

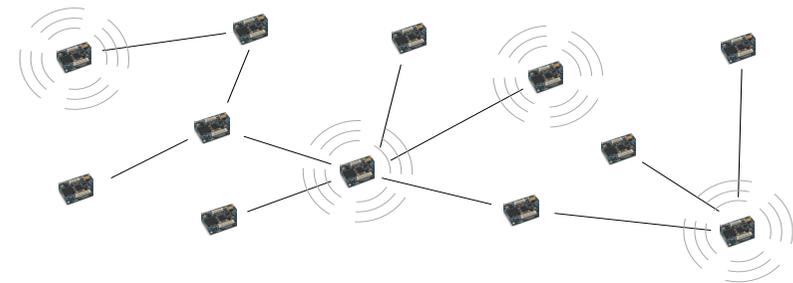
- Medium Access Control / Färbungen, Teil 2
  - kurze Wiederholung
  - Schöner verteilter Färbungsalgorithmus
- Kapazität & Scheduling
  - Interferenz etwas realistischer
  - neue Probleme und Herangehensweisen

## Algorithmen für Ad-hoc- und Sensornetze

VL 10 – Eine kurze Geschichte vom Färben (Teil 2)

Markus Völker | 04. Juli 2012 (Version 1)

INSTITUT FÜR THEORETISCHE INFORMATIK - LEHRSTUHL FÜR ALGORITHMIK (PROF. WAGNER)



## Erinnerung: MAC-Layer

- Übertragungen können nicht beliebig parallel stattfinden
  - Interferenzen naher Übertragungen verhindert erfolgreiches Dekodieren
- Lösung 1: CSMA/CA
  - Spontane Benutzung des Mediums, aber vorsichtig
  - Warten auf freien Kanal, Anfragen beim Empfänger
  - Übliche Lösung, kostet aber Energie (Kollisionen, Mithören)
- Lösung 2: TDMA/CDMA/FDMA
  - feste Zuweisung von Zeiten/Codierung/Frequenzen an Sender/Übertragungen
  - gleiche Ressource immer nur so genutzt, dass es keine Konflikte geben sollte
  - (Noch) wenig populär in WSN

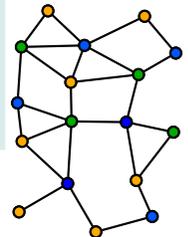
## Erinnerung: Färbungen

### Definition

Eine Knotenfärbung eines Graphen  $G = (V, E)$  ist eine Abbildung  $c : V \rightarrow \mathbb{N}$ , so dass für jede Kante die beiden Endpunkte unterschiedliche Farben haben, also

$$\{u, v\} \in E \Rightarrow c(u) \neq c(v)$$

- Eine Färbung  $c$  hat die Größe  $|c| = \max_{v \in V} c(v)$
- darauf lassen sich allgemeinere Färbungsprobleme reduzieren
- es gibt immer eine  $\Delta + 1$ -Färbung
  - in jeder schlechteren Färbung kann man die Farbe eines Knotens herabsetzen
  - das ging auch verteilt, aber bisher nur langsam



# Satz

## Satz

Es gibt einen verteilten Algorithmus, der eine  $\Delta + 1$ -Färbung in  $O(\Delta^2 + \log_2^*(n))$  Schritten berechnet.

- $\log_2^*(n)$ : Anzahl der Anwendungen von  $\log_2$ , bevor das Argument unter 1 fällt. (Bsp.:  $\log_2^*(10^{50}) = 5$ )
- Andere Möglichkeit sich das zu merken:

$$\log_2^*(x) = 5 \Leftrightarrow 2^{2^{2^2}} < x \leq 2^{2^{2^{2^2}}}$$

# Beweisstruktur

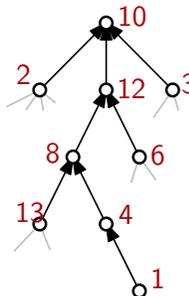
## Beweisstruktur

- es gibt einen Algorithmus, der in einem Baum in  $\log^*(n)$  Schritten eine 6-Färbung berechnet
- es gibt einen Algorithmus, der in einem Baum in  $k$  Runden eine Färbung mit  $3 + k$  Farben auf 3 Farben reduziert
- beide Verfahren funktionieren auch auf „Pseudo-Wäldern“
- jeder Graph läßt sich in konstanter Zeit verteilt in  $\Delta$  Pseudo-Wälder zerlegen
- $\Delta$  Färbungen mit je 3 Farben lassen sich in  $\Delta^2$  Runden auf  $\Delta + 1$  Farben reduzieren

# 6-Färbung eines Baumes

## 6-Färbung eines Baumes

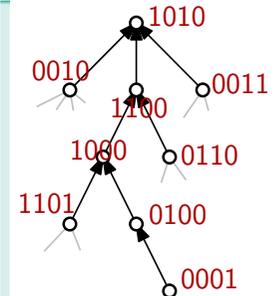
- 1 schreibe Label binär in  $\log_2 n$  Bits
  - (dafür muss a priori zumindest obere Schranke an  $n$  bekannt sein)
- 2 Setze  $L = \log_2 n$  (aktuelle Labellänge)
- 3 Solange  $L > 3$  und dann noch einmal:
  - suche ein Bit, in dem sich das eigene Label und das des Vorgängers unterscheiden
  - kodiere dessen Position in  $\log_2(L)$  Bits
  - verwende als neues Label die kodierte Position und den Wert des eigenen Bits
  - die Wurzel wählt dafür beliebiges Bit aus
  - setze  $L = \log_2(L) + 1$  (neue Labellänge)



# 6-Färbung eines Baumes

## 6-Färbung eines Baumes

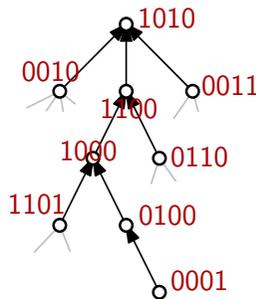
- 1 schreibe Label binär in  $\log_2 n$  Bits
  - (dafür muss a priori zumindest obere Schranke an  $n$  bekannt sein)
- 2 Setze  $L = \log_2 n$  (aktuelle Labellänge)
- 3 Solange  $L > 3$  und dann noch einmal:
  - suche ein Bit, in dem sich das eigene Label und das des Vorgängers unterscheiden
  - kodiere dessen Position in  $\log_2(L)$  Bits
  - verwende als neues Label die kodierte Position und den Wert des eigenen Bits
  - die Wurzel wählt dafür beliebiges Bit aus
  - setze  $L = \log_2(L) + 1$  (neue Labellänge)



## 6-Färbung eines Baumes

### 6-Färbung eines Baumes

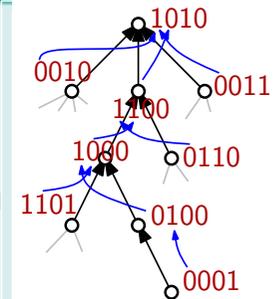
- 1 schreibe Label binär in  $\log_2 n$  Bits
  - (dafür muss a priori zumindest obere Schranke an  $n$  bekannt sein)
- 2 Setze  $L = \log_2 n$  (aktuelle Labellänge)
- 3 Solange  $L > 3$  und dann noch einmal:
  - suche ein Bit, in dem sich das eigene Label und das des Vorgängers unterscheiden
  - kodiere dessen Position in  $\log_2(L)$  Bits
  - verwende als neues Label die kodierte Position und den Wert des eigenen Bits
  - die Wurzel wählt dafür beliebiges Bit aus
  - setze  $L = \log_2(L) + 1$  (neue Labellänge)



## 6-Färbung eines Baumes

### 6-Färbung eines Baumes

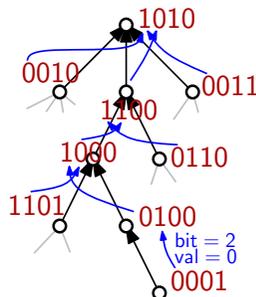
- 1 schreibe Label binär in  $\log_2 n$  Bits
  - (dafür muss a priori zumindest obere Schranke an  $n$  bekannt sein)
- 2 Setze  $L = \log_2 n$  (aktuelle Labellänge)
- 3 Solange  $L > 3$  und dann noch einmal:
  - suche ein Bit, in dem sich das eigene Label und das des Vorgängers unterscheiden
  - kodiere dessen Position in  $\log_2(L)$  Bits
  - verwende als neues Label die kodierte Position und den Wert des eigenen Bits
  - die Wurzel wählt dafür beliebiges Bit aus
  - setze  $L = \log_2(L) + 1$  (neue Labellänge)



## 6-Färbung eines Baumes

### 6-Färbung eines Baumes

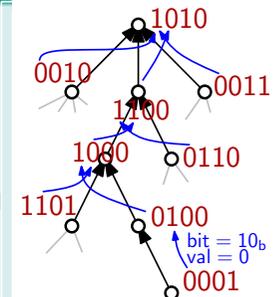
- 1 schreibe Label binär in  $\log_2 n$  Bits
  - (dafür muss a priori zumindest obere Schranke an  $n$  bekannt sein)
- 2 Setze  $L = \log_2 n$  (aktuelle Labellänge)
- 3 Solange  $L > 3$  und dann noch einmal:
  - suche ein Bit, in dem sich das eigene Label und das des Vorgängers unterscheiden
  - kodiere dessen Position in  $\log_2(L)$  Bits
  - verwende als neues Label die kodierte Position und den Wert des eigenen Bits
  - die Wurzel wählt dafür beliebiges Bit aus
  - setze  $L = \log_2(L) + 1$  (neue Labellänge)



## 6-Färbung eines Baumes

### 6-Färbung eines Baumes

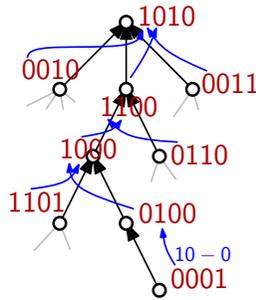
- 1 schreibe Label binär in  $\log_2 n$  Bits
  - (dafür muss a priori zumindest obere Schranke an  $n$  bekannt sein)
- 2 Setze  $L = \log_2 n$  (aktuelle Labellänge)
- 3 Solange  $L > 3$  und dann noch einmal:
  - suche ein Bit, in dem sich das eigene Label und das des Vorgängers unterscheiden
  - kodiere dessen Position in  $\log_2(L)$  Bits
  - verwende als neues Label die kodierte Position und den Wert des eigenen Bits
  - die Wurzel wählt dafür beliebiges Bit aus
  - setze  $L = \log_2(L) + 1$  (neue Labellänge)



## 6-Färbung eines Baumes

### 6-Färbung eines Baumes

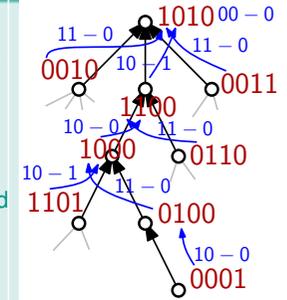
- 1 schreibe Label binär in  $\log_2 n$  Bits
  - (dafür muss a priori zumindest obere Schranke an  $n$  bekannt sein)
- 2 Setze  $L = \log_2 n$  (aktuelle Labellänge)
- 3 Solange  $L > 3$  und dann noch einmal:
  - suche ein Bit, in dem sich das eigene Label und das des Vorgängers unterscheiden
  - kodiere dessen Position in  $\log_2(L)$  Bits
  - verwende als neues Label die kodierte Position und den Wert des eigenen Bits
  - die Wurzel wählt dafür beliebiges Bit aus
  - setze  $L = \log_2(L) + 1$  (neue Labellänge)



## 6-Färbung eines Baumes

### 6-Färbung eines Baumes

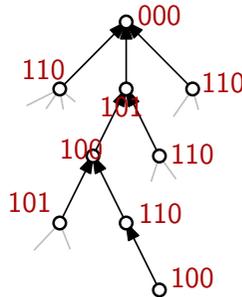
- 1 schreibe Label binär in  $\log_2 n$  Bits
  - (dafür muss a priori zumindest obere Schranke an  $n$  bekannt sein)
- 2 Setze  $L = \log_2 n$  (aktuelle Labellänge)
- 3 Solange  $L > 3$  und dann noch einmal:
  - suche ein Bit, in dem sich das eigene Label und das des Vorgängers unterscheiden
  - kodiere dessen Position in  $\log_2(L)$  Bits
  - verwende als neues Label die kodierte Position und den Wert des eigenen Bits
  - die Wurzel wählt dafür beliebiges Bit aus
  - setze  $L = \log_2(L) + 1$  (neue Labellänge)



## 6-Färbung eines Baumes

### 6-Färbung eines Baumes

- 1 schreibe Label binär in  $\log_2 n$  Bits
  - (dafür muss a priori zumindest obere Schranke an  $n$  bekannt sein)
- 2 Setze  $L = \log_2 n$  (aktuelle Labellänge)
- 3 Solange  $L > 3$  und dann noch einmal:
  - suche ein Bit, in dem sich das eigene Label und das des Vorgängers unterscheiden
  - kodiere dessen Position in  $\log_2(L)$  Bits
  - verwende als neues Label die kodierte Position und den Wert des eigenen Bits
  - die Wurzel wählt dafür beliebiges Bit aus
  - setze  $L = \log_2(L) + 1$  (neue Labellänge)



## Korrektheit Baum-6-Färbung

### Satz

Der beschriebene Algorithmus berechnet eine 6-Färbung.

- Nach jeder Runde liegt eine Färbung des Baumes vor
  - Annahme: zwei Knoten haben nach einer Runde dasselbe Label
  - dann haben sie beide dasselbe Bit ausgewählt und hatten dort vorher denselben Wert stehen
  - aber das Kind sollte Bit wählen, in dem es sich vom Vorgänger unterscheidet! Widerspruch!
- Die Färbung enthält zum Schluss maximal 6 Farben
  - $L$  ist immer aktuelle Labellänge
  - sobald  $L \leq 3$  wird noch eine Runde ausgeführt.
  - dann reichen die drei Beschreibungen 00, 01, 10 für die Position
  - dann gibt es nur noch die Labels 000, 001, 010, 011, 100, 101

## Laufzeit Baum-6-Färbung

### Satz

Der beschriebene Algorithmus terminiert nach  $O(\log^*(n))$  Schritten.

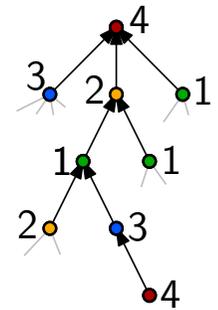
- Das werden wir nicht beweisen.
- Grobe Vorstellung:
  - in jeder Runde wird die Labellänge von  $L$  auf  $\log_2(L) + 1$  gedrückt
  - ohne das  $+1$  wäre die Aussage klar
  - ein bisschen Formelschieberei löst das Problem (bei Interesse an den Details: siehe Buch)

Das war schon Teil 1: Wir können einen Baum in  $O(\log^*(n))$  Schritten mit 6 Farben färben!

## Kompression von 6 auf 3 Farben im Baum

### Baum-(3+k)-zu-3-Färbung

Für  $i = (3 + k), \dots, 4$  tue folgendes:

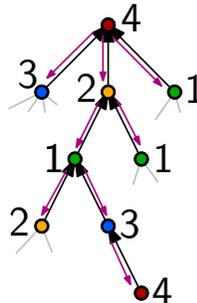


## Kompression von 6 auf 3 Farben im Baum

### Baum-(3+k)-zu-3-Färbung

Für  $i = (3 + k), \dots, 4$  tue folgendes:

- 1 jeder Knoten nimmt die Farbe seines Vorgängers an („shift down“)
  - Wurzel nimmt *neue Farbe* aus  $\{1, 2, 3\}$

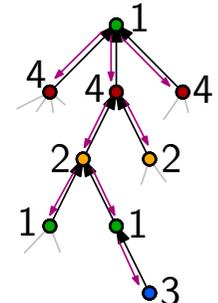


## Kompression von 6 auf 3 Farben im Baum

### Baum-(3+k)-zu-3-Färbung

Für  $i = (3 + k), \dots, 4$  tue folgendes:

- 1 jeder Knoten nimmt die Farbe seines Vorgängers an („shift down“)
  - Wurzel nimmt *neue Farbe* aus  $\{1, 2, 3\}$



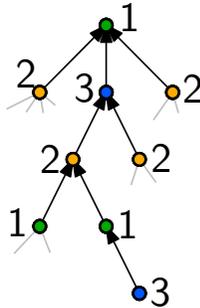
# Kompression von 6 auf 3 Farben im Baum



## Baum-(3+k)-zu-3-Färbung

Für  $i = (3 + k), \dots, 4$  tue folgendes:

- 1 jeder Knoten nimmt die Farbe seines Vorgängers an („shift down“)
  - Wurzel nimmt *neue Farbe* aus  $\{1, 2, 3\}$
- 2 jeder Knoten mit Farbe  $i$  wählt Farbe aus  $\{1, 2, 3\}$ , die kein Nachbar hat



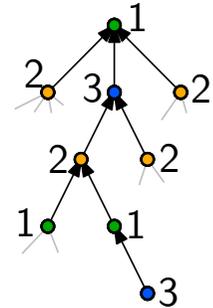
# Kompression von 6 auf 3 Farben im Baum



## Baum-(3+k)-zu-3-Färbung

Für  $i = (3 + k), \dots, 4$  tue folgendes:

- 1 jeder Knoten nimmt die Farbe seines Vorgängers an („shift down“)
    - Wurzel nimmt *neue Farbe* aus  $\{1, 2, 3\}$
  - 2 jeder Knoten mit Farbe  $i$  wählt Farbe aus  $\{1, 2, 3\}$ , die kein Nachbar hat
- Korrektheit:
    - nach einem Shift hat jeder Knoten die Farbe seines Vorgängers, und seine Kinder seine alte Farbe
      - das ist eine Färbung!
      - jeder Knoten „sieht“ nur 2 benachbarte Farben
    - in jeder Runde werden wir Farbe  $i$  los!
  - nach  $k$  Runden je 2 Schritte haben wir noch 3 Farben!



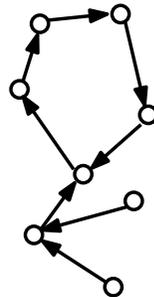
# Pseudo-Wälder (klingt schlimmer als es ist)



## Definition Pseudo-Wald

Ein gerichteter Graph heißt *Pseudo-Wald*, wenn jeder Knoten höchstens eine ausgehende Kante hat.

- jeder Knoten zeigt auf maximal einen „Vorgänger“
  - Vorsicht: Graph kann gerichtete Kreise enthalten



## Satz

Die Dreifärbung von Pseudo-Wäldern funktioniert exakt wie bei Bäumen.

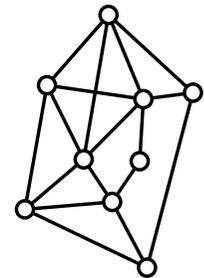
- Beweise gehen *exakt* wie vorher, nur gibt es ggf. mehrere Knoten ohne Vorgänger

# Pseudo-Wald-Dekomposition



## Definition

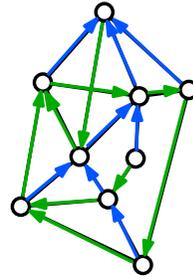
Eine *Pseudo-Wald-Dekomposition* eines Graphen  $G$  ist eine Menge von Pseudo-Wäldern  $F_1, \dots, F_\Delta$ , so dass jede Kante von  $G$  in mindestens einem  $F_i$  vorkommt (in beliebiger Richtung).



## Pseudo-Wald-Dekomposition

### Definition

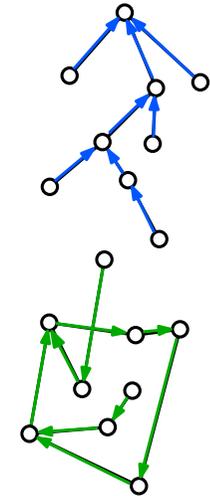
Eine *Pseudo-Wald-Dekomposition* eines Graphen  $G$  ist eine Menge von Pseudo-Wäldern  $F_1, \dots, F_\Delta$ , so dass jede Kante von  $G$  in mindestens einem  $F_i$  vorkommt (in beliebiger Richtung).



## Pseudo-Wald-Dekomposition

### Definition

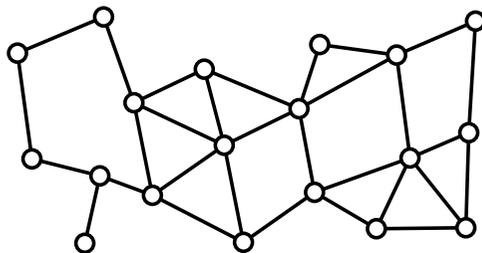
Eine *Pseudo-Wald-Dekomposition* eines Graphen  $G$  ist eine Menge von Pseudo-Wäldern  $F_1, \dots, F_\Delta$ , so dass jede Kante von  $G$  in mindestens einem  $F_i$  vorkommt (in beliebiger Richtung).



## Verteilte Pseudo-Wald-Dekomposition

### Verteilte Dekomposition, Runde $i = 1, \dots, \Delta$

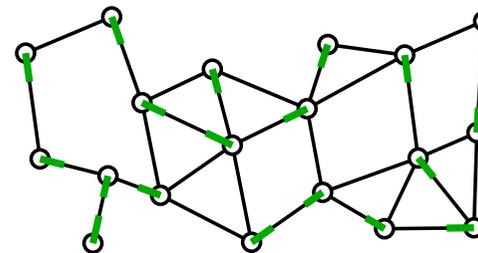
- 1 jeder Knoten wählt eine inzidente unmarkierte Kante aus, wenn noch möglich
- 2 eine gewählte Kante  $\{u, v\}$  wird markiert und zu  $F_i$  hinzugefügt, und zwar von einem Knoten weg, der sie wählte.



## Verteilte Pseudo-Wald-Dekomposition

### Verteilte Dekomposition, Runde $i = 1, \dots, \Delta$

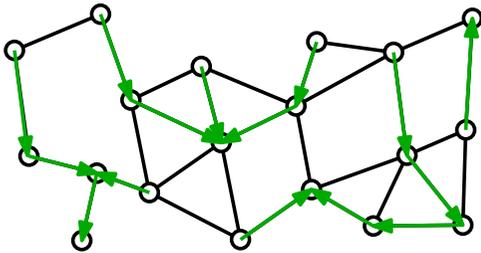
- 1 jeder Knoten wählt eine inzidente unmarkierte Kante aus, wenn noch möglich
- 2 eine gewählte Kante  $\{u, v\}$  wird markiert und zu  $F_i$  hinzugefügt, und zwar von einem Knoten weg, der sie wählte.



## Verteilte Pseudo-Wald-Dekomposition

### Verteilte Dekomposition, Runde $i = 1, \dots, \Delta$

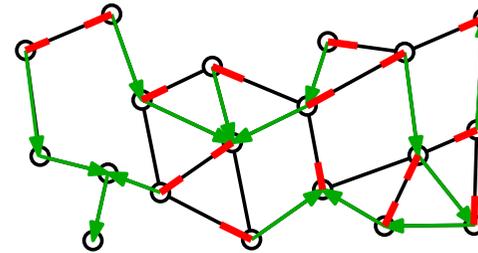
- 1 jeder Knoten wählt eine inzidente unmarkierte Kante aus, wenn noch möglich
- 2 eine gewählte Kante  $\{u, v\}$  wird markiert und zu  $F_i$  hinzugefügt, und zwar von einem Knoten weg, der sie wählte.



## Verteilte Pseudo-Wald-Dekomposition

### Verteilte Dekomposition, Runde $i = 1, \dots, \Delta$

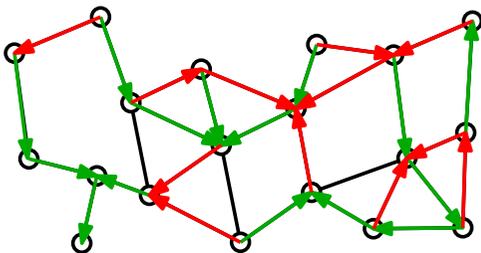
- 1 jeder Knoten wählt eine inzidente unmarkierte Kante aus, wenn noch möglich
- 2 eine gewählte Kante  $\{u, v\}$  wird markiert und zu  $F_i$  hinzugefügt, und zwar von einem Knoten weg, der sie wählte.



## Verteilte Pseudo-Wald-Dekomposition

### Verteilte Dekomposition, Runde $i = 1, \dots, \Delta$

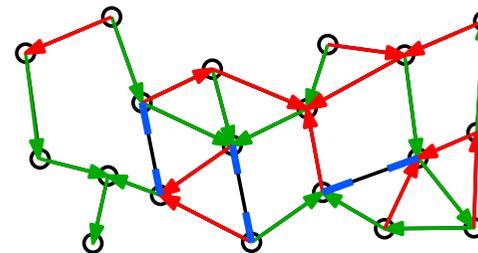
- 1 jeder Knoten wählt eine inzidente unmarkierte Kante aus, wenn noch möglich
- 2 eine gewählte Kante  $\{u, v\}$  wird markiert und zu  $F_i$  hinzugefügt, und zwar von einem Knoten weg, der sie wählte.



## Verteilte Pseudo-Wald-Dekomposition

### Verteilte Dekomposition, Runde $i = 1, \dots, \Delta$

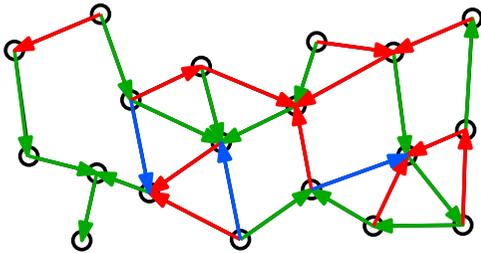
- 1 jeder Knoten wählt eine inzidente unmarkierte Kante aus, wenn noch möglich
- 2 eine gewählte Kante  $\{u, v\}$  wird markiert und zu  $F_i$  hinzugefügt, und zwar von einem Knoten weg, der sie wählte.



## Verteilte Pseudo-Wald-Dekomposition

### Verteilte Dekomposition, Runde $i = 1, \dots, \Delta$

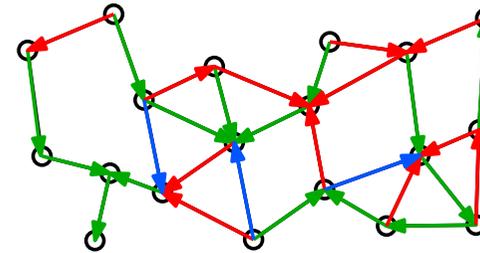
- 1 jeder Knoten wählt eine inzidente unmarkierte Kante aus, wenn noch möglich
- 2 eine gewählte Kante  $\{u, v\}$  wird markiert und zu  $F_i$  hinzugefügt, und zwar von einem Knoten weg, der sie wählte.



## Verteilte Pseudo-Wald-Dekomposition

### Verteilte Dekomposition, Runde $i = 1, \dots, \Delta$

- 1 jeder Knoten wählt eine inzidente unmarkierte Kante aus, wenn noch möglich
- 2 eine gewählte Kante  $\{u, v\}$  wird markiert und zu  $F_i$  hinzugefügt, und zwar von einem Knoten weg, der sie wählte.

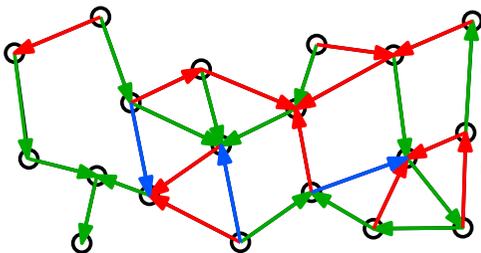


- nach  $\Delta$  Runden (oder früher) sind alle Kanten markiert

## Verteilte Pseudo-Wald-Dekomposition

### Verteilte Dekomposition, Runde $i = 1, \dots, \Delta$

- 1 jeder Knoten wählt eine inzidente unmarkierte Kante aus, wenn noch möglich
- 2 eine gewählte Kante  $\{u, v\}$  wird markiert und zu  $F_i$  hinzugefügt, und zwar von einem Knoten weg, der sie wählte.



- nach  $\Delta$  Runden (oder früher) sind alle Kanten markiert
- in jeder Runde wird ein Pseudo-Wald gewählt

## Wo stehen wir?

- Wir können den Graphen in einem Schritt in  $\Delta$  Pseudo-Wälder dekomponieren
  - Wir können Bäume und Pseudo-Wälder verteilt mit 3 Farben einfärben
    - erst färben wir in  $\log^*(n)$  Runden mit konstant vielen Schritten 6-farbig,
    - dann reduzieren wir die Farben in 3 Runden mit konstant vielen Schritten auf 3
- ⇒ Wir können in  $O(\log^*(n))$  Schritten jeden der Pseudo-Wälder dreifärben
- das kann in allen Pseudowäldern parallel geschehen

was fangen wir mit  $\Delta$  3-Färbungen für die Pseudo-Wälder an? Wir wollen eine  $\Delta + 1$ -Färbung des Graphen!

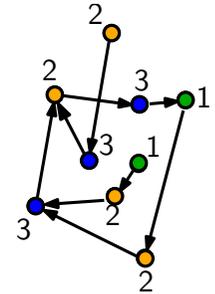
## Färbungen zusammenführen

Idee: Wir blenden nacheinander die Kanten der Pseudo-Wälder ein und *warten* eine Färbung mit  $\Delta + 1$  Farben!

## Färbungen zusammenführen

Idee: Wir blenden nacheinander die Kanten der Pseudo-Wälder ein und *warten* eine Färbung mit  $\Delta + 1$  Farben!

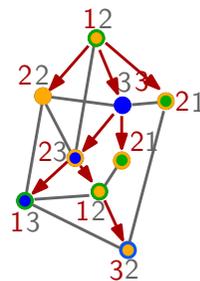
- Wir beginnen mit  $F_1$  und dessen 3 Farben
  - das ist eine  $\Delta + 1$ -Färbung von  $F_i$



## Färbungen zusammenführen

Idee: Wir blenden nacheinander die Kanten der Pseudo-Wälder ein und *warten* eine Färbung mit  $\Delta + 1$  Farben!

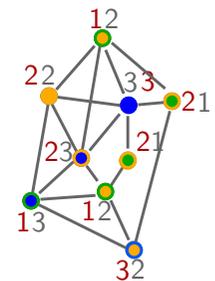
- Wir beginnen mit  $F_1$  und dessen 3 Farben
  - das ist eine  $\Delta + 1$ -Färbung von  $F_i$
- die 3-Färbung von  $F_j$  und die  $\Delta + 1$ -Färbung von  $F_1 \cup \dots \cup F_{j-1}$  ergeben  $3(\Delta + 1)$  „Mischfarben“  $(1, 0), \dots, (3, \Delta + 1)$



## Färbungen zusammenführen

Idee: Wir blenden nacheinander die Kanten der Pseudo-Wälder ein und *warten* eine Färbung mit  $\Delta + 1$  Farben!

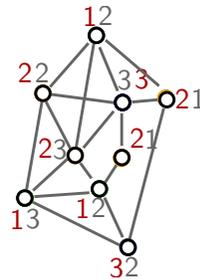
- Wir beginnen mit  $F_1$  und dessen 3 Farben
  - das ist eine  $\Delta + 1$ -Färbung von  $F_i$
- die 3-Färbung von  $F_j$  und die  $\Delta + 1$ -Färbung von  $F_1 \cup \dots \cup F_{j-1}$  ergeben  $3(\Delta + 1)$  „Mischfarben“  $(1, 0), \dots, (3, \Delta + 1)$
- in  $3(\Delta + 1)$  Runden können immer Knoten mit einer bestimmten Farbe eine einfache Farbe aus  $\{1, \Delta + 1\}$  wählen, die keiner der Nachbarn hat



## Färbungen zusammenführen

Idee: Wir blenden nacheinander die Kanten der Pseudo-Wälder ein und *warten* eine Färbung mit  $\Delta + 1$  Farben!

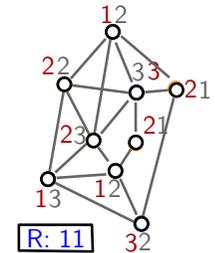
- Wir beginnen mit  $F_1$  und dessen 3 Farben
  - das ist eine  $\Delta + 1$ -Färbung von  $F_i$
- die 3-Färbung von  $F_j$  und die  $\Delta + 1$ -Färbung von  $F_1 \cup \dots \cup F_{j-1}$  ergeben  $3(\Delta + 1)$  „Mischfarben“  $(1, 0), \dots, (3, \Delta + 1)$
- in  $3(\Delta + 1)$  Runden können immer Knoten mit einer bestimmten Farbe eine einfache Farbe aus  $\{1, \Delta + 1\}$  wählen, die keiner der Nachbarn hat



## Färbungen zusammenführen

Idee: Wir blenden nacheinander die Kanten der Pseudo-Wälder ein und *warten* eine Färbung mit  $\Delta + 1$  Farben!

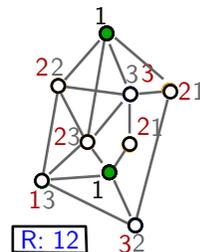
- Wir beginnen mit  $F_1$  und dessen 3 Farben
  - das ist eine  $\Delta + 1$ -Färbung von  $F_i$
- die 3-Färbung von  $F_j$  und die  $\Delta + 1$ -Färbung von  $F_1 \cup \dots \cup F_{j-1}$  ergeben  $3(\Delta + 1)$  „Mischfarben“  $(1, 0), \dots, (3, \Delta + 1)$
- in  $3(\Delta + 1)$  Runden können immer Knoten mit einer bestimmten Farbe eine einfache Farbe aus  $\{1, \Delta + 1\}$  wählen, die keiner der Nachbarn hat



## Färbungen zusammenführen

Idee: Wir blenden nacheinander die Kanten der Pseudo-Wälder ein und *warten* eine Färbung mit  $\Delta + 1$  Farben!

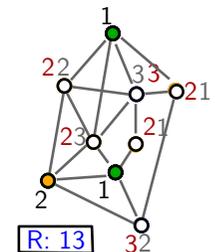
- Wir beginnen mit  $F_1$  und dessen 3 Farben
  - das ist eine  $\Delta + 1$ -Färbung von  $F_i$
- die 3-Färbung von  $F_j$  und die  $\Delta + 1$ -Färbung von  $F_1 \cup \dots \cup F_{j-1}$  ergeben  $3(\Delta + 1)$  „Mischfarben“  $(1, 0), \dots, (3, \Delta + 1)$
- in  $3(\Delta + 1)$  Runden können immer Knoten mit einer bestimmten Farbe eine einfache Farbe aus  $\{1, \Delta + 1\}$  wählen, die keiner der Nachbarn hat



## Färbungen zusammenführen

Idee: Wir blenden nacheinander die Kanten der Pseudo-Wälder ein und *warten* eine Färbung mit  $\Delta + 1$  Farben!

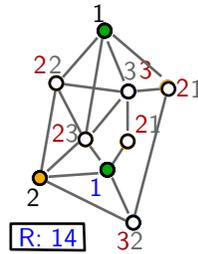
- Wir beginnen mit  $F_1$  und dessen 3 Farben
  - das ist eine  $\Delta + 1$ -Färbung von  $F_i$
- die 3-Färbung von  $F_j$  und die  $\Delta + 1$ -Färbung von  $F_1 \cup \dots \cup F_{j-1}$  ergeben  $3(\Delta + 1)$  „Mischfarben“  $(1, 0), \dots, (3, \Delta + 1)$
- in  $3(\Delta + 1)$  Runden können immer Knoten mit einer bestimmten Farbe eine einfache Farbe aus  $\{1, \Delta + 1\}$  wählen, die keiner der Nachbarn hat



## Färbungen zusammenführen

Idee: Wir blenden nacheinander die Kanten der Pseudo-Wälder ein und *warten* eine Färbung mit  $\Delta + 1$  Farben!

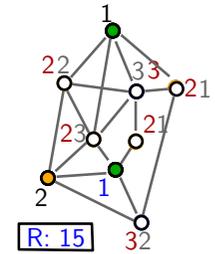
- Wir beginnen mit  $F_1$  und dessen 3 Farben
  - das ist eine  $\Delta + 1$ -Färbung von  $F_i$
- die 3-Färbung von  $F_j$  und die  $\Delta + 1$ -Färbung von  $F_1 \cup \dots \cup F_{j-1}$  ergeben  $3(\Delta + 1)$  „Mischfarben“  $(1, 0), \dots, (3, \Delta + 1)$
- in  $3(\Delta + 1)$  Runden können immer Knoten mit einer bestimmten Farbe eine einfache Farbe aus  $\{1, \Delta + 1\}$  wählen, die keiner der Nachbarn hat



## Färbungen zusammenführen

Idee: Wir blenden nacheinander die Kanten der Pseudo-Wälder ein und *warten* eine Färbung mit  $\Delta + 1$  Farben!

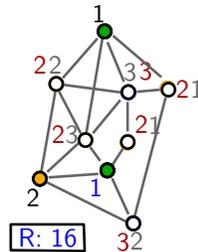
- Wir beginnen mit  $F_1$  und dessen 3 Farben
  - das ist eine  $\Delta + 1$ -Färbung von  $F_i$
- die 3-Färbung von  $F_j$  und die  $\Delta + 1$ -Färbung von  $F_1 \cup \dots \cup F_{j-1}$  ergeben  $3(\Delta + 1)$  „Mischfarben“  $(1, 0), \dots, (3, \Delta + 1)$
- in  $3(\Delta + 1)$  Runden können immer Knoten mit einer bestimmten Farbe eine einfache Farbe aus  $\{1, \Delta + 1\}$  wählen, die keiner der Nachbarn hat



## Färbungen zusammenführen

Idee: Wir blenden nacheinander die Kanten der Pseudo-Wälder ein und *warten* eine Färbung mit  $\Delta + 1$  Farben!

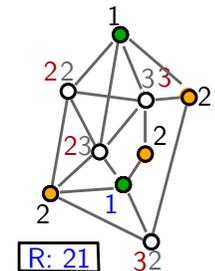
- Wir beginnen mit  $F_1$  und dessen 3 Farben
  - das ist eine  $\Delta + 1$ -Färbung von  $F_i$
- die 3-Färbung von  $F_j$  und die  $\Delta + 1$ -Färbung von  $F_1 \cup \dots \cup F_{j-1}$  ergeben  $3(\Delta + 1)$  „Mischfarben“  $(1, 0), \dots, (3, \Delta + 1)$
- in  $3(\Delta + 1)$  Runden können immer Knoten mit einer bestimmten Farbe eine einfache Farbe aus  $\{1, \Delta + 1\}$  wählen, die keiner der Nachbarn hat



## Färbungen zusammenführen

Idee: Wir blenden nacheinander die Kanten der Pseudo-Wälder ein und *warten* eine Färbung mit  $\Delta + 1$  Farben!

- Wir beginnen mit  $F_1$  und dessen 3 Farben
  - das ist eine  $\Delta + 1$ -Färbung von  $F_i$
- die 3-Färbung von  $F_j$  und die  $\Delta + 1$ -Färbung von  $F_1 \cup \dots \cup F_{j-1}$  ergeben  $3(\Delta + 1)$  „Mischfarben“  $(1, 0), \dots, (3, \Delta + 1)$
- in  $3(\Delta + 1)$  Runden können immer Knoten mit einer bestimmten Farbe eine einfache Farbe aus  $\{1, \Delta + 1\}$  wählen, die keiner der Nachbarn hat





- Färbungen sind ein leichtes Mittel, um gegenseitige Ausschlüsse zu modellieren
  - Knoten dürfen nicht gleichzeitig senden oder
  - Kanten dürfen nicht gleichzeitig aktiv sein
  - das lässt sich immer auf Knotenfärbungen reduzieren
  
- wir haben einen schnellen und verteilten Algorithmus für  $\Delta + 1$ -Färbungen kennengelernt!