

# Algorithmen II

## Vorlesung am 23.01.2014

Parametrisierte Algorithmen

INSTITUT FÜR THEORETISCHE INFORMATIK · PROF. DR. DOROTHEA WAGNER



# Parametrisierte Algorithmen – Einführung

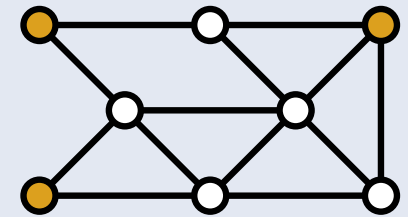


# Drei Probleme

**Gegeben:** Ein Graph  $G = (V, E)$ , sowie ein Parameter  $k \in \mathbb{N}$ .

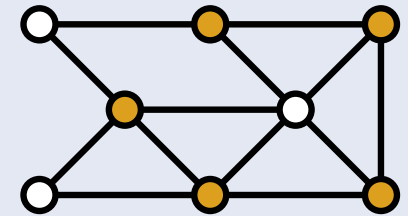
## Problem: INDEPENDENT SET

Finde *unabhängige Menge*  $V' \subseteq V$  mit  $|V'| \geq k$ .  $V'$  heißt unabhängig genau dann, wenn für alle  $v, w \in V'$  gilt  $\{v, w\} \notin E$ .



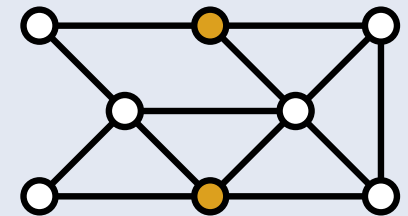
## Problem: VERTEX COVER

Finde *Vertex Cover*  $V' \subseteq V$  mit  $|V'| \leq k$ .  $V'$  heißt Vertex Cover genau dann, wenn für jede Kante  $\{v, w\} \in E$  gilt  $v \in V'$  oder  $w \in V'$ .



## Problem: DOMINATING SET

Finde *Dominating Set*  $V' \subseteq V$  mit  $|V'| \leq k$ .  $V'$  heißt Dominating Set genau dann, wenn für jeden Knoten  $v \in V$  gilt, dass  $v$  oder einer seiner Nachbarn in  $V'$  enthalten ist.



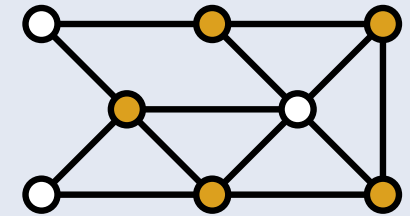
INDEPENDENT SET, VERTEX COVER und DOMINATING SET sind  $\mathcal{NP}$ -schwer.

→ Aufzählung aller  $\binom{n}{k}$  Teilmengen (Brute-Force)  $V'$  mit  $|V'| = k$  liefert Algorithmus mit Laufzeit  $O(n^k \cdot (n + m))$ . Für konstantes  $k$  polynomiell! Geht es noch besser?

# Algorithmus für VERTEX COVER

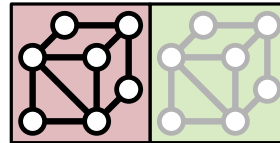
## Problem: VERTEX COVER

Finde *Vertex Cover*  $V' \subseteq V$  mit  $|V'| \leq k$ .  $V'$  heißt Vertex Cover genau dann, wenn für jede Kante  $\{v, w\} \in E$  gilt  $v \in V'$  oder  $w \in V'$ .



noch zu überdeckender Teilgraph

ausgewählte Knoten & überdeckte Kanten

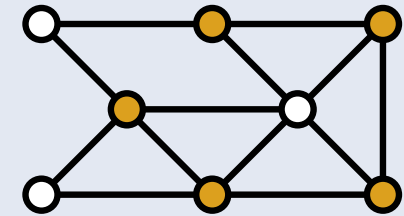


Gibt es ein VERTEX COVER mit maximal 3 Knoten?

# Algorithmus für VERTEX COVER

## Problem: VERTEX COVER

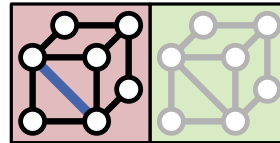
Finde *Vertex Cover*  $V' \subseteq V$  mit  $|V'| \leq k$ .  $V'$  heißt Vertex Cover genau dann, wenn für jede Kante  $\{v, w\} \in E$  gilt  $v \in V'$  oder  $w \in V'$ .



noch zu überdeckender Teilgraph

ausgewählte Knoten & überdeckte Kanten

Jede Kante muss noch überdeckt werden. → wähle eine beliebige.

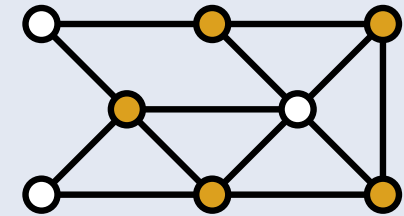


Gibt es ein VERTEX COVER mit maximal 3 Knoten?

# Algorithmus für VERTEX COVER

## Problem: VERTEX COVER

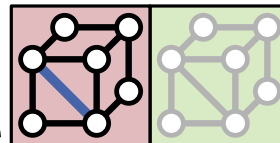
Finde *Vertex Cover*  $V' \subseteq V$  mit  $|V'| \leq k$ .  $V'$  heißt Vertex Cover genau dann, wenn für jede Kante  $\{v, w\} \in E$  gilt  $v \in V'$  oder  $w \in V'$ .



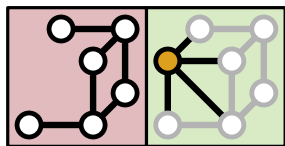
noch zu überdeckender Teilgraph

ausgewählte Knoten & überdeckte Kanten

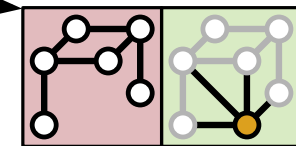
Jede Kante muss noch überdeckt werden.  $\rightarrow$  wähle eine beliebige.



Gibt es ein VERTEX COVER mit maximal 3 Knoten?



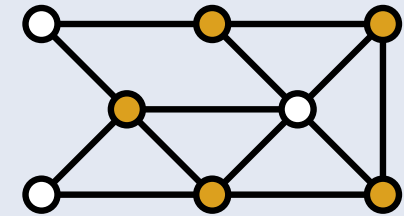
Für die Kante  $\{v, w\}$  muss  $v$  oder  $w$  enthalten sein.  
 $\rightarrow$  binärer Entscheidungsbaum



# Algorithmus für VERTEX COVER

## Problem: VERTEX COVER

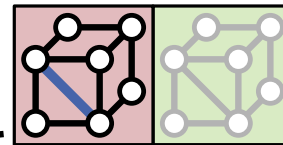
Finde *Vertex Cover*  $V' \subseteq V$  mit  $|V'| \leq k$ .  $V'$  heißt Vertex Cover genau dann, wenn für jede Kante  $\{v, w\} \in E$  gilt  $v \in V'$  oder  $w \in V'$ .



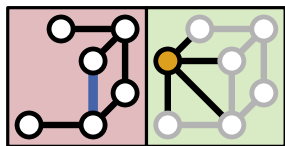
noch zu überdeckender Teilgraph

ausgewählte Knoten & überdeckte Kanten

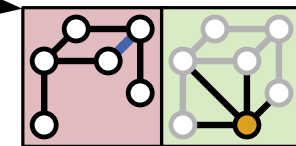
Jede Kante muss noch überdeckt werden.  $\rightarrow$  wähle eine beliebige.



Gibt es ein VERTEX COVER mit maximal 3 Knoten?



Für die Kante  $\{v, w\}$  muss  $v$  oder  $w$  enthalten sein.  
 $\rightarrow$  binärer Entscheidungsbaum

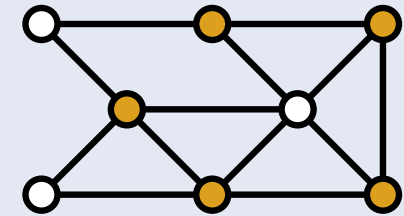




# Algorithmus für VERTEX COVER

## Problem: VERTEX COVER

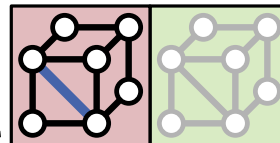
Finde *Vertex Cover*  $V' \subseteq V$  mit  $|V'| \leq k$ .  $V'$  heißt Vertex Cover genau dann, wenn für jede Kante  $\{v, w\} \in E$  gilt  $v \in V'$  oder  $w \in V'$ .



noch zu überdeckender Teilgraph

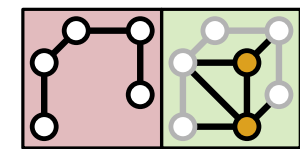
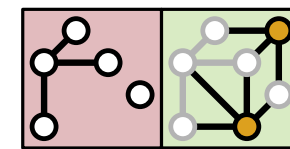
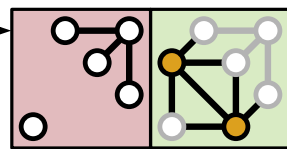
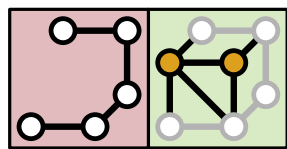
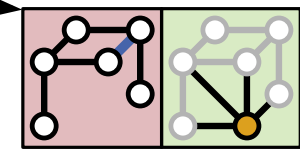
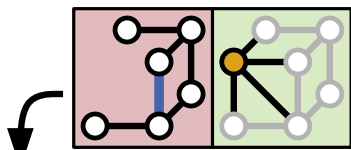
ausgewählte Knoten & überdeckte Kanten

Jede Kante muss noch überdeckt werden. → wähle eine beliebige.



Gibt es ein VERTEX COVER mit maximal 3 Knoten?

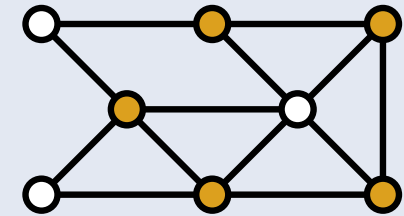
Für die Kante  $\{v, w\}$  muss  $v$  oder  $w$  enthalten sein.  
→ binärer Entscheidungsbaum



# Algorithmus für VERTEX COVER

## Problem: VERTEX COVER

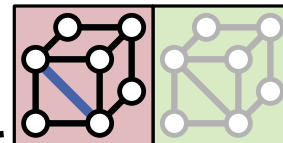
Finde *Vertex Cover*  $V' \subseteq V$  mit  $|V'| \leq k$ .  $V'$  heißt Vertex Cover genau dann, wenn für jede Kante  $\{v, w\} \in E$  gilt  $v \in V'$  oder  $w \in V'$ .



noch zu überdeckender Teilgraph

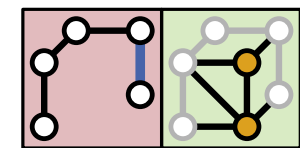
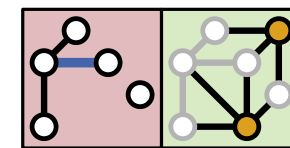
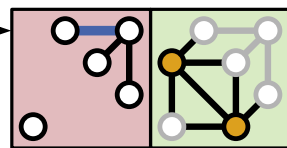
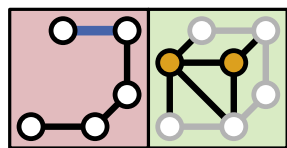
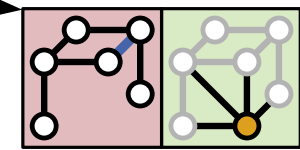
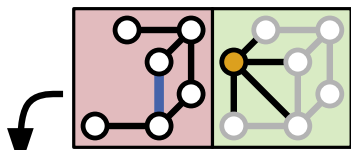
ausgewählte Knoten & überdeckte Kanten

Jede Kante muss noch überdeckt werden. → wähle eine beliebige.



Gibt es ein VERTEX COVER mit maximal 3 Knoten?

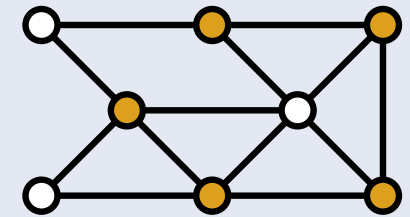
Für die Kante  $\{v, w\}$  muss  $v$  oder  $w$  enthalten sein.  
→ binärer Entscheidungsbaum



# Algorithmus für VERTEX COVER

## Problem: VERTEX COVER

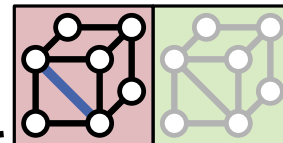
Finde *Vertex Cover*  $V' \subseteq V$  mit  $|V'| \leq k$ .  $V'$  heißt Vertex Cover genau dann, wenn für jede Kante  $\{v, w\} \in E$  gilt  $v \in V'$  oder  $w \in V'$ .



noch zu überdeckender Teilgraph

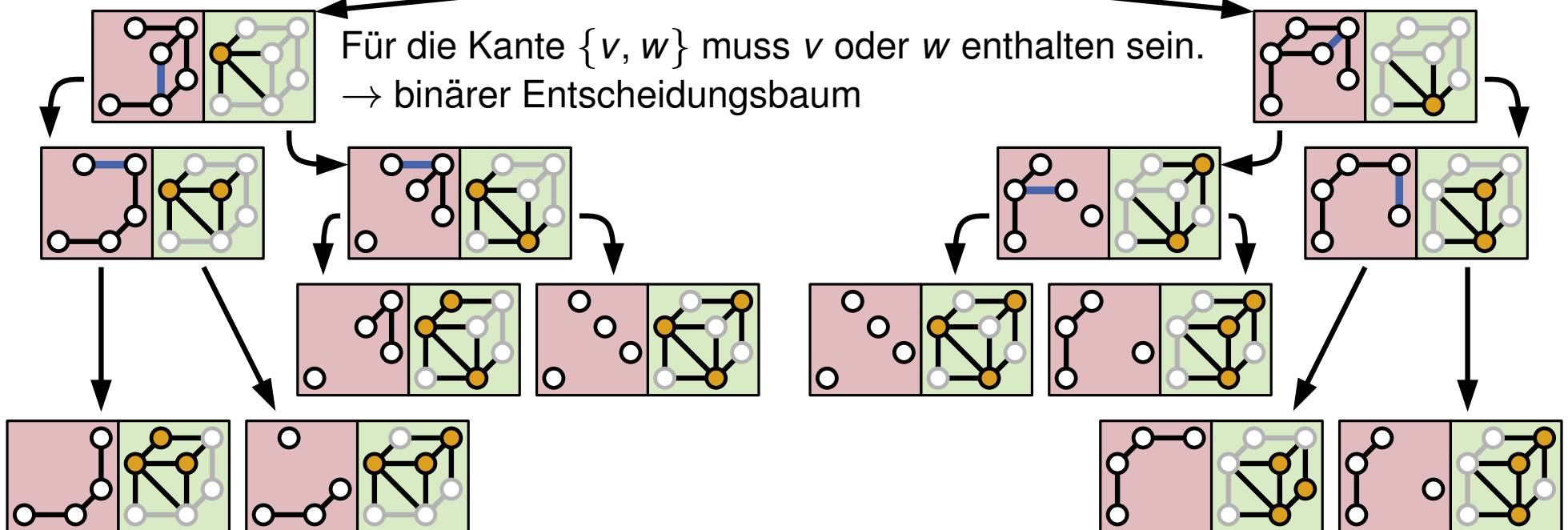
ausgewählte Knoten & überdeckte Kanten

Jede Kante muss noch überdeckt werden. → wähle eine beliebige.



Gibt es ein VERTEX COVER mit maximal 3 Knoten?

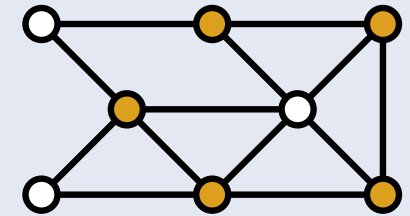
Für die Kante  $\{v, w\}$  muss  $v$  oder  $w$  enthalten sein.  
→ binärer Entscheidungsbaum



# Algorithmus für VERTEX COVER

## Problem: VERTEX COVER

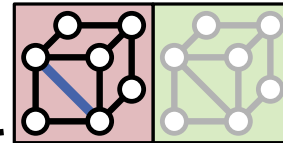
Finde *Vertex Cover*  $V' \subseteq V$  mit  $|V'| \leq k$ .  $V'$  heißt Vertex Cover genau dann, wenn für jede Kante  $\{v, w\} \in E$  gilt  $v \in V'$  oder  $w \in V'$ .



noch zu überdeckender Teilgraph

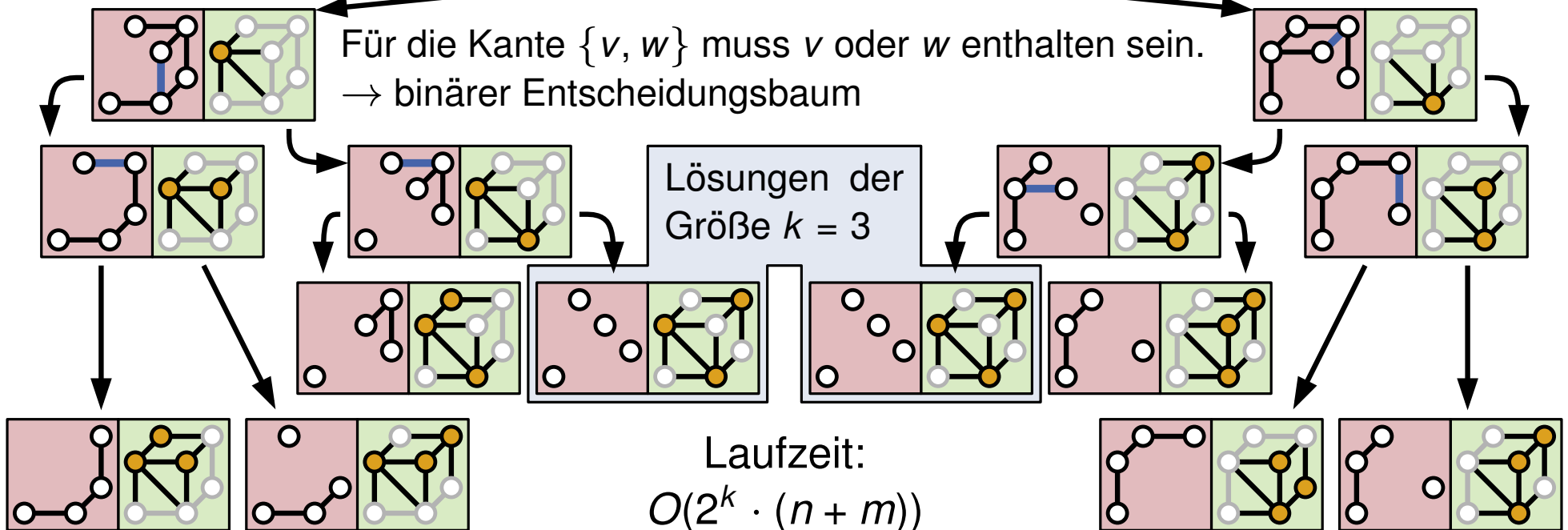
ausgewählte Knoten & überdeckte Kanten

Jede Kante muss noch überdeckt werden. → wähle eine beliebige.



Gibt es ein VERTEX COVER mit maximal 3 Knoten?

Für die Kante  $\{v, w\}$  muss  $v$  oder  $w$  enthalten sein.  
→ binärer Entscheidungsbaum



# Fixed Parameter Tractable (FPT)

- Der Brute-Force Ansatz hat Laufzeit  $O(n^k \cdot (n + m))$ .
  - Polynomielle Laufzeit für konstantes  $k$ .
- Der Algorithmus für VERTEX COVER benötigt  $O(2^k \cdot (n + m))$  Zeit.
  - Polynomielle Laufzeit für konstantes  $k$  und
  - Asymptotische Laufzeit hängt nicht von  $k$  ab, falls  $k$  konstant.

# Fixed Parameter Tractable (FPT)

- Der Brute-Force Ansatz hat Laufzeit  $O(n^k \cdot (n + m))$ .
  - Polynomielle Laufzeit für konstantes  $k$ .
- Der Algorithmus für VERTEX COVER benötigt  $O(2^k \cdot (n + m))$  Zeit.
  - Polynomielle Laufzeit für konstantes  $k$  und
  - Asymptotische Laufzeit hängt nicht von  $k$  ab, falls  $k$  konstant.

## Definition: Fixed Parameter Tractable

(Definition 10.1)

Ein parametrisiertes Problem  $\Pi$  heißt *fixed parameter tractable*, wenn es in  $O(\mathcal{C}(k) \cdot p(n))$  gelöst werden kann. Dabei ist  $n$  die Eingabegröße,  $p$  ein Polynom,  $k$  der Parameter und  $\mathcal{C}$  eine berechenbare Funktion, die nur von  $k$  abhängt.

## Definition: Komplexitätsklasse FPT

(Definition 10.2)

FPT ist die Klasse aller Problem, die fixed parameter tractable sind.

## Bemerkung:

(Bemerkung 10.3)

- VERTEX COVER ist in FPT.
- Es ist unbekannt ob INDEPENDENT SET oder DOMINATING SET in FPT sind. Man vermutet, dass sie nicht in FPT sind.

## Definition: Parametrisierte Reduktion

(Definition 10.4)

Seien  $\Pi$  und  $\Pi'$  parametrisierte Probleme. Eine *parametrisierte Reduktion* von  $\Pi$  auf  $\Pi'$  besteht aus einer Funktion  $f$ , die einer Instanz  $\mathcal{I}$  mit Parameter  $k$  von  $\Pi$  eine Instanz  $\mathcal{I}'$  von  $\Pi'$  zuordnet, sowie Funktionen  $C', C'' : \mathbb{N} \rightarrow \mathbb{N}$ , sodass gilt:

- $\mathcal{I}' = f(\mathcal{I}, k)$  kann in  $O(C''(k) \cdot p(n))$  berechnet werden ( $n$  ist Eingabegröße von  $\mathcal{I}$  und  $p$  ein Polynom).
- $(\mathcal{I}, k)$  ist Ja-Instanz von  $\Pi$  genau dann wenn  $(\mathcal{I}', C'(k))$  ist Ja-Instanz von  $\Pi'$ .

## Definition: Parametrisierte Reduktion

(Definition 10.4)

Seien  $\Pi$  und  $\Pi'$  parametrisierte Probleme. Eine *parametrisierte Reduktion* von  $\Pi$  auf  $\Pi'$  besteht aus einer Funktion  $f$ , die einer Instanz  $\mathcal{I}$  mit Parameter  $k$  von  $\Pi$  eine Instanz  $\mathcal{I}'$  von  $\Pi'$  zuordnet, sowie Funktionen  $C', C'' : \mathbb{N} \rightarrow \mathbb{N}$ , sodass gilt:

- $\mathcal{I}' = f(\mathcal{I}, k)$  kann in  $O(C''(k) \cdot p(n))$  berechnet werden ( $n$  ist Eingabegröße von  $\mathcal{I}$  und  $p$  ein Polynom).
- $(\mathcal{I}, k)$  ist Ja-Instanz von  $\Pi$  genau dann wenn  $(\mathcal{I}', C'(k))$  ist Ja-Instanz von  $\Pi'$ .

## Bemerkung: Vergleich zur polynomiellen Reduktion

- Ist  $\Pi$  **polynomiell** auf  $\Pi'$  reduzierbar, so folgt aus  $\Pi' \in \mathcal{P}$  auch  $\Pi \in \mathcal{P}$ .  
→  $\Pi$  ist unter Vernachlässigung **polynomieller** Laufzeit nicht schwerer als  $\Pi'$ .
- Ist  $\Pi$  **parametrisiert** auf  $\Pi'$  reduzierbar, so folgt aus  $\Pi' \in \text{FPT}$  auch  $\Pi \in \text{FPT}$ .  
→  $\Pi$  ist unter Vernachlässigung von  $O(C(k) \cdot p(n))$  Laufzeit nicht schwerer als  $\Pi'$ .



## Definition: Parametrisierte Reduktion

(Definition 10.4)

Seien  $\Pi$  und  $\Pi'$  parametrisierte Probleme. Eine *parametrisierte Reduktion* von  $\Pi$  auf  $\Pi'$  besteht aus einer Funktion  $f$ , die einer Instanz  $\mathcal{I}$  mit Parameter  $k$  von  $\Pi$  eine Instanz  $\mathcal{I}'$  von  $\Pi'$  zuordnet, sowie Funktionen  $C', C'' : \mathbb{N} \rightarrow \mathbb{N}$ , sodass gilt:

- $\mathcal{I}' = f(\mathcal{I}, k)$  kann in  $O(C''(k) \cdot p(n))$  berechnet werden ( $n$  ist Eingabegröße von  $\mathcal{I}$  und  $p$  ein Polynom).
- $(\mathcal{I}, k)$  ist Ja-Instanz von  $\Pi$  genau dann wenn  $(\mathcal{I}', C'(k))$  ist Ja-Instanz von  $\Pi'$ .

## Bemerkung: Hierarchie von Komplexitätsklassen

(Bemerkung 10.5)

- Es gibt eine Hierarchie von Komplexitätsklassen  $W[t]$ , die vermutlich echt ist und FPT enthält:  $\text{FPT} \subseteq W[1] \subseteq W[2] \dots$
- Mithilfe der parametrisierten Reduktion lässt sich ein Vollständigkeitsbegriff definieren, um die schweren Probleme in  $W[t]$  zu identifizieren.
- INDEPENDENT SET ist  $W[1]$ -vollständig, DOMINATING SET ist  $W[2]$ -vollständig.

# Parametrisierte Algorithmen – Grundtechniken

# Zwei Grundtechniken

- 1. Kernbildung:** Reduziere die Instanz durch Anwendung verschiedener Regeln auf einen (schweren) Problemerkern.
- 2. Tiefenbeschränkte Suchbäume:** Erschöpfende Suche in einem geeigneten Suchbaum mit beschränkter Tiefe. (zuvor für VERTEX COVER gesehen)

**1. Kernbildung:** Reduziere die Instanz durch Anwendung verschiedener Regeln auf einen (schweren) Problemerkern.

## Idee:

- Reduziere Instanz  $(\mathcal{I}, k)$  in  $O(p(|\mathcal{I}|))$  Zeit auf eine äquivalente Instanz  $\mathcal{I}'$ , sodass  $|\mathcal{I}'|$  nur von  $k$  (und nicht von  $|\mathcal{I}|$ ) abhängt.
- Löse  $\mathcal{I}'$  mit erschöpfender Suche.

**1. Kernbildung:** Reduziere die Instanz durch Anwendung verschiedener Regeln auf einen (schweren) Problemerkern.

## Idee:

- Reduziere Instanz  $(\mathcal{I}, k)$  in  $O(p(|\mathcal{I}|))$  Zeit auf eine äquivalente Instanz  $\mathcal{I}'$ , sodass  $|\mathcal{I}'|$  nur von  $k$  (und nicht von  $|\mathcal{I}|$ ) abhängt.
- Löse  $\mathcal{I}'$  mit erschöpfender Suche.

## Beispiel: VERTEX COVER

(Beispiel 10.6)

Um ein Vertex Cover  $V' \subseteq V$  für  $G = (V, E)$  mit  $|V'| \leq k$  zu bestimmen, kann man folgende Beobachtungen nutzen:

- Für  $v \in V$  liegt  $v$  selbst oder seine gesamte Nachbarschaft  $N(v)$  in  $V'$ .
- $V'$  enthält jeden Knoten  $v \in V$  mit  $\deg(v) > k$ .
- Falls  $\Delta(G) \leq k$  und  $|E| > k^2$ , so hat jedes Vertex Cover mehr als  $k$  Knoten. Dabei ist  $\Delta(G)$  der Maximalgrad von  $G$ .

# Kernbildung

## Beispiel: VERTEX COVER

(Beispiel 10.6)

Um ein Vertex Cover  $V' \subseteq V$  für  $G = (V, E)$  mit  $|V'| \leq k$  zu bestimmen, kann man folgende Beobachtungen nutzen:

- Für  $v \in V$  liegt  $v$  selbst oder seine gesamte Nachbarschaft  $N(v)$  in  $V'$ .
- $V'$  enthält jeden Knoten  $v \in V$  mit  $\deg(v) > k$ .
- Falls  $\Delta(G) \leq k$  und  $|E| > k^2$ , so hat jedes Vertex Cover mehr als  $k$  Knoten. Dabei ist  $\Delta(G)$  der Maximalgrad von  $G$ .

VERTEX COVER( $G, k$ )

$H \leftarrow \{v \in V \mid \deg(v) > k\}$

**if**  $|H| > k$  **then** Gib aus: „ $G$  hat kein Vertex Cover der Größe  $k$ “

**else**

$k' \leftarrow k - |H|$

$G' \leftarrow G - H$

**if**  $|E(G')| > k \cdot k'$  **then** Gib aus: „ $G$  hat kein Vertex Cover der Größe  $k$ “

**else**

        entferne alle isolierten Knoten aus  $G'$

        berechne ein Vertex Cover der Größe  $k'$  im verbleibenden Graphen

# Kernbildung

## Beispiel: VERTEX COVER

(Beispiel 10.6)

Um ein Vertex Cover  $V' \subseteq V$  für  $G = (V, E)$  mit  $|V'| \leq k$  zu bestimmen, kann man folgende Beobachtungen nutzen:

- Für  $v \in V$  liegt  $v$  selbst oder seine gesamte Nachbarschaft  $N(v)$  in  $V'$ .
- $V'$  enthält jeden Knoten  $v \in V$  mit  $\deg(v) > k$ .
- Falls  $\Delta(G) \leq k$  und  $|E| > k^2$ , so hat jedes Vertex Cover mehr als  $k$  Knoten. Dabei ist  $\Delta(G)$  der Maximalgrad von  $G$ .

VERTEX COVER( $G, k$ )	$O(2^k \cdot k^2 + n \cdot k)$
$H \leftarrow \{v \in V \mid \deg(v) > k\}$	$O(n \cdot k)$
<b>if</b> $ H  > k$ <b>then</b> Gib aus: „ $G$ hat kein Vertex Cover der Größe $k$ “	$O(1)$
<b>else</b>	
$k' \leftarrow k -  H $	$O(1)$
$G' \leftarrow G - H$	$O(n \cdot k)$
<b>if</b> $ E(G')  > k \cdot k'$ <b>then</b> Gib aus: „ $G$ hat kein Vertex Cover der Größe $k$ “	$O(k \cdot k')$
<b>else</b>	
entferne alle isolierten Knoten aus $G'$	$O(n)$
berechne ein Vertex Cover der Größe $k'$ im verbleibenden Graphen	$O(2^k \cdot k^2)$

# Tiefenbeschränkte Suchbäume

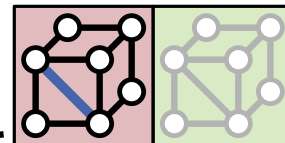
**2. Tiefenbeschränkte Suchbäume:** Erschöpfende Suche in einem geeigneten Suchbaum mit beschränkter Tiefe. (zuvor für VERTEX COVER gesehen)

## Erinnerung:

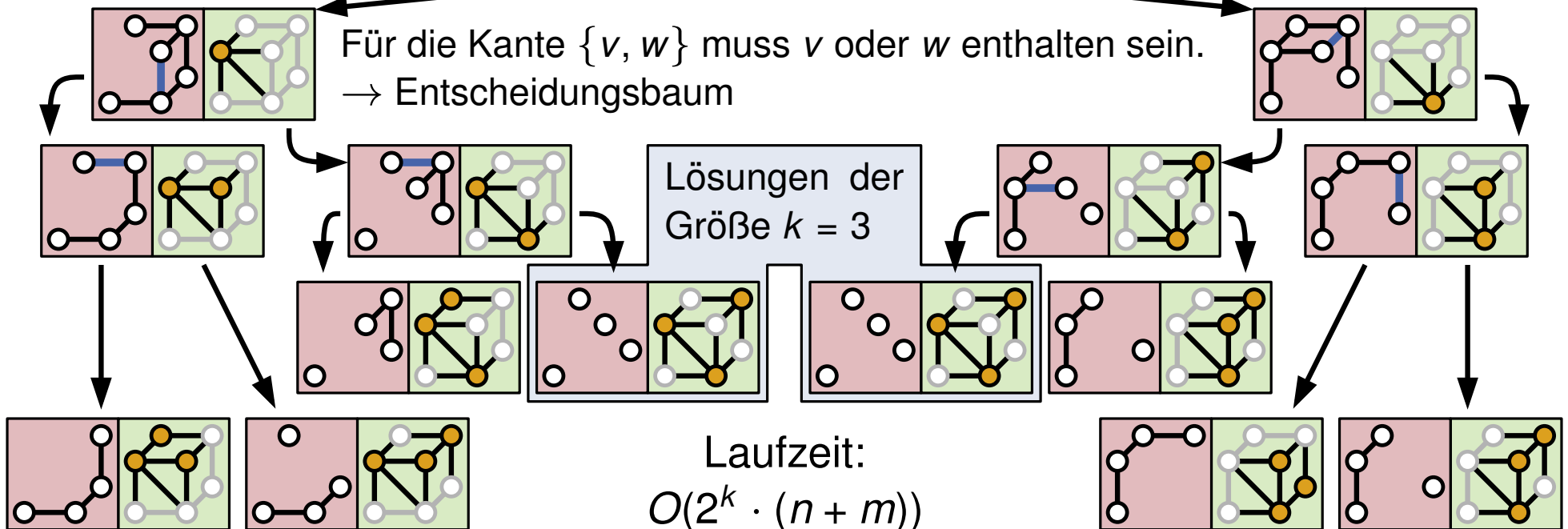
noch zu überdeckender Teilgraph

ausgewählte Knoten & überdeckte Kanten

Jede Kante muss noch überdeckt werden. → wähle eine beliebige.



Für die Kante  $\{v, w\}$  muss  $v$  oder  $w$  enthalten sein.  
→ Entscheidungsbaum



Kann der Faktor  $2^k$  verbessert werden?



**2. Tiefenbeschränkte Suchbäume:** Erschöpfende Suche in einem geeigneten Suchbaum mit beschränkter Tiefe. (zuvor für VERTEX COVER gesehen)

Laufzeit hängt wesentlich von der Struktur des Baumes ab.

**Binärbaum:**  $T(k) = T(k - 1) + T(k - 1) + c$  (wie im vorherigen Fall)

**Allgemein:**  $T(k) \leq T(k - t_1) + \dots + T(k - t_s) + c$

→ Der Vektor  $(t_1, \dots, t_s)$  heißt *Verzweigungsvektor*. (Für einen Binärbaum also  $(1, 1)$ )

Die Basis  $b$  im Term  $b^k$  hängt vom Verzweigungsvektor ab.

Verzweigungsvektor	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(3, 3)	(3, 3, 6)	(3, 4, 6)
Basis $b$	2	1.618	1.4656	1.3803	1.325	1.325	1.26	1.342	1.305

**2. Tiefenbeschränkte Suchbäume:** Erschöpfende Suche in einem geeigneten Suchbaum mit beschränkter Tiefe. (zuvor für VERTEX COVER gesehen)

Laufzeit hängt wesentlich von der Struktur des Baumes ab.

**Binärbaum:**  $T(k) = T(k - 1) + T(k - 1) + c$  (wie im vorherigen Fall)

**Allgemein:**  $T(k) \leq T(k - t_1) + \dots + T(k - t_s) + c$

→ Der Vektor  $(t_1, \dots, t_s)$  heißt *Verzweigungsvektor*. (Für einen Binärbaum also  $(1, 1)$ )

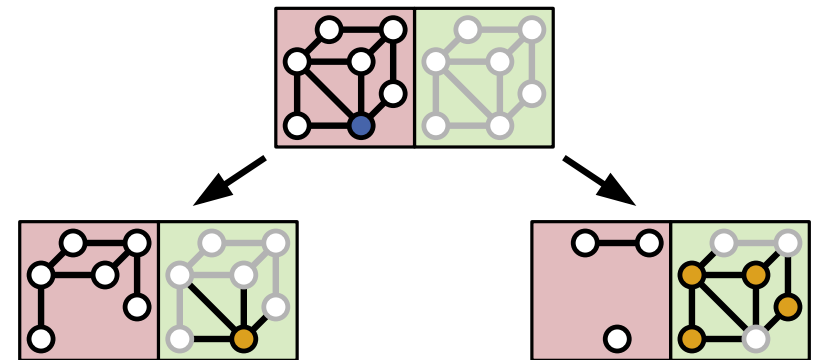
Die Basis  $b$  im Term  $b^k$  hängt vom Verzweigungsvektor ab.

Verzweigungsvektor	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(3, 3)	(3, 3, 6)	(3, 4, 6)
Basis $b$	2	1.618	1.4656	1.3803	1.325	1.325	1.26	1.342	1.305

## Idee für VERTEX COVER:

- Annahme: Es gibt immer einen Knoten  $v$  mit  $\text{deg}(v) \geq 4$ .
- Wähle in jedem Schritt entweder  $v$  oder alle seine Nachbarn.
- Verzweigungsvektor:  $(1, 4)$ .

Ergibt Laufzeit  $O(kn + 1.3803^k \cdot k^2)$

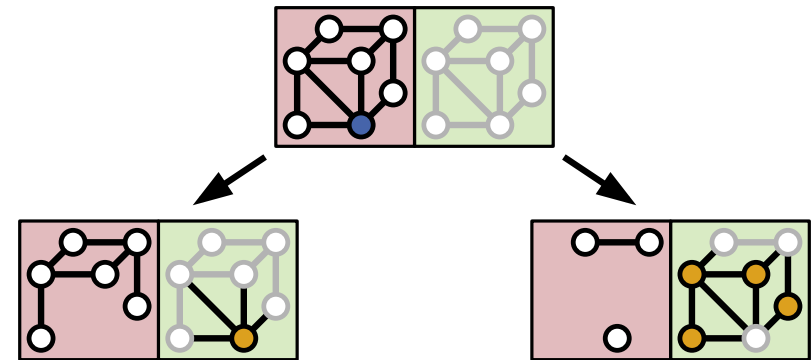


# Tiefenbeschränkte Suchbäume

Verzweigungsvektor	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(3, 3)	(3, 3, 6)	(3, 4, 6)
Basis $b$	2	1.618	1.4656	1.3803	1.325	1.325	1.26	1.342	1.305

## Idee für VERTEX COVER:

- Annahme: Es gibt immer einen Knoten  $v$  mit  $\deg(v) \geq 4$  gibt.
- Wähle in jedem Schritt entweder  $v$  oder alle seine Nachbarn.
- Verzweigungsvektor: (1, 4).  
Ergibt Laufzeit  $O(kn + 1.3803^k \cdot k^2)$



**Problem:** Man findet nicht immer einen Knoten mit Mindestgrad 4.

**Idee:** Verzweige an jedem Knoten je nach eintretendem Fall mit Verzweigungsvektor (1, 5), (2, 3), (3, 3), (3, 4, 6) oder (3, 3, 6) (oder höchstens einmal (1, 4)).  
 $\Rightarrow$  Laufzeit wird dominiert durch (3, 3, 6)  $\Rightarrow$  Gesamtlaufzeit  $O(kn + 1.342^k \cdot k^2)$ .

## Im Folgenden:

- Menge von Regeln zur Verzweigung (sodass immer eine der Regeln anwendbar ist).
- Erlaubte Verzweigungsvektoren:  $(1, 5)$ ,  $(2, 3)$ ,  $(3, 3)$ ,  $(3, 4, 6)$ ,  $(3, 3, 6)$ , höchstens einmal  $(1, 4)$
- Wende bei jeder Verzweigung die Regel mit der kleinsten Nummer an.

Regel 1: Es existiert ein Knoten  $v$  mit  $\deg(v) = 1$ .

Regel 2: Es existiert ein Knoten  $v$  mit  $\deg(v) \geq 5$ .

Regel 3: Es existiert ein Knoten  $v$  mit  $\deg(v) = 2$ .

Regel 4: Es existiert ein Knoten  $v$  mit  $\deg(v) = 3$ .

Regel 5: Alle Knoten haben Grad 4.

→ Offensichtlich ist immer eine der Regeln anwendbar.

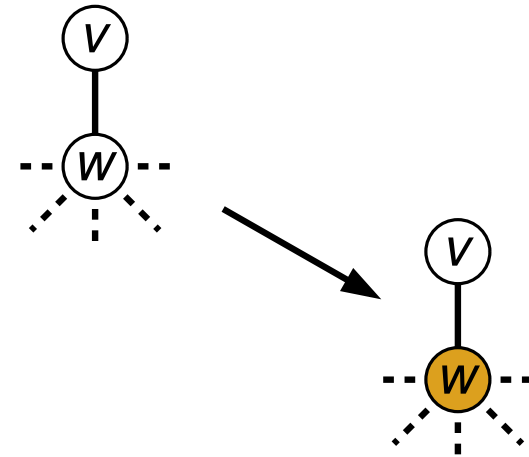
# Verzweigungsregeln

Regel 1: Es existiert ein Knoten  $v$  mit  $\deg(v) = 1$ .

Sei  $w$  der Nachbar von  $v$ .

Es kann nie besser sein,  $v$  statt  $w$  zu nehmen.

⇒ keine Verzweigung nötig.



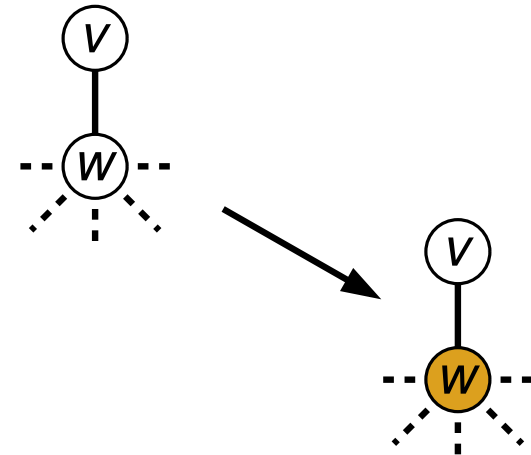
# Verzweigungsregeln

Regel 1: Es existiert ein Knoten  $v$  mit  $\deg(v) = 1$ .

Sei  $w$  der Nachbar von  $v$ .

Es kann nie besser sein,  $v$  statt  $w$  zu nehmen.

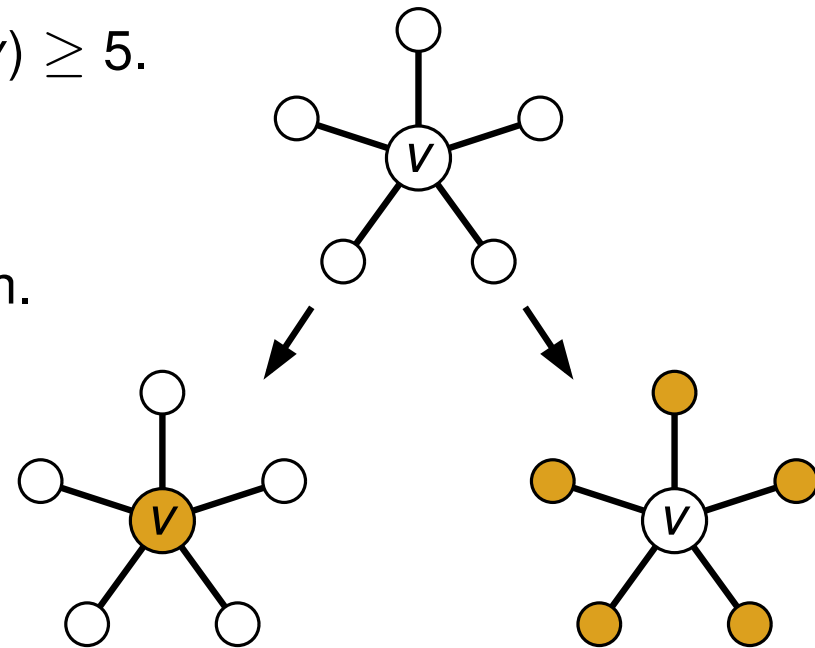
⇒ keine Verzweigung nötig.



Regel 2: Es existiert ein Knoten  $v$  mit  $\deg(v) \geq 5$ .

Wähle entweder  $v$  oder alle seine Nachbarn.

⇒ Verzweigungsvektor  $(1, 5)$



# Verzweigungsregeln

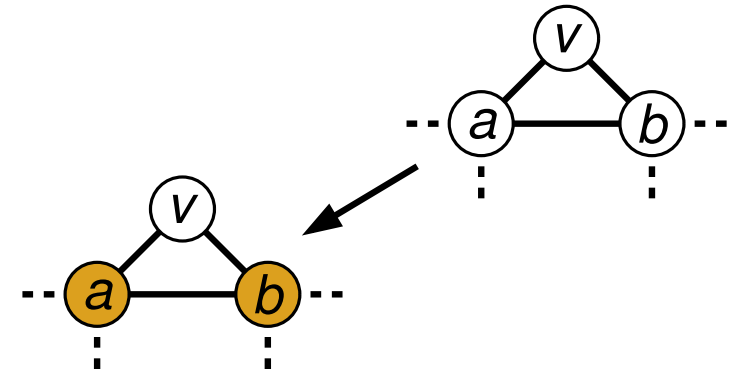
Regel 3: Es existiert ein Knoten  $v$  mit  $\deg(v) = 2$ .

Fall 3.1: Die Nachbarn  $a$  und  $b$  von  $v$  sind adjazent.

⇒ Zwei der Knoten  $v$ ,  $a$ ,  $b$  müssen gewählt werden.

⇒ Es ist nie besser  $v$  zu wählen.

⇒ Keine Verzweigung nötig.



# Verzweigungsregeln

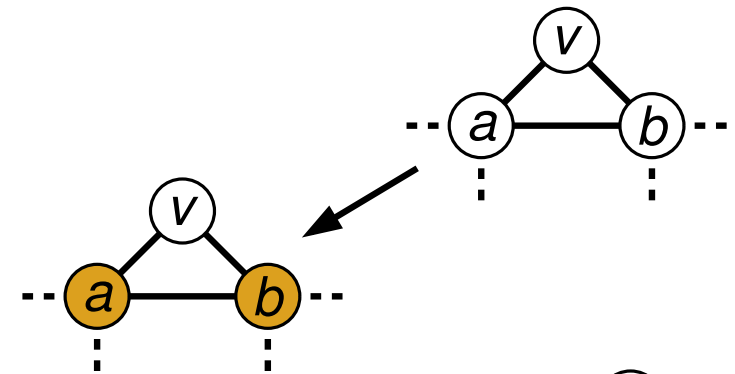
Regel 3: Es existiert ein Knoten  $v$  mit  $\deg(v) = 2$ .

Fall 3.1: Die Nachbarn  $a$  und  $b$  von  $v$  sind adjazent.

⇒ Zwei der Knoten  $v$ ,  $a$ ,  $b$  müssen gewählt werden.

⇒ Es ist nie besser  $v$  zu wählen.

⇒ Keine Verzweigung nötig.

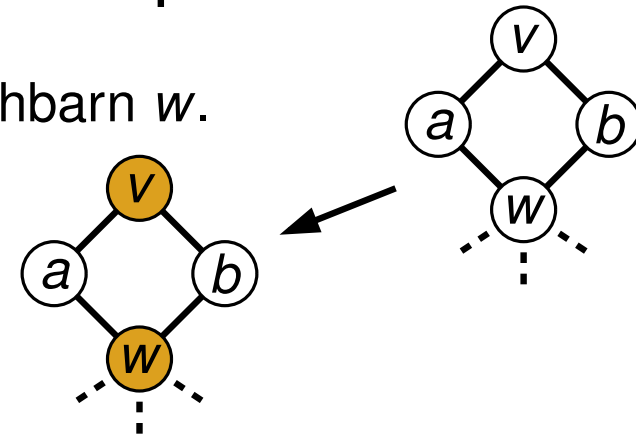


Fall 3.2:  $a$  und  $b$  haben Grad zwei und gemeinsamen Nachbarn  $w$ .

⇒ Zwei der Knoten  $v$ ,  $a$ ,  $b$ ,  $w$  müssen gewählt werden.

⇒ Es ist nie besser  $a$  und  $b$  zu wählen.

⇒ Keine Verzweigung nötig.





# Verzweigungsregeln

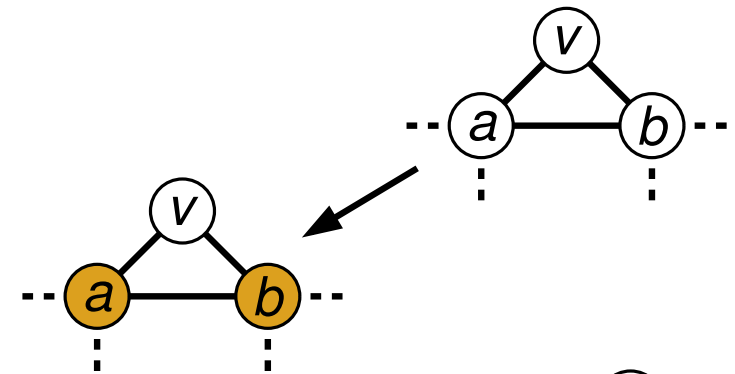
Regel 3: Es existiert ein Knoten  $v$  mit  $\deg(v) = 2$ .

Fall 3.1: Die Nachbarn  $a$  und  $b$  von  $v$  sind adjazent.

⇒ Zwei der Knoten  $v$ ,  $a$ ,  $b$  müssen gewählt werden.

⇒ Es ist nie besser  $v$  zu wählen.

⇒ Keine Verzweigung nötig.

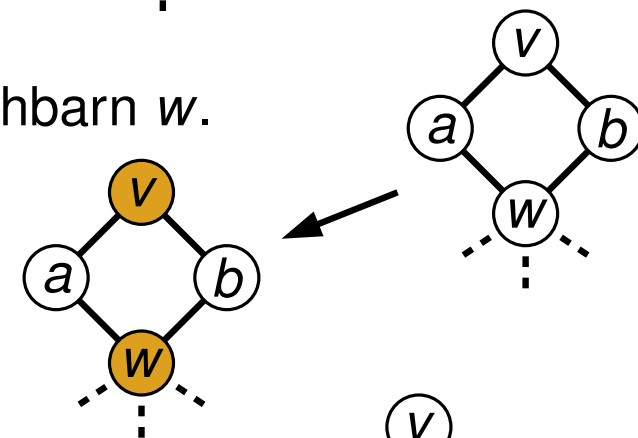


Fall 3.2:  $a$  und  $b$  haben Grad zwei und gemeinsamen Nachbarn  $w$ .

⇒ Zwei der Knoten  $v$ ,  $a$ ,  $b$ ,  $w$  müssen gewählt werden.

⇒ Es ist nie besser  $a$  und  $b$  zu wählen.

⇒ Keine Verzweigung nötig.



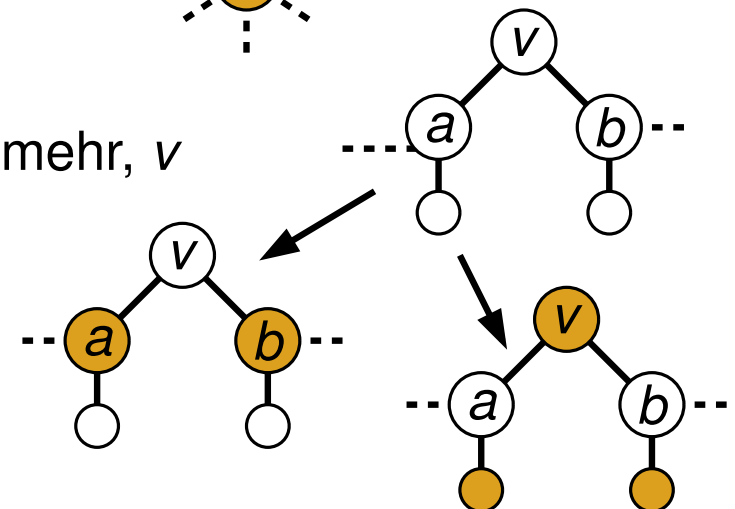
Fall 3.3: sonst

Falls  $a$  oder  $b$  gewählt wurden, macht es keinen Sinn mehr,  $v$  zu wählen.

⇒ Wähle entweder  $a$  und  $b$  oder  $N(a) \cup N(b)$ .

Es gilt  $|N(a) \cup N(b)| \geq 3$

⇒ Verzweigungsvektor  $(2, 3)$



# Verzweigungsregeln

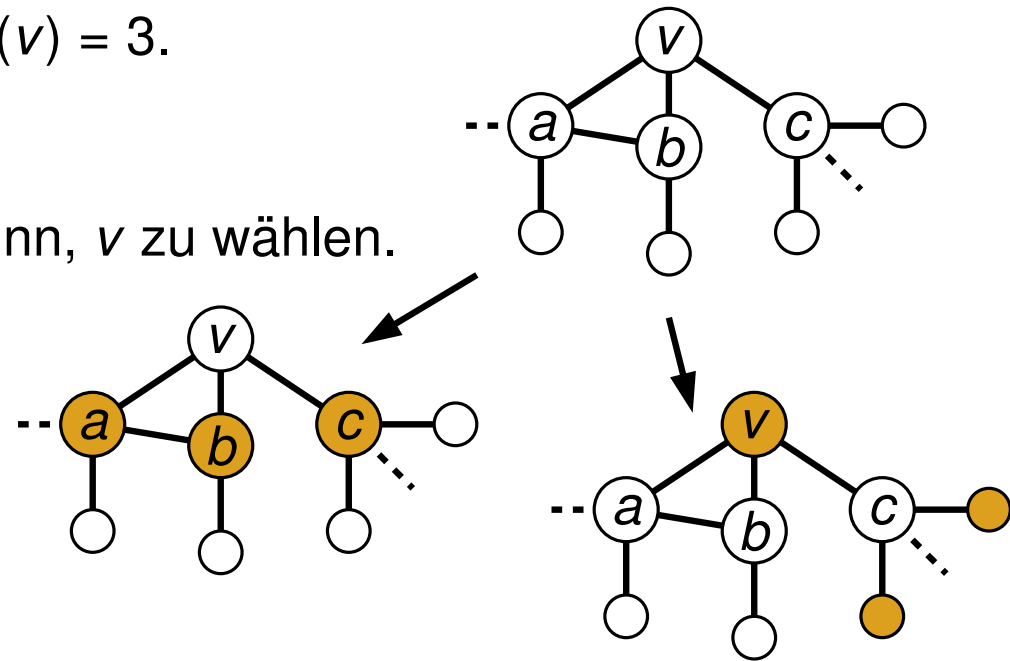
Regel 4: Es existiert ein Knoten  $v$  mit  $\deg(v) = 3$ .

Fall 4.1:  $v$  ist Teil eines Dreiecks  $v, a, b$ .

Falls  $c$  gewählt wurde, macht es keinen Sinn,  $v$  zu wählen.

⇒ Füge entweder  $N(v)$  oder  $N(c)$  hinzu.

⇒ Verzweigungsvektor  $(3, 3)$



# Verzweigungsregeln

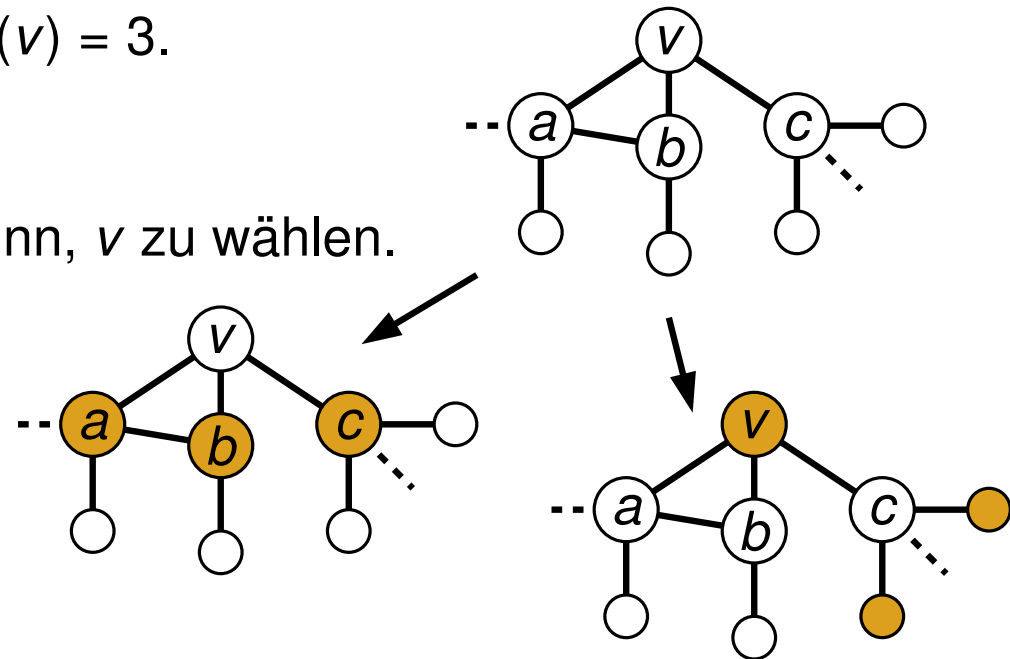
Regel 4: Es existiert ein Knoten  $v$  mit  $\deg(v) = 3$ .

Fall 4.1:  $v$  ist Teil eines Dreiecks  $v, a, b$ .

Falls  $c$  gewählt wurde, macht es keinen Sinn,  $v$  zu wählen.

⇒ Füge entweder  $N(v)$  oder  $N(c)$  hinzu.

⇒ Verzweigungsvektor  $(3, 3)$

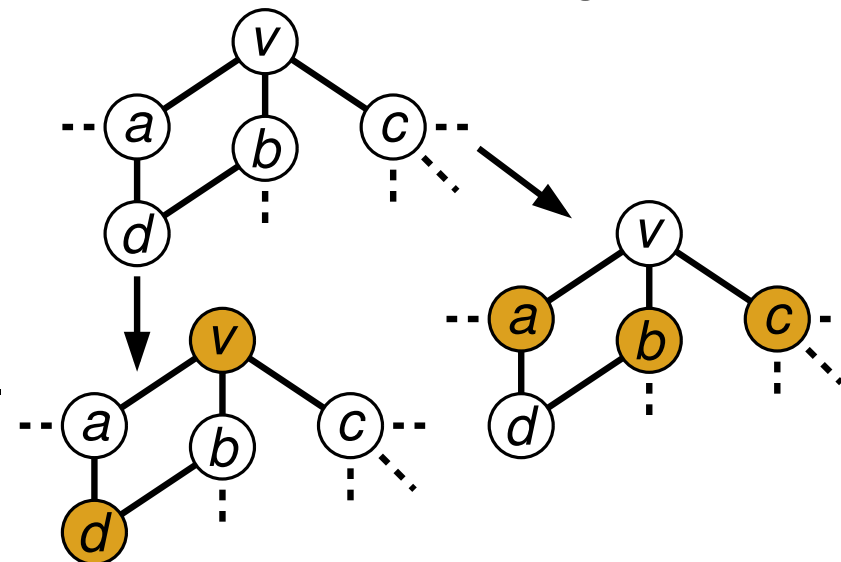


Fall 4.2:  $v$  ist Teil eines Vierecks  $v, a, d, b$ .

In jedem Vertex Cover sind zumindest  $v$  und  $d$  oder  $a$  und  $b$  enthalten. Wenn  $a$  und  $b$  enthalten sind macht es keinen Sinn  $v$  zu wählen.

⇒ Füge entweder  $a, b$  und  $c$  oder  $v$  und  $d$  hinzu.

⇒ Verzweigungsvektor  $(2, 3)$



# Verzweigungsregeln

Regel 4: Es existiert ein Knoten  $v$  mit  $\deg(v) = 3$ .

Fall 4.3:  $v$  ist in keinem Drei- oder Viereck enthalten und es gilt  $\deg(a) \geq 4$  (für  $b$  und  $c$  analog).

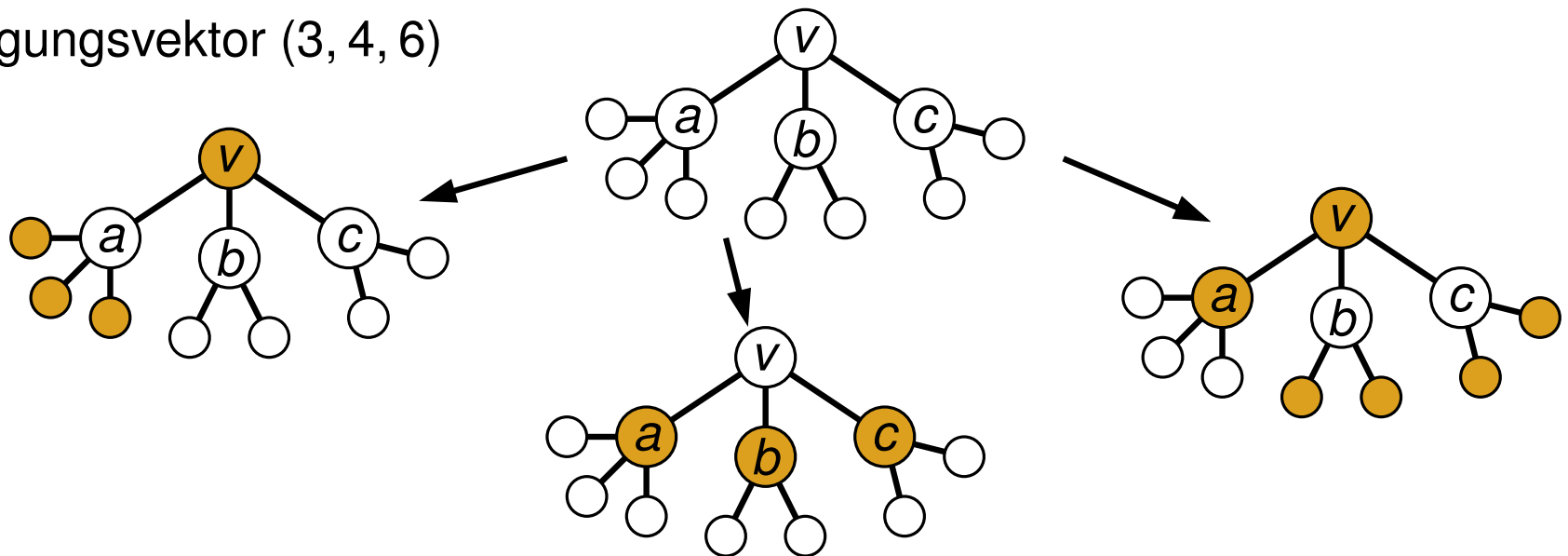
Ein Vertex Cover, das  $a$  nicht enthält, enthält  $N(a)$ .

Ist  $a$  sowie einer von  $b$  oder  $c$  enthalten, so macht es keinen Sinn  $v$  zu wählen.

⇒ Wenn  $a$  enthalten ist, sind entweder  $b$  und  $c$  enthalten oder beide nicht.

⇒ Füge entweder  $N(a)$  oder  $a, b$  und  $c$  oder  $a, N(b)$  und  $N(c)$  hinzu.

⇒ Verzweigungsvektor  $(3, 4, 6)$



# Verzweigungsregeln

Regel 4: Es existiert ein Knoten  $v$  mit  $\deg(v) = 3$ .

Fall 4.4:  $v$  ist in keinem Drei- oder Viereck enthalten und es gilt  $\deg(a) = \deg(b) = \deg(c) = 3$

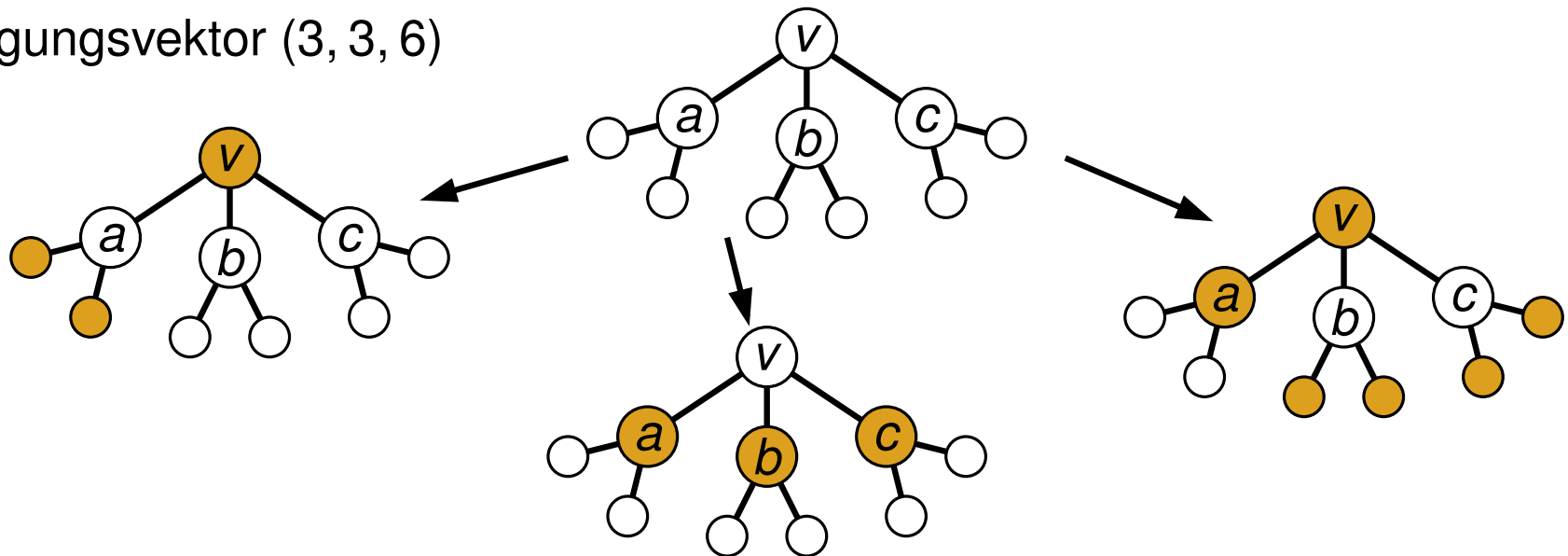
Ein Vertex Cover, das  $a$  nicht enthält, enthält  $N(a)$ .

Ist  $a$  sowie einer von  $b$  oder  $c$  enthalten, so macht es keinen Sinn  $v$  zu wählen.

⇒ Wenn  $a$  enthalten ist sind entweder  $b$  und  $c$  beide enthalten oder beide nicht.

⇒ Füge entweder  $N(a)$  oder  $a, b$  und  $c$  oder  $a, N(b)$  und  $N(c)$  hinzu.

⇒ Verzweigungsvektor  $(3, 3, 6)$



# Verzweigungsregeln

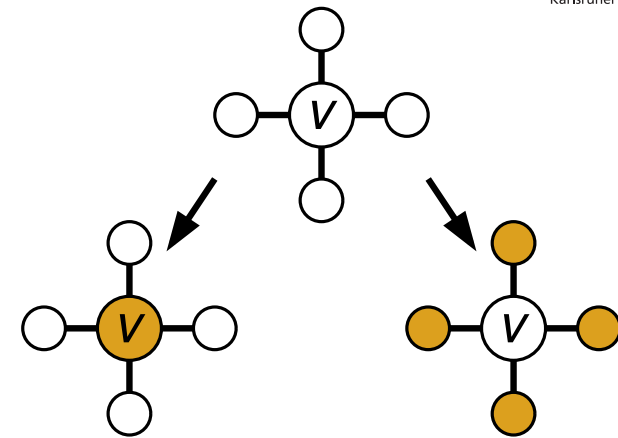
Regel 5: Alle Knoten haben Grad 4.

Wähle einen Knoten  $v$  oder alle seine Nachbarn.

⇒ Verzweigungsvektor  $(1, 4)$

## Beachte:

- Die Regel wird nur angewendet, wenn jeder Knoten Grad 4 hat.
- Jeder Subgraph enthält Knoten mit kleinerem Grad.  
⇒ Regel 5 wird pro Pfad von der Wurzel zu einem Blatt nur einmal ausgeführt.

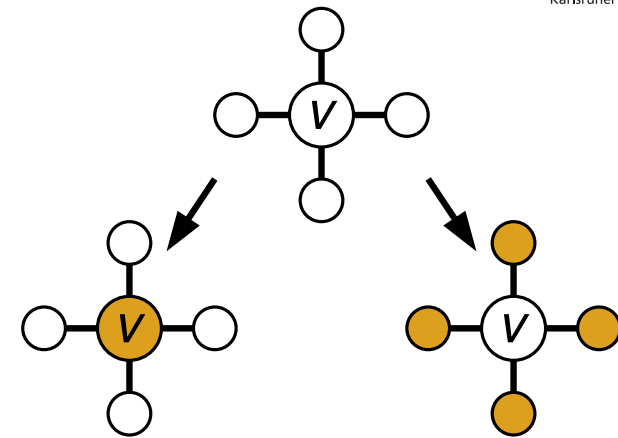


# Verzweigungsregeln

Regel 5: Alle Knoten haben Grad 4.

Wähle einen Knoten  $v$  oder alle seine Nachbarn.

⇒ Verzweigungsvektor  $(1, 4)$



## Zusammenfassung:

- Menge von Regeln zur Verzweigung (sodass immer eine der Regeln anwendbar ist).
- Erlaubte Verzweigungsvektoren:  $(1, 5)$ ,  $(2, 3)$ ,  $(3, 3)$ ,  $(3, 4, 6)$ ,  $(3, 3, 6)$ , höchstens einmal  $(1, 4)$
- Wende bei jeder Verzweigung die Regel mit der kleinsten Nummer an.

Regel 1: Es existiert ein Knoten  $v$  mit  $\deg(v) = 1$ .

Regel 2: Es existiert ein Knoten  $v$  mit  $\deg(v) \geq 5$ .

Regel 3: Es existiert ein Knoten  $v$  mit  $\deg(v) = 2$ .

Regel 4: Es existiert ein Knoten  $v$  mit  $\deg(v) = 3$ .

Regel 5: Alle Knoten haben Grad 4.

**Laufzeit für VERTEX COVER:**

$$O(kn + 1.342^k \cdot k^2)$$