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”.
- 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.
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.
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.
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.
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.
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.
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 of Bandwidth

Definition (Bandwidth)

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

- A **labeling of** $G$ is a function $e : V \to \mathbb{N}$ 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 $\text{bw}(G) = \min_{e : V \to \mathbb{N}}\{\text{bw}(e, G)\}$. 
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 $\text{bw}(e, G) = \max\{\text{dist}(e, v, v') | \{v, v'\} \in E\}$.

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

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 $e : V \to \mathbb{N}$ 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 $\text{bw}(G) = \min_{e: V \to \mathbb{N}} \{\text{bw}(e, G)\}$. 
Example
Example

\[ e(v_1) = 1 \]

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

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

\begin{align*}
\text{Example} \\
\text{Graph:} v_1, v_2, v_3, v_4, v_5, v_6 \\
\text{Edges:} e(v_1) = 1, e(v_2) = 2
\end{align*}
Example

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

The bandwidth of a graph is defined as the minimum value of the bandwidth of all its edge labelings. The bandwidth of an edge is the difference between the labels of its endpoints.

Consider the following graph:

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

The bandwidth of this graph is $5$, as it is the minimum difference between the labels of the endpoints of any edge.

The adjacency matrix of the graph is:

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

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

Bandwidth:

```
\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}
```

Bandwidth = 2
Example

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

\[ v_2 \]
\[ v_1 \]
\[ v_3 \]
\[ v_6 \]
\[ v_4 \]
\[ v_5 \]
Example

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

\text{bandwidth} = 5
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*}

\text{bandwidth} = 5
Example

Bandwidth
Pathwidth
Treewidth
k-Trees
Applications

4:3 Definition 10/18

Walter Unger 15.6.2016 18:26 SS2016 RWTH

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

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

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

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

\[ e(v_2) = 2 \]
\[ e(v_1) = 1 \]
\[ e(v_6) = 6 \]
\[ e(v_3) = 3 \]
\[ e(v_4) = 4 \]
\[ e(v_5) = 5 \]
\[ \text{bandwidth} = 5 \]
Example

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

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

- **Bandwidth**
  - $e(v_2) = 2$
  - $e(v_1) = 1$
  - $e(v_6) = 6$
  - $e(v_5) = 5$

  $\text{bandwidth} = 5$

- **Pathwidth**
  - $e(v_2) = 1$
  - $e(v_1) = 2$
  - $e(v_6) = 4$

- **Treewidth**
  - $e(v_3) = 3$
  - $e(v_4) = 4$
  - $e(v_5) = 3$
Example

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

\text{bandwidth} = 5
Example

Bandwidth
Pathwidth
Treewidth
k-Trees
Applications

\[ e(v_2) = 2 \]
\[ e(v_1) = 1 \]
\[ e(v_6) = 6 \]
\[ e(v_5) = 5 \]
\[ \text{bandwidth} = 5 \]

\[ e(v_2) = 1 \]
\[ e(v_1) = 2 \]
\[ e(v_6) = 4 \]
\[ e(v_5) = 6 \]
Example

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

bandwidth = 5

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

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

**Example**

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

bandwidth = 5

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

bandwidth = 2
**Example**

**Bandwidth**

- \( e(v_1) = 1 \)
- \( e(v_2) = 2 \)
- \( e(v_3) = 3 \)
- \( e(v_6) = 6 \)
- \( e(v_5) = 5 \)

**Pathwidth**

**Treewidth**

**k-Trees**

**Applications**

- Bandwidth: 5
  - \( \begin{pmatrix}
    0 & 1 & 1 & 0 & 0 & 1 \\
    1 & 0 & 1 & 0 & 0 & 0 \\
    1 & 1 & 0 & 1 & 1 & 0 \\
    0 & 0 & 1 & 0 & 1 & 0 \\
    0 & 0 & 1 & 1 & 0 & 1 \\
    1 & 0 & 0 & 0 & 1 & 0 
  \end{pmatrix} \)

- Bandwidth: 2
  - \( \begin{pmatrix}
    0 & 1 & 1 & 0 & 0 & 0 \\
    1 & 0 & 1 & 1 & 0 & 0 \\
    1 & 1 & 0 & 0 & 1 & 0 \\
    0 & 1 & 0 & 0 & 1 & 1 \\
    0 & 0 & 1 & 1 & 0 & 1 \\
    0 & 0 & 0 & 1 & 1 & 0 
  \end{pmatrix} \)
Example: second view

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

\text{bandwidth} = 5

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

\text{bandwidth} = 2
Example: second view

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

Bandwidth = 5

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

Bandwidth = 2
Example: second view

\[ e(v_2) = 2 \]
\[ e(v_1) = 1 \]
\[ e(v_6) = 6 \]
\[ e(v_5) = 5 \]

bandwidth = 5

\[ e(v_2) = 1 \]
\[ e(v_1) = 2 \]
\[ e(v_6) = 4 \]
\[ e(v_5) = 6 \]

bandwidth = 2
Example: second view

```
e(v2) = 2
e(v1) = 1
e(v6) = 6

bandwidth = 5
```

```
e(v2) = 1
e(v1) = 2
e(v6) = 4

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?

Theorem

The bandwidth-problem is NP-complete.
Bandwidth

**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.*
Bandwidth on Caterpillars

**Definition (Caterpillar)**

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

\[
\begin{align*}
\Sigma &= 0 \\
\end{align*}
\]

**Theorem**

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

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

**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?

**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*
**k-bandwidth**

**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 graphs with $bw(G) = k$, the following problem may be solved in linear time:

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

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

Theorem

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

Theorem

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

- Let $bw(G) = k$.
- 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 $B_{i+1}$. 
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 $B_{i+1}$.
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 \(v_1, v_2, \ldots, v_{i+k}\).
- 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 \(B_{i+1}\).
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 $B_{i+1}$.
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 $B_{i+1}$.
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 $B_{i+1}$.
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 \( B_{i+1} \).
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 $B_{i+1}$. 
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

Σ = 0
3-Colouring
3-Colouring
3-Colouring
3-Colouring
3-Colouring
3-Colouring
3-Colouring
3-Colouring
3-Colouring
3-Colouring

\[
\begin{align*}
&v_0 \quad v_1 \quad v_2 \quad v_3 \\
v_4 \quad v_5 \quad v_6 \quad v_7 \\
v_8 \quad v_9
\end{align*}
\]

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

\[ \Sigma = 0 \]
Independent Set

The diagram illustrates a graph with nodes labeled $v_0$ to $v_9$, with the edges connecting them. The coloring and numbering of nodes suggest an analysis or solution to a problem, potentially a problem related to independent sets or similar graph theory concepts.
Independent Set
Independent Set
Independent Set
Independent Set
Independent Set

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

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

- Compute $C_{i+1}$ from $C_i$ by: $(I, s) \in C_{i+1} \iff \exists (I', s') \in C_i$
  
  $$I = I' \setminus \{v_i\}, s = s' \text{ or}$$
  $$I = (I' \cup \{v_{i+k+1}\}) \setminus \{v_i\}, s = s' + 1, I \text{ is stable set}$$
Independent Set

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

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

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

- **Compute $C_{i+1}$** from $C_i$ by: $(I, s) \in C_{i+1} \iff \exists (I', s') \in C_i$
  
  $$I = I' \setminus \{v_i\}, s = s' \text{ or }$$

  $$I = (I' \cup \{v_{i+k+1}\}) \setminus \{v_i\}, s = s' + 1, I \text{ is stable set}$$
Independent Set

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

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

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

- **Compute** $C_{i+1}$ from $C_i$ by: $(l, s) \in C_{i+1} \iff \exists (l', s') \in C_i$
  
  $$l = l' \setminus \{v_i\}, s = s' \text{ or }$$
  
  $$l = (l' \cup \{v_{i+k+1}\}) \setminus \{v_i\}, s = s' + 1, l \text{ is stable set}$$
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: $(l, s) \in C_i \iff$
  
  $$\exists S \subset \{v_1, \ldots, v_{i+k}\} : S \cap \{v_i, \ldots v_{i+k}\} = l, |S| = s, S \text{ is independent set}$$

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

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

  $l = l' \setminus \{v_i\}, s = s'$ or
  
  $l = (l' \cup \{v_{i+k+1}\}) \setminus \{v_i\}, s = s' + 1, l \text{ is stable set}$
Hamilton Cycle
Hamilton Cycle
Hamilton Cycle

- **v0**
- **v1**
- **v2**
- **v3**
- **v4**
- **v5**
- **v6**
- **v7**
- **v8**
- **v9**

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

\[ \Sigma = 0 \]
Hamilton Cycle

Graph representation of a Hamilton cycle with nodes v0, v1, v2, v3, v4, v5, v6, v7, v8, v9, and their corresponding colors indicating open endpoint, visited node, and nonvisited node.
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

$v_0\rightarrow v_1\rightarrow v_2\rightarrow v_3\rightarrow \ldots\rightarrow v_9$

- $v_0$: Open Endpoint
- $v_1\ldots v_9$: Visited Node
- Red: Visited Node
- Yellow: Nonvisited Node
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node
Hamilton Cycle

- Hamilton Cycle is a cycle that visits each vertex exactly once and returns to the starting vertex.
- The graph shown represents a Hamilton Cycle with nodes labeled from $v_0$ to $v_9$.
- Each node has a color indicating its status:
  - Green: Open Endpoint
  - Red: Visited Node
  - Yellow: Nonvisited Node
Hamilton Cycle

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

\[ \Sigma = 0 \]
Hamilton Cycle

Nodes: v0, v1, v2, v3, v4, v5, v6, v7, v8, v9

- Open Endpoint
- Visited Node
- Nonvisited Node

Σ = 0
Hamilton Cycle
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node
Hamilton Cycle

\[ \Sigma = \text{Open Endpoint} \]
\[ \text{Visited Node} \]
\[ \text{Nonvisited Node} \]
Hamilton Cycle

- **v0**
- **v1**
- **v2**
- **v3**
- **v4**
- **v5**
- **v6**
- **v7**
- **v8**
- **v9**

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

- **v0**
- **v1**
- **v2**
- **v3**
- **v4**
- **v5**
- **v6**
- **v7**
- **v8**
- **v9**

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

- Open Endpoint
- Visited Node
- Nonvisited Node
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

$\sum = 0$
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node

Diagram shows a network of nodes (v0, v1, v2, v3, v4, v5, v6, v7, v8, v9) connected in a cycle with different colors indicating open endpoints, visited nodes, and nonvisited nodes.
Hamilton Cycle
Hamilton Cycle

\[ \Sigma = 0 \]
Hamilton Cycle

\[ \Sigma = 0 \]
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node
Hamilton Cycle

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

- Open Endpoint
- Visited Node
- Nonvisited Node

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

- Open Endpoint
- Visited Node
- Nonvisited Node
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node
Hamilton Cycle

\[ \Sigma = 0 \]
Hamilton Cycle

- Open Endpoint
- Visited Node
- Nonvisited Node

Diagram showing a Hamilton Cycle with nodes and edges labeled with colors to indicate open endpoints, visited nodes, and nonvisited nodes.
Hamilton Cycle

![Diagram of Hamilton Cycle with nodes and connections showing open endpoints, visited nodes, and nonvisited nodes.]
Hamilton Cycle
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 $\{v_1, v_2, \ldots, v_n\}$.
  - $\exists a, b : n - k \leq a, b \leq n$:
    - $\forall j \in \{1, 2, \ldots, n\} \setminus \{a, b\} : \delta_H(v_a) = 2$.
    - $\delta_H(v_a) = \delta_H(v_b) = 1$.
    - $\{v_a, v_b\} \in E$. 
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 $v_j \in C$.
  - 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 $\{v_1, v_2, \ldots, v_n\}$.
  - $\exists a, b : n - k \leq a, b \leq n$:
    - $\forall j \in \{1, 2, \ldots, n\} \setminus \{a, b\} : \delta_H(v_a) = 2$.
    - $\delta_H(v_a) = \delta_H(v_b) = 1$.
    - $\{v_a, v_b\} \in E$. 
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 \( \{v_1, v_2, \ldots, v_n\} \).
  - \( \exists a, b : n - k \leq a, b \leq n : \)
    - \( \forall j \in \{1, 2, \ldots, n\} \setminus \{a, b\} : \delta_H(v_a) = 2 \).
    - \( \delta_H(v_a) = \delta_H(v_b) = 1 \).
    - \( \{v_a, v_b\} \in E \).
Input: \( G = (V, E) \) with \( \text{bw}(G) \leq k \):

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

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 \( \{v_1, v_2, \ldots, v_n\} \).
- \( \exists a, b : n - k \leq a, b \leq n \):
  - \( \forall j \in \{1, 2, \ldots, n\} \setminus \{a, b\} : \delta_H(v_a) = 2 \).
  - \( \delta_H(v_a) = \delta_H(v_b) = 1 \).
  - \( \{v_a, v_b\} \in E \).
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 $\{v_1, v_2, \ldots, v_{i+k}\}$:
  
  - $H = (\{v_1, v_2, \ldots, v_{i+k}\}, F)$ 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 $\{v_1, v_2, \ldots, v_n\}$.
  - $\exists a, b : n - k \leq a, b \leq n$:
    
    - $\forall j \in \{1, 2, \ldots, n\} \setminus \{a, b\} : \delta_H(v_a) = 2$.
    - $\delta_H(v_a) = \delta_H(v_b) = 1$.
    - $\{v_a, v_b\} \in E$. 
Hamilton-Cycle

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

- **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 $\{v_1, v_2, \ldots, v_n\}$.
  - $\exists a, b : n - k \leq a, b \leq n$:
    - $\forall j \in \{1, 2, \ldots, n\} \setminus \{a, b\} : \delta_H(v_a) = 2$.
    - $\delta_H(v_a) = \delta_H(v_b) = 1$.
    - $\{v_a, v_b\} \in E$. 
Hamilton-Cycle

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

- **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 $\{v_1, v_2, \ldots, v_n\}$.
  - $\exists a, b : n - k \leq a, b \leq n$:
    
    - $\forall j \in \{1, 2, \ldots, n\} \setminus \{a, b\} : \delta_H(v_a) = 2$.
    - $\delta_H(v_a) = \delta_H(v_b) = 1$.
    - $\{v_a, v_b\} \in E$. 
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 $\{v_1, v_2, \ldots, v_{i+k}\}$:
  - $H = (\{v_1, v_2, \ldots, v_{i+k}\}, F)$ 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 $\{v_1, v_2, \ldots, v_n\}$.
  - $\exists a, b : n - k \leq a, b \leq n$:
    - $\forall j \in \{1, 2, \ldots, n\} \setminus \{a, b\} : \delta_H(v_a) = 2$.
    - $\delta_H(v_a) = \delta_H(v_b) = 1$.
    - $\{v_a, v_b\} \in E$. 
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 $\{v_1, v_2, \ldots, v_n\}$.
  - $\exists a, b : n - k \leq a, b \leq n$:
    - $\forall j \in \{1, 2, \ldots, n\} \setminus \{a, b\} : \delta_H(v_a) = 2$.
    - $\delta_H(v_a) = \delta_H(v_b) = 1$.
    - $\{v_a, v_b\} \in E$. 
Hamilton-Cycle

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

- **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 $\{v_1, v_2, \ldots, v_n\}$.
  - $\exists a, b : n - k \leq a, b \leq n$:
    - $\forall j \in \{1, 2, \ldots, n\} \setminus \{a, b\} : \delta_H(v_a) = 2$.
    - $\delta_H(v_a) = \delta_H(v_b) = 1$.
    - $\{v_a, v_b\} \in E$. 
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 $v_j \in C$.
  - 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 $\{v_1, v_2, \ldots, v_n\}$.
  - $\exists a, b : n - k \leq a, b \leq n$:
    - $\forall j \in \{1, 2, \ldots, n\} \setminus \{a, b\} : \delta_H(v_a) = 2$.
    - $\delta_H(v_a) = \delta_H(v_b) = 1$.
    - $\{v_a, v_b\} \in E$. 

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 $\{v_1, v_2, \ldots, v_{i+k}\}$:
  
  - $H = (\{v_1, v_2, \ldots, v_{i+k}\}, F)$ 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 $\{v_1, v_2, \ldots, v_n\}$.
  - $\exists a, b : n - k \leq a, b \leq n$:
    
    - $\forall j \in \{1, 2, \ldots, n\} \setminus \{a, b\} : \delta_H(v_a) = 2$.
    - $\delta_H(v_a) = \delta_H(v_b) = 1$.
    - $\{v_a, v_b\} \in E$. 
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 $v_j \in C$.
  - 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 $\{v_1, v_2, \ldots, v_n\}$.
  - $\exists a, b : n - k \leq a, b \leq n$:
    
    - $\forall j \in \{1, 2, \ldots, n\} \setminus \{a, b\} : \delta_H(v_a) = 2$.
    - $\delta_H(v_a) = \delta_H(v_b) = 1$.
    - $\{v_a, v_b\} \in E$. 
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 on Bandwidth

Definition (Diameter and Radius)

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

- 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 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:

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

**Theorem (Lower Bound for Bandwidth of a Complete Tree)**

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

\[
\text{bw}(G) \geq \left\lceil \frac{2^k - 1}{k} \right\rceil
\]
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:

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

**Theorem (Lower Bound for Bandwidth of a Complete Tree)**

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

$$bw(G) \geq \left\lceil \frac{2^k - 1}{k} \right\rceil = \left\lceil \frac{2^{k+1} - 2}{2k} \right\rceil.$$
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 $\mathcal{P} \neq \mathcal{NP}$.

Theorem

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

Theorem

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

**Theorem**

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

**Theorem**

*It is not possible to approximate the bandwidth-problem for caterpillars 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 $\mathcal{P} \neq \mathcal{NP}$.

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

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

Diagram showing the concept of pathwidth with labeled nodes and edges.
Idea of Pathwidth

Σ = 0
Idea of Pathwidth
Idea of Pathwidth

The diagram illustrates the concept of pathwidth in graph theory. The pathwidth of a graph is a measure of how similar the graph is to a path. It is defined as the minimum, over all possible topological sorts of the vertices of the graph, of the largest number of vertices that are adjacent to the same set of vertices in the sorted order.

In the diagram, vertices are represented by circles, and edges are represented by lines connecting the circles. The pathwidth is determined by the number of edges that cross between different parts of the graph when it is sorted.

The red circles highlight a particular part of the graph where the pathwidth is being calculated. The pathwidth is calculated by considering all possible ways to sort the vertices and then determining the maximum number of edges that cross between any two consecutive groups of vertices in the sorted order. The pathwidth is the minimum of these maximum edge crossings over all possible sorts.
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth
Idea of Pathwidth

The diagram illustrates the concept of Pathwidth in graph theory. Each node represents a vertex in the graph, and the arcs between nodes represent the edges. The pathwidth is defined as the minimum width of a path decomposition of the graph. A path decomposition is a sequence of subsets of vertices, where each subset is a path in the graph, and every edge is contained in at least one of these subsets. The width of a path decomposition is the maximum number of vertices in any subset minus one. The idea is to find a decomposition that minimizes the maximum width, which corresponds to the pathwidth of the graph.
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 $\text{pw}(K_{1,2n+1}) = 1$
**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 $\text{pw}(K_{1,2n+1}) = 1$
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 \( f(b) \cap f(a) \subset f(c) \) hold.
3. \( \forall x \in V_p : |f(x)| \leq k + 1. \)

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 \( \text{pw}(K_{1,2n+1}) = 1 \)
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 $\text{pw}(K_{1,2n+1}) = 1$
**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 $\text{pw}(K_{1, 2n+1}) = 1$
**Pathwidth**

**Definition**

A graph $G = (V, E)$ has pathwidth $k$, iff there is a path $P = (V_p, E_p)$ and a mapping $f : V_p \to \mathcal{P}(V)$ 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 $\text{pw}(K_{1,2n+1}) = 1$
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 $\text{pw}(K_{1,2n+1}) = 1$
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 $\{v_i, v_{i+d}\} \subseteq f(p_i)$ follows.
- Thus we have: $pw(G) \leq k$
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 $P_{n-k} = \{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} \mid 1 \leq j \leq n-k\}$ 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.
- Thus we have: $\text{pw}(G) \leq k$
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 $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} \mid 1 \leq j \leq n-k\})$ 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.
- Thus we have: $\text{pw}(G) \leq k$
### Theorems I

**Theorem**

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

- 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 $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} \mid 1 \leq j \leq n - k\})$ 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.
- Thus we have: $\text{pw}(G) \leq k$
Theorem I

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 $P_{n-k} = \{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} \mid 1 \leq j \leq n - k\}$ 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.
- Thus we have: $\text{pw}(G) \leq k$
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 \( P_{n-k} = (\{p_1,p_2,\ldots,p_{n-k+1}\}, \{\{p_j,p_{j+1}\} | 1 \leq j \leq n-k\}) \) be a path with \( n-k+1 \) nodes.
- Define function \( f : \{p_1,p_2,\ldots,p_{n-k+1}\} \mapsto P(V) : \)
  \( f(p_i) = \{v_i,v_{i+1},\ldots,v_{i+k}\} \).
- 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: \( pw(G) \leq k \).
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 $f : \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto \mathcal{P}(V)$:
  - $f(p_i) = \{v_i, v_{i+1}, \ldots, v_{i+k}\}$.
  - 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: $pw(G) \leq k$
Theorems I

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 $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} \mid 1 \leq j \leq n-k\})$ 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.
- Thus we have: $\text{pw}(G) \leq k$
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: $|f(p_i)| = k + 1$.
- And if $\{v_i, v_{i+d}\} \in E$ hold, then $\{v_i, v_{i+d}\} \subseteq f(p_i)$ follows.
- Thus we have: $pw(G) \leq k$
Theorems 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 $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} \mid 1 \leq j \leq n-k\})$ 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.
- Thus we have: $pw(G) \leq k$
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 $P_{n-k} = \{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} | 1 \leq j \leq n-k\}$ be a path with $n-k+1$ nodes.
- Define function $f : \{p_1, p_2, \ldots, p_{n-k+1}\} \mapsto P(V)$:
  
  $$f(p_i) = \{v_i, v_{i+1}, \ldots, v_{i+k}\}.$$ 

  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: $pw(G) \leq k$
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 $k$.

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

- Independent-Set, Clique, Vertex-Cover
- Colouring-problem
- Hamilton-Cycle, Hamilton-Path
Theorem II

<table>
<thead>
<tr>
<th>Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let ( G = (V, E) ) be a graph. Then the following hold:</td>
</tr>
<tr>
<td>- The problem, to compute the pathwidth of a graph, is NP-complete.</td>
</tr>
<tr>
<td>- For a fixed ( k ) it is possible to check in linear time ( O(n + m) ), if a graph has pathwidth ( k ).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let ( G = (V, E) ) be a graph with ( \text{pw}(G) = k ). The following problem may be solved in linear time:</td>
</tr>
<tr>
<td>- Independent-Set, Clique, Vertex-Cover</td>
</tr>
<tr>
<td>- Colouring-problem</td>
</tr>
<tr>
<td>- Hamilton-Cycle, Hamilton-Path</td>
</tr>
</tbody>
</table>
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
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 $pw(G) = k$. 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 $|V| = n$.
- 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 $I^j_i \subset f(p_i)$ 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 $f(p_{i+1})$:
- For each subset $I^j_{i+1}$ 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 $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} | 1 \leq j \leq n - k\})$ 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 $I_i^j \subset f(p_i)$ store:
  - $l_i^j$ and $w_i^j$ with
  - $w_i^j = |l|$ the size of the largest independent set $l$ on $\bigcup_{j=1}^i f(p_j)$ with $l_i^j \subset l$.

- Iteration step on $f(p_{i+1})$:
  - For each subset $l_{i+1}^j$ 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 $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} | 1 \leq j \leq n - k\})$ 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:
- **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 $k$.
- 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
- 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 $k$.
- For each subset $I_i \subset f(p_i)$ 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
  - 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 $k$.

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

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 \( k \).
- 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 $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} | 1 \leq j \leq n - k\})$ 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
  - 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 \( k \).
- For each subset \( l^j_i \subset f(p_i) \) store:
  - \( l^j_i \) and \( w^j_i \) with
  - \( w^j_i = |l| \) the size of the largest independent set \( l \) on \( \bigcup_{j=1}^{i} f(p_j) \) with \( l^j_i \subset l \).
- Iteration step on \( f(p_{i+1}) \):
  - For each subset \( l^j_{i+1} \) 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 $P_{n-k} = (\{p_1, p_2, \ldots, p_{n-k+1}\}, \{\{p_j, p_{j+1}\} | 1 \leq j \leq n - k\})$ 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
- 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 $k$ with:
  - $|f(p_i) \oplus f(p_{i+1})| = 1$,
    - $f(p_i) \hat{\cup} \{x\} = f(p_{i+1})$ or
    - $f(p_i) = f(p_{i+1}) \hat{\cup} \{x\}$
- 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)

- 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)

- 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)

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)

- 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:
  - $|f(p_i) \odot f(p_{i+1})| = 1$,
    - $f(p_i) \cup \{x\} = f(p_{i+1})$ 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** $f(p_i)$: $D(f(p_i))$

- 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 $f(p_{i+1})$. 
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 $f(p_{i+1})$. 
Example (Independent Set)

Thus we only have to define the following:

- What we store for $f(p_i)$: $D(f(p_i))$
- The procedure $D(f(p_1)) := Init(f(p_1))$, 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)

- $D(f(p_i)) = \{(l, w) \mid l \text{ is IS on } f(p_i) \land w = \text{Wert}(l, i)\}$ 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
  - $w = \max\{w' \mid (l \cup \{x\}, w') \in D(f(p_i))$ 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
  - $(l \setminus \{x\}, w - 1) \in D(f(p_i))$ and $l$ is IS.
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:
  \[
  \text{Wert}(l, i) = \max\{|l'| \mid l' \subseteq \bigcup_{j=1}^{i} f(p_j) \land l' \text{ ist IS} \land l \subseteq l'\}
  \]

- \( \text{Init}(f(p_1)) = \{(l, w) \mid l \text{ is IS on } f(p_1) \land w = |l|\}, \) compute all IS \( l \subseteq f(p_1) \) 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
  - \((l \setminus \{x\}, w - 1) \in D(f(p_i)) \) and \( l \) is IS.
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
  - $w = \max\{w' \mid (l \cup \{x\}, w') \in D(f(p_i))$ 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
  - $(l \setminus \{x\}, w - 1) \in D(f(p_i))$ and $l$ is IS.
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:
  \[
  \text{Wert}(l, i) = \max\{|l'| \mid l' \subseteq \bigcup_{j=1}^{i} f(p_j) \land l' \text{ ist IS} \land l \subseteq l'\}
  \]

- \( \text{Init}(f(p_1)) = \{(l, w) \mid l \text{ is IS on } f(p_1) \land w = |l|\}, \)
  compute all IS \( l \subseteq f(p_1) \) 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
  - \((l \setminus \{x\}, w - 1) \in D(f(p_i))\) and \(l\) is IS.
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:
  \[
  \text{Wert}(l, i) = \max\{ |l'| \mid l' \subset \bigcup_{j=1}^{i} f(p_j) \land l' \text{ ist IS } \land l \subset l' \}\]

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

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

- $(l, w) \in Add(D(f(p_i)), x)$ iff:
  - $(l, w) \in D(f(p_i))$ or
  - $(l \setminus \{x\}, w - 1) \in D(f(p_i))$ and $l$ is IS.
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
  - \( w = \max\{w' \mid (l \cup \{x\}, w') \in D(f(p_i)) \text{ or } (l, w') \in D(f(p_i))\} \).

- \( (l, w) \in \text{Add}(D(f(p_i))), x \) iff:
  - \( (l, w) \in D(f(p_i)) \) or
  - \( (l \setminus \{x\}, w - 1) \in D(f(p_i)) \) and \( l \) is IS.
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
  - \(w = \max\{w' \mid (l \cup \{x\}, w') \in D(f(p_i))\) 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
  - \((l \setminus \{x\}, w - 1) \in D(f(p_i))\) and \(l\) is IS.
Example (Independent Set)

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

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

- \((l, w) \in Del(D(f(p_i))), x) \) 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 \((l, w') \in D(f(p_i))\}\).

- \((l, w) \in Add(D(f(p_i))), x) \) 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 $f : V_T \rightarrow \mathcal{P}(V)$ 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 $k$. 
Definition

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

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

A graph \( G = (V, E) \) has treewidth \( k \), iff there is a tree \( T = (V_T, E_T) \) and a mapping \( f : V_T \to \mathcal{P}(V) \) 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 \( k \).
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 $f(b) \cap f(a) \subset f(c)$ 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)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)
Example (Pathwidth and Treewidth)

Diagram showing a network of nodes and edges illustrating the concepts of Pathwidth and Treewidth. The nodes are labeled with numbers, and the edges connect them in a hierarchical manner, demonstrating the network structure.

Table below the diagram:

<table>
<thead>
<tr>
<th>00</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>08</th>
<th>09</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01</td>
<td>02</td>
<td>03</td>
<td>02</td>
<td>03</td>
<td>14</td>
<td>03</td>
<td>13</td>
<td>03</td>
<td>04</td>
<td>13</td>
<td>14</td>
<td>04</td>
<td>13</td>
<td>14</td>
<td>04</td>
</tr>
<tr>
<td>01</td>
<td>02</td>
<td>03</td>
<td>14</td>
<td>14</td>
<td>13</td>
<td>14</td>
<td>13</td>
<td>14</td>
<td>14</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>
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)
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)
Theorems

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 $\text{tw}(G) = k$. 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-Isomorphie, Is-A-Disk-Graph-Problem,
Theorems

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 graphs with $\text{tw}(G) = k$. 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-Isomorphie, Is-A-Disk-Graph-Problem,
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 $\text{tw}(G) = k$. 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-Isomorphie, Is-A-Disk-Graph-Problem,
Theorem

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

- The problem, to compute the treewidth of a graph, is \( \text{NP-complete} \).
- 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 graphs with \( \text{tw}(G) = k \). 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-Isomorphie, 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
- $f(t) = f(s)$ holds if there is a second successor of $s$.
- $|f(t) \oplus f(s)| = 1$ if there is no second successor of $s$:
  - $f(s) = add(f(t), x)$ and
  - $f(s) = del(f(t), x)$
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
  - $f(s) = \text{del}(f(t), x)$
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
  - $f(s) = \text{del}(f(t), x)$
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
  - $f(s) = \text{del}(f(t), x)$
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 $s$:
  - $f(s) = add(f(t), x)$ and
  - $f(s) = del(f(t), x)$
Example (Independent Set)

Thus we only have to define the following:

- **What we store for** \( f(p_i) \): \( D(f(p_i)) \).
- 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)

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)

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 $f(p_{i+1})$.
- The procedure $D(f(s)) := Join(D(f(t)), D(f(t')))$. 


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)

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)

- \( D(f(s)) = \{(l, w) \mid l \text{ is IS on } f(s) \land w = value(l, s)\} \) with:
  - \( value(l, s) = \max\{|I'| \mid I' \subseteq \bigcup_{t \in V(T_s)} f(t) \land I' \text{ is IS } \land l \subseteq I'\} \) and
  - \( T_s \) is the subtree with root \( s \).
- \( Init(f(t)) = \{(l, w) \mid l \text{ is IS on } f(t) \land w = |l|\} \), compute all IS \( l \subseteq f(t) \) and set \( w = |l| \).
- \( (l, w) \in Del(D(f(t)), x) \) iff:
  - \( (l \cup \{x\}, w') \in D(f(p_i)) \) or \( (l, w'') \in D(f(p_i)) \) and
  - \( w = \max\{w' \mid (l \cup \{x\}, w') \in D(f(p_i)) \text{ or } (l, w') \in D(f(p_i))\} \).
- \( (l, w) \in Add(D(f(t)), x) \) iff:
  - \( (l, w) \in D(f(t)) \) or
  - \( (l \setminus \{x\}, w - 1) \in D(f(t)) \) and \( l \) is IS.
- \( (l, w) \in Join(D(f(t)), D(f(t'))) \) 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:

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

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

**Theorem**
Let $G = (V, E)$ be a graph. Then $pw(G) \leq vc(G)$ holds.
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 $pw(G) \leq vc(G)$ holds.
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 \( pw(G) \leq vc(G) \) holds.
**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 $P = (\{p_{k+1}, p_{k+2}, \ldots, p_n\}, \\{\{p_j, p_{j+1}\} \mid k + 1 \leq j < n\})$ 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:
  - $|f(p_j)| \leq \text{vc}(G) + 1$ 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.
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 $P = (\{p_{k+1}, p_{k+2}, \ldots, p_n\}, \{\{p_j, p_{j+1}\} \mid k + 1 \leq j < n\})$ 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:
  - $|f(p_j)| \leq \text{vc}(G) + 1$ 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.
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 $P = (\{p_{k+1}, p_{k+2}, \ldots, p_n\}, \{\{p_j, p_{j+1}\} | k + 1 \leq j < n\})$ 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 $\{v_c, v_j\} \subseteq C$, then we get $\{v_c, v_j\} \subseteq f(p_n)$.
  - $\{v_c, v_j\} \in E$ and $\{v_c, v_j\} \cap C = v_c$, then we get $\{v_c, v_j\} \subseteq f(p_j)$.
- Thus $\text{pw}(G) \leq \text{vc}(G)$ holds.
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.
- 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 $\{v_c, v_j\} \subset f(p_j)$.
- Thus $pw(G) \leq vc(G)$ holds.
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 \( P = (\{p_{k+1}, p_{k+2}, \ldots, p_n\}, \{\{p_j, p_{j+1}\} \mid k + 1 \leq j < n\}) \) 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:
  - \( |f(p_j)| \leq \text{vc}(G) + 1 \) 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.
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 $\{v_c, v_j\} \subset f(p_j)$.
- Thus $pw(G) \leq vc(G)$ holds.
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 \( P = (\{p_{k+1}, p_{k+2}, \ldots, p_n\}, \{\{p_j, p_{j+1}\} \mid k + 1 \leq j < n\}) \) 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:
  - \( |f(p_j)| \leq \text{vc}(G) + 1 \) 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.
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.
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 \( P = (\{p_{k+1}, p_{k+2}, \ldots, p_n\}, \{\{p_j, p_{j+1}\} \mid k+1 \leq j < n\}) \) 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:
  - \( |f(p_j)| \leq \text{vc}(G) + 1 \) 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)
Idea (1-Tree)
Idea (1-Tree)
Idea (1-Tree)

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

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

\[ \Sigma = 0 \]
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 $C = \{c_1, c_2, ..., c_k\}$ 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

**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.

**Note**

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

Note

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

Note

A $k$-tree is chordal and perfect.
Theorems II

**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 partial $k$-tree, iff $tw(G) \leq k$. 
Theorems II

**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.
- 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.
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.
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.
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.
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.
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.
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.
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.
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.
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.**
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: $\text{tw}(G) \leq k + 1$
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
Idea Cactus

\[ \sum = 0 \]
Idea Cactus
Idea Cactus
Idea Cactus

\[
\Sigma = a_0 c_0 e_0 a_1 c_1 e_1 r_1 c_2 r_2 e_3
\]
Idea Cactus

Graph representation of a cactus graph with nodes labeled as follows:
- c0, c1, c2, e0, e1, e3, a0, a1, r1, r2

Diagram shows connections with red edges and a tree-like structure.
Idea Cactus
Idea Cactus
Idea Cactus

**Diagram 1:**
- Node `c2` connects to `a1`, `c1`, `e1`, and `r1`.
- Node `a0` connects to `c0` and `e0`.
- Node `e3` is connected to a path from `c2`.

**Diagram 2:**
- Node `c2` connects to `a1c2a0`, `c1c2c0`, and `e1c2e0`.
- Node `r2` connects to `c2e3` and `r2e3`.
- Node `e3` is connected to a path from `c2`.

**Mathematical Notation:**
\[ \Sigma = \sum \]

**Additional Text:**
- Cactus and near-trees
- Walter Unger 15.6.2016 18:26
Idea Cactus

- Bandwidth: 
- Pathwidth: 
- Treewidth: 
- k-Trees: 
- Applications: 

4:48 Cactus and near-trees 11/14

Walter Unger 15.6.2016 18:26

SS2016
Idea Cactus

\[
\begin{align*}
\sum &= 0 \\
\end{align*}
\]
Idea Cactus
Idea 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 $f(y) = f(y) \cup \{b\}$.
- We have for each node $z$: $|f(z)| \leq 3$. 
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 $f(x) = \{v\}$.
- 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 $z$: $|f(z)| \leq 3$. 
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 $\text{deg}(v) > 3$ 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 $z$: $|f(z)| \leq 3$. 
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 \( z \): \( |f(z)| \leq 3 \).
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 $\text{deg}(v) > 3$ 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 $z$: $|f(z)| \leq 3$. 
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 $\text{deg}(v) > 3$ 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 $z$: $|f(z)| \leq 3$. 
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 $z$: $|f(z)| \leq 3$. 
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 $z$: $|f(z)| \leq 3$. 
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 $z$: $|f(z)| \leq 3$. 

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 $\text{deg}(v) > 3$ 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 $z$: $|f(z)| \leq 3$. 
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 $\text{deg}(v) > 3$ 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 $z$: $|f(z)| \leq 3$. 
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 $z$: $|f(z)| \leq 3$. 
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 $z$: $|f(z)| \leq 3$. 
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 $\text{deg}(v) > 3$ 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
Idea near-tree
Idea near-tree
Idea near-tree
Idea near-tree
Idea near-tree
Idea near-tree
Idea near-tree
Idea near-tree
Idea near-tree

\[ \Sigma = \ldots \]
Idea near-tree
Idea near-tree
Idea near-tree
Proof (near-tree)

- Let $G = (V, E)$ be a near-tree($k$) with $V = \{v_1, v_2, v_3, \ldots, v_n\}$.
- 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 $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
  - 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 $z$: $|f(z)| \leq 2 + k$. 
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 $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
  - 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 $z$: $|f(z)| \leq 2 + k$. 
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 $z$: $|f(z)| \leq 2 + k$. 
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 $z$: $|f(z)| \leq 2 + k$. 
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 $z$: $|f(z)| \leq 2 + k$. 

Proof completed.
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 $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
  - 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 $z$: $|f(z)| \leq 2 + k$. 
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 $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
    - 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 $z$: $|f(z)| \leq 2 + k$. 
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 $e_{ij}^i$.

Then is $T = (V, E \setminus \{e_{ij}^i \mid 1 \leq i \leq d \land 1 \leq j \leq d_i\})$ 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 $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
- For each edge $e_{ij}^i = \{a, b\}$ and for each node $y$ on the path from $a$ to $b$ define $f(y) = f(y) \cup \{a\}$.

We have for each node $z$: $|f(z)| \leq 2 + k$. 
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 $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
  - 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 $z$: $|f(z)| \leq 2 + k$. 
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 $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
- 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 $z$: $|f(z)| \leq 2 + k$. 
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 $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
  - 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 $z$: $|f(z)| \leq 2 + k$. 
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 $\{a, v_{a,b}\}, \{v_{a,b}, b\}$.
  - 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: $tw(G) \leq 3$*

**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: $tw(G) \leq 2$*
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: $\text{tw}(G) \leq 3$

**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

\[ \Sigma = \]

\[
\begin{array}{c}
\text{a0} \\
\text{c0} \\
\text{e0} \\
\text{a1} \\
\text{c1} \\
\text{e1} \\
\text{c2} \\
\text{r1} \\
\text{e3} \\
\text{r2}
\end{array}
\]
Idea Halin

\[ \Sigma = a_0 \quad c_0 \quad e_0 \quad a_1 \quad c_1 \quad e_1 \quad r_1 \quad c_2 \quad r_2 \quad e_3 \]
Idea Halin
Idea Halin

Diagram of Halin graphs and outer-planar graphs.
Idea Halin
Idea Halin

Hallin-graphs and outer-planar graphs

\[ \Sigma = \]

\[ a_0 \quad a_1 \quad c_0 \quad c_1 \quad e_0 \quad e_1 \quad r_1 \quad r_2 \]

\[ c_2 \quad e_3 \]

\[ a_{0a1} \quad c_{0c1} \quad e_{0e1} \quad a_1 \quad c_1 \quad e_1 \quad r_{1c2} \quad r_2 \]

\[ c_{2e3} \quad r_{2e3} \]

\[ \]
Idea Halin
Idea Halin
Idea Halin

\[ \Sigma = \]
Idea Halin
Idea Halin

\[
\Sigma = \begin{align*}
\text{a0} & \rightarrow \text{c0} & \rightarrow \text{e0} \\
\text{a1} & \rightarrow \text{c1} & \rightarrow \text{e1} \\
\text{c2} & \rightarrow \text{r2} \\
\text{e3} & \rightarrow \text{r2}
\end{align*}
\]
Idea Halin
Idea Halin
Idea Halin
Idea Halin
Idea Halin

\[ \Sigma = 0 \]
Idea Halin
Idea Halin
Idea Halin
Idea Halin

\[ \Sigma = \]
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 $z$: $|f(z)| \leq 4$. 
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 $z$: $|f(z)| \leq 4$. 
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 $\text{deg}(v) > 3$ 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 $z$: $|f(z)| \leq 4$. 
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 $z$: $|f(z)| \leq 4$. 
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 $z$: $|f(z)| \leq 4$. 
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 $z$: $|f(z)| \leq 4$. 
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 $z$: $|f(z)| \leq 4$. 
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 $z$: $|f(z)| \leq 4$. 
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 $\text{deg}(v) > 3$ 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 $z$: $|f(z)| \leq 4$. 
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 $\text{deg}(v) > 3$ 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 $z$: $|f(z)| \leq 4$. 
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 $z$: $|f(z)| \leq 4$. 

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
Idea outer-planar Graph
Idea 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.
- 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 $f(x) = V(x)$. 
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 $f(x) = V(x)$. 

Proof II (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.
- 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 $x$.
- Define tree $(V', \{\{a, b\} \mid E(a) \cap E(b) \neq \emptyset\})$
- Define $f$ by $f(x) = V(x)$. 
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 $(V', \{\{a, b\} \mid E(a) \cap E(b) \neq \emptyset\})$
- Define $f$ by $f(x) = V(x)$. 
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 $(V', \{\{a, b\} \mid E(a) \cap E(b) \neq \emptyset\})$
- Define $f$ by $f(x) = V(x)$. 
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 $x$.
- Define tree $(V', \{\{a, b\} \mid E(a) \cap E(b) \neq \emptyset\})$
- Define $f$ by $f(x) = V(x)$. 
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 $(V', \{\{a, b\} \mid E(a) \cap E(b) \neq \emptyset\})$
- Define $f$ by $f(x) = V(x)$. 
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 $(V', \{\{a, b\} \mid E(a) \cap E(b) \neq \emptyset\})$
- Define $f$ by $f(x) = V(x)$. 

\[ \]
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 \( (V', \{\{a, b\} \mid E(a) \cap E(b) \neq \emptyset\}) \)
- 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: $\text{tw}(G) \leq 3 \cdot k - 1$
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.
- 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.
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.
- 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.
**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.
- 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.
**Theorems VI**

**Definition**

A graph $G = (V, E)$ is called a 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.
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.
- 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.

Theorem

A SP-graph has treewidth 2.
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.
- 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.
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:
- $G' = (V \setminus b, (E \setminus \{\{v, b\} \mid v \in V\}) \cup \{\{v, a\} \mid \{v, b\} \in E\})$
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:
- $G' = (V \setminus b, (E \setminus \{\{v, b\} \mid v \in V\}) \cup \{\{v, a\} \mid \{v, b\} \in E\})$
**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:
  
  $$G' = (V \setminus b, (E \setminus \{\{v, b\} \mid v \in V\}) \cup \{\{v, a\} \mid \{v, b\} \in E\})$$
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:
  \[
  G' = (V \setminus b, (E \setminus \{\{v, b\} \mid v \in V\}) \cup \{\{v, a\} \mid \{v, b\} \in E\})
  \]
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 $\{a, b\} \in E$,
- Then the node-merging of $a$ and $b$ is possible:

\[
G' = (V \setminus b, (E \setminus \{\{v, b\} \mid v \in V\}) \cup \{\{v, a\} \mid \{v, b\} \in E\})
\]
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 $MS_2$ on a graph $G$ with $\text{tw}(G) \leq k$ is solvable in polynomial time.
Theorems VII

Theorem

A $G$ 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 $MS_2$ on a graph $G$ with $\text{tw}(G) \leq k$ is solvable in polynomial time.
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 $MS_2$ on a graph $G$ with $\text{tw}(G) \leq k$ is solvable in polynomial time.
Questions

1. What is the definition of bandwidth?
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. 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. 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. 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. 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. 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. 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. 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. 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. 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.
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?
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?
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?
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?
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?
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?
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?
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 graph?
8. What is the treewidth of a $k$-outer-planar graph?
9. What is the treewidth of a SP-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?
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?
Legend

- Not of relevance
- implicitly used basics
- idea of proof or algorithm
- structure of proof or algorithm
- Full knowledge