E')$ be the tree of $G$ with root $v_1$.
- Let $(a_1, a_2, \ldots, a_k)$ be the cycle connecting the leaves.
- Modify now $T$ as follows:
  - For each node $v$ define $f(v) = \{v\}$.
  - For each edge $\{a, b\} \in E(T)$ generate a new node $v_{a,b}$ and define $f(v_{a,b}) = \{a, b\}$.
  - Replace each edge $\{a, b\} \in E(T)$ by $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
  - Replace each node $v$ with $\deg(v) > 3$ by a tree of degree 3.
  - For all nodes $x$ which replace $v$ define $f(x) = \{v\}$.
  - For each edge $\{a_i, a_{i+1}\}$ and for each node $y$ on the path from $a_i$ to $a_{i+1}$ define $f(y) = f(y) \cup \{a_i\}$. 


Proof I (Halin Graph)

- Let $G = (V, E)$ be a Halin-graph with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $T = (V, E')$ be the tree of $G$ with root $v_1$.
- Let $(a_1, a_2, \ldots, a_k)$ be the cycle connecting the leaves.
- Modify now $T$ as follows:
  - For each node $v$ define $f(v) = \{v\}$.
  - For each edge $\{a, b\} \in E(T)$ generate a new node $v_{a,b}$ and define $f(v_{a,b}) = \{a, b\}$.
  - Replace each edge $\{a, b\} \in E(T)$ by $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
  - Replace each node $v$ with $\deg(v) > 3$ by a tree of degree 3.
  - For all nodes $x$ which replace $v$ define $f(x) = \{v\}$.
  - For each edge $\{a_i, a_{i+1}\}$ and for each node $y$ on the path from $a_i$ to $a_{i+1}$ define $f(y) = f(y) \cup \{a_i\}$.
  - For edge $\{a_k, a_1\}$ and for each node $y$ on the path from $a_k$ to $a_1$ define $f(y) = f(y) \cup \{a_k\}$.
Proof I (Halin Graph)

- Let $G = (V, E)$ be a Halin-graph with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- Let $T = (V, E')$ be the tree of $G$ with root $v_1$.
- Let $(a_1, a_2, \ldots, a_k)$ be the cycle connecting the leaves.
- Modify now $T$ as follows:
  - For each node $v$ define $f(v) = \{v\}$.
  - For each edge $\{a, b\} \in E(T)$ generate a new node $v_{a,b}$ and define $f(v_{a,b}) = \{a, b\}$.
  - Replace each edge $\{a, b\} \in E(T)$ by $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
  - Replace each node $v$ with $\deg(v) > 3$ by a tree of degree 3.
  - For all nodes $x$ which replace $v$ define $f(x) = \{v\}$.
  - For each edge $\{a_i, a_{i+1}\}$ and for each node $y$ on the path from $a_i$ to $a_{i+1}$ define $f(y) = f(y) \cup \{a_i\}$.
  - For edge $\{a_k, a_1\}$ and for each node $y$ on the path from $a_k$ to $a_1$ define $f(y) = f(y) \cup \{a_k\}$.

We have for each node $z$: $|f(z)| \leq 4$. 
Idea outer-planar Graph
Idea outer-planar Graph

\[ \Sigma = \sum_{v_1, v_2, v_4} + \sum_{v_2, v_3, v_5} + \sum_{v_2, v_4, v_5} + \sum_{v_4, v_5, v_6} + \sum_{v_2, v_3, v_7} \]
Idea outer-planar Graph

\[ \Sigma = v_1v_2v_4 \]

\[ v_1v_2v_4 \]

\[ \Sigma = \]
Idea outer-planar Graph
Proof II (Outer-planar Graph)

Let $G = (V, E)$ be a outer-planar graph.
Proof II (Outer-planar Graph)

- Let $G = (V, E)$ be a outer-planar graph.
- Let $G = (V, E)$ be maximal, i.e. $G = (V, E \cup \{e\})$ is not outer-planar.
Proof II (Outer-planar Graph)

- Let $G = (V, E)$ be a outer-planar graph.
- Let $G = (V, E)$ be maximal, i.e. $G = (V, E \cup \{e\})$ is not outer-planar.
- Let $V'$ be the set of inner regions (windows).
Proof II (Outer-planar Graph)

- Let $G = (V, E)$ be a outer-planar graph.
- Let $G = (V, E)$ be maximal, i.e. $G = (V, E \cup \{e\})$ is not outer-planar.
- Let $V'$ be the set of inner regions (windows).
- Then is $V'$ a set of triangles.
Proof II (Outer-planar Graph)

- Let $G = (V, E)$ be a outer-planar graph.
- Let $G = (V, E)$ be maximal, i.e. $G = (V, E \cup \{e\})$ is not outer-planar.
- Let $V'$ be the set of inner regions (windows).
- Then is $V'$ a set of triangles.
- For $x \in V'$ let:
Proof II (Outer-planar Graph)

- Let $G = (V, E)$ be a outer-planar graph.
- Let $G = (V, E)$ be maximal, i.e. $G = (V, E \cup \{e\})$ is not outer-planar.
- Let $V'$ be the set of inner regions (windows).
- Then is $V'$ a set of triangles.
- For $x \in V'$ let:
  - $V(x)$ be the nodes of triangle $x$. 
Proof II (Outer-planar Graph)

- Let $G = (V, E)$ be a outer-planar graph.
- Let $G = (V, E)$ be maximal, i.e. $G = (V, E \cup \{e\})$ is not outer-planar.
- Let $V'$ be the set of inner regions (windows).
- Then $V'$ a set of triangles.
- For $x \in V'$ let:
  - $V(x)$ be the nodes of triangle $x$.
  - $E(x)$ be the edges of triangle $x$. 

Proof II (Outer-planar Graph)

- Let $G = (V, E)$ be a outer-planar graph.
- Let $G = (V, E)$ be maximal, i.e. $G = (V, E \cup \{e\})$ is not outer-planar.
- Let $V'$ be the set of inner regions (windows).
- Then is $V'$ a set of triangles.
- For $x \in V'$ let:
  - $V(x)$ be the nodes of triangle $x$.
  - $E(x)$ be the edges of triangle $x$.
- Define tree $(V', \{\{a, b\} \mid E(a) \cap E(b) \neq \emptyset\})$
Proof II (Outer-planar Graph)

- Let $G = (V, E)$ be a outer-planar graph.
- Let $G = (V, E)$ be maximal,
i.e. $G = (V, E \cup \{e\})$ is not outer-planar.
- Let $V'$ be the set of inner regions (windows).
- Then is $V'$ a set of triangles.
- For $x \in V'$ let:
  - $V(x)$ be the nodes of triangle $x$.
  - $E(x)$ be the edges of triangle $x$.
- Define tree $(V', \{\{a, b\} \mid E(a) \cap E(b) \neq \emptyset\})$
- Define $f$ by $f(x) = V(x)$. 
Theorems V

**Definition**
A planar graph $G = (V, E)$ is called $k$-outer-planar, iff there is a planar embedding of $G$ such that after deleting $k - 1$ times all nodes of the outer window the remaining graph embedded as an outer-planar graph.

**Theorem**
For $k$-outer-planar graphs $G = (V, E)$ holds: $\text{tw}(G) \leq 3 \cdot k - 1$
Theorems V

Definition

A planar graph $G = (V, E)$ is called $k$-outer-planar, iff there is a planar embedding of $G$ such that after deleting $k - 1$ times all nodes of the outer window the remaining graph embedded as an outer-planar graph.

Theorem

For $k$-outer-planar graphs $G = (V, E)$ holds: $tw(G) \leq 3 \cdot k - 1$
Definition

A graph $G = (V, E)$ is called SP-graph (series-parallel graph), iff it may be constructed by using series-parallel operations:
Theorems VI

Definition

A graph \( G = (V, E) \) is called SP-graph (series-parallel graph), iff it may be constructed by using series-parallel operations:

- \( G = (\{a, b\}, \emptyset, a, b) \) is a SP-graph.
Definition

A graph $G = (V, E)$ is called SP-graph (series-parallel graph), iff it may be constructed by using series-parallel operations:

- $G = (\{a, b\}, \emptyset, a, b)$ is a SP-graph.
- $G = (\{a, b\}, \{\{a, b\}\}, a, b)$ is a SP-graph.
Theorems VI

**Definition**

A graph $G = (V, E)$ is called SP-graph (series-parallel graph), iff it may be constructed by using series-parallel operations:

- $G = (\{a, b\}, \emptyset, a, b)$ is a SP-graph.
- $G = (\{a, b\}, \{\{a, b\}\}, a, b)$ is a SP-graph.
- If $G = (V, E, a, b)$ and $G = (V', E', a, b)$ are SP-graphs with $V \cap V' = \{a, b\}$ then $G = (V \cup V', E \cup E', a, b)$ is a SP-graph.
A graph $G = (V, E)$ is called SP-graph (series-parallel graph), iff it may be constructed by using series-parallel operations:

- $G = (\{a, b\}, \emptyset, a, b)$ is a SP-graph.
- $G = (\{a, b\}, \{\{a, b\}\}, a, b)$ is a SP-graph.
- If $G = (V, E, a, b)$ and $G = (V', E', a, b)$ are SP-graphs with $V \cap V' = \{a, b\}$ then $G = (V \cup V', E \cup E', a, b)$ is a SP-graph.
- If $G = (V, E, a, x)$ and $G = (V', E', x, b)$ are SP-graphs with $V \cap V' = \{x\}$ then $G = (V \cup V', E \cup E', a, b)$ is a SP-graph.

Theorem: A SP-graph has treewidth 2.
A graph $G = (V, E)$ is called SP-graph (series-parallel graph), iff it may be constructed by using series-parallel operations:

- $G = (\{a, b\}, \emptyset, a, b)$ is a SP-graph.
- $G = (\{a, b\}, \{\{a, b\}\}, a, b)$ is a SP-graph.
- If $G = (V, E, a, b)$ and $G = (V', E', a, b)$ are SP-graphs with $V \cap V' = \{a, b\}$ then $G = (V \cup V', E \cup E', a, b)$ is a SP-graph.
- If $G = (V, E, a, x)$ and $G = (V', E', x, b)$ are SP-graphs with $V \cap V' = \{x\}$ then $G = (V \cup V', E \cup E', a, b)$ is a SP-graph.
Definition

A graph $G = (V, E)$ is called SP-graph (series-parallel graph), iff it may be constructed by using series-parallel operations:

- $G = (\{a, b\}, \emptyset, a, b)$ is a SP-graph.
- $G = (\{a, b\}, \{\{a, b\}\}, a, b)$ is a SP-graph.
- If $G = (V, E, a, b)$ and $G = (V', E', a, b)$ are SP-graphs with $V \cap V' = \{a, b\}$ then $G = (V \cup V', E \cup E', a, b)$ is a SP-graph.
- If $G = (V, E, a, x)$ and $G = (V', E', x, b)$ are SP-graphs with $V \cap V' = \{x\}$ then $G = (V \cup V', E \cup E', a, b)$ is a SP-graph.

Theorem

A SP-graph has treewidth 2.
Minors

Definition (Minor)

A graph $G'$ is the minor of a graph $G$, iff an isomorphic image of $G'$ could be generated from $G$ by node-merging of connected nodes.
Definition (Minor)

A graph $G'$ is the minor of a graph $G$, iff an isomorphic image of $G'$ could be generated from $G$ by node-merging of connected nodes.

Merging of nodes:

- Let $G = (V, E)$
Definition (Minor)

A graph $G'$ is the minor of a graph $G$, iff an isomorphic image of $G'$ could be generated from $G$ by node-merging of connected nodes.

Merging of nodes:

- Let $G = (V, E)$
- Let $\{a, b\} \in E$, 
Minors

**Definition (Minor)**

A graph $G'$ is the minor of a graph $G$, iff an isomorphic image of $G'$ could be generated from $G$ by node-merging of connected nodes.

**Merging of nodes:**

- Let $G = (V, E)$
- Let $\{a, b\} \in E$
- Then the node-merging of $a$ and $b$ is possible:
Definition (Minor)

A graph $G'$ is the minor of a graph $G$, iff an isomorphic image of $G'$ could be generated from $G$ by node-merging of connected nodes.

Merging of nodes:

- Let $G = (V, E)$
- Let $\{a, b\} \in E$,
- Then the node-merging of $a$ and $b$ is possible:
- $G' = (V \setminus b, (E \setminus \{\{v, b\} \mid v \in V\}) \cup \{\{v, a\} \mid \{v, b\} \in E\})$
Theorem

A graph $G$ with $\text{tw}(G) \leq k$ has no $K_{k+2}$ minor.
Theorems VII

Theorem

A graph $G$ with $tw(G) \leq k$ has no $K_{k+2}$ minor.

Theorem

Graphs $G$ with $tw(G) \leq k$ could be described by a bounded sequence of minors.
Theorems VII

Theorem

A graph $G$ with $\text{tw}(G) \leq k$ has no $K_{k+2}$ minor.

Theorem

Graphs $G$ with $\text{tw}(G) \leq k$ could be described by a bounded sequence of minors.

Theorem

Any problem described in $\text{MS}_2$ on a graph $G$ with $\text{tw}(G) \leq k$ is solvable in polynomial time.
Questions

1. What is the definition of bandwidth?
Questions

1. What is the definition of bandwidth?
2. Which problems may be solved on graphs with bounded bandwidth?
Questions

1. What is the definition of bandwidth?
2. Which problems may be solved on graphs with bounded bandwidth?
3. What is the idea for this?
Questions

1. What is the definition of bandwidth?
2. Which problems may be solved on graphs with bounded bandwidth?
3. What is the idea for this?
4. What is the definition of pathwidth?
Questions

1. What is the definition of bandwidth?
2. Which problems may be solved on graphs with bounded bandwidth?
3. What is the idea for this?
4. What is the definition of pathwidth?
5. Which problems may be solved on graphs with bounded pathwidth?
Questions

1. What is the definition of bandwidth?
2. Which problems may be solved on graphs with bounded bandwidth?
3. What is the idea for this?
4. What is the definition of pathwidth?
5. Which problems may be solved on graphs with bounded pathwidth?
6. What is the idea for this?
Questions

1. What is the definition of bandwidth?
2. Which problems may be solved on graphs with bounded bandwidth?
3. What is the idea for this?
4. What is the definition of pathwidth?
5. Which problems may be solved on graphs with bounded pathwidth?
6. What is the idea for this?
7. Compare the bandwidth and the pathwidth of a graph.
Questions

1. What is the definition of bandwidth?
2. Which problems may be solved on graphs with bounded bandwidth?
3. What is the idea for this?
4. What is the definition of pathwidth?
5. Which problems may be solved on graphs with bounded pathwidth?
6. What is the idea for this?
7. Compare the bandwidth and the pathwidth of a graph.
8. What is the definition of treewidth?
Questions

1. What is the definition of bandwidth?
2. Which problems may be solved on graphs with bounded bandwidth?
3. What is the idea for this?
4. What is the definition of pathwidth?
5. Which problems may be solved on graphs with bounded pathwidth?
6. What is the idea for this?
7. Compare the bandwidth and the pathwidth of a graph.
8. What is the definition of treewidth?
9. Which problems may be solved on graphs with bounded treewidth?
Questions

1. What is the definition of bandwidth?
2. Which problems may be solved on graphs with bounded bandwidth?
3. What is the idea for this?
4. What is the definition of pathwidth?
5. Which problems may be solved on graphs with bounded pathwidth?
6. What is the idea for this?
7. Compare the bandwidth and the pathwidth of a graph.
8. What is the definition of treewidth?
9. Which problems may be solved on graphs with bounded treewidth?
10. What is the idea for this?
Questions

1. Compare the treewidth and the pathwidth of a graph.
Questions

1. Compare the treewidth and the pathwidth of a graph.
2. What is the definition of a partial $k$-tree?
Questions

1. Compare the treewidth and the pathwidth of a graph.
2. What is the definition of a partial $k$-tree?
3. Compare treewidth and partial $k$-tree.
Questions

1. Compare the treewidth and the pathwidth of a graph.
2. What is the definition of a partial $k$-tree?
3. Compare treewidth and partial $k$-tree.
4. What is the treewidth of a Halin-graph?
Questions

1. Compare the treewidth and the pathwidth of a graph.
2. What is the definition of a partial $k$-tree?
3. Compare treewidth and partial $k$-tree.
4. What is the treewidth of a Halin-graph?
5. What is the treewidth of a cactus?
Questions

1. Compare the treewidth and the pathwidth of a graph.
2. What is the definition of a partial $k$-tree?
3. Compare treewidth and partial $k$-tree.
4. What is the treewidth of a Halin-graph?
5. What is the treewidth of a cactus?
6. What is the treewidth of a near-tree($k$)?
Questions

1. Compare the treewidth and the pathwidth of a graph.
2. What is the definition of a partial $k$-tree?
3. Compare treewidth and partial $k$-tree.
4. What is the treewidth of a Halin-graph?
5. What is the treewidth of a cactus?
6. What is the treewidth of a near-tree($k$)?
7. What is the treewidth of an outer-planar graphen?
Questions

1. Compare the treewidth and the pathwidth of a graph.
2. What is the definition of a partial $k$-tree?
3. Compare treewidth and partial $k$-tree.
4. What is the treewidth of a Halin-graph?
5. What is the treewidth of a cactus?
6. What is the treewidth of a near-tree($k$)?
7. What is the treewidth of an outer-planar graphen?
8. What is the treewidth of a $k$-outer-planar Graphen?
Questions

1. Compare the treewidth and the pathwidth of a graph.
2. What is the definition of a partial $k$-tree?
3. Compare treewidth and partial $k$-tree.
4. What is the treewidth of a Halin-graph?
5. What is the treewidth of a cactus?
6. What is the treewidth of a near-tree($k$)?
7. What is the treewidth of an outer-planar graphen?
8. What is the treewidth of a $k$-outer-planar Graphen?
9. What is the treewidth of a SP-graph?