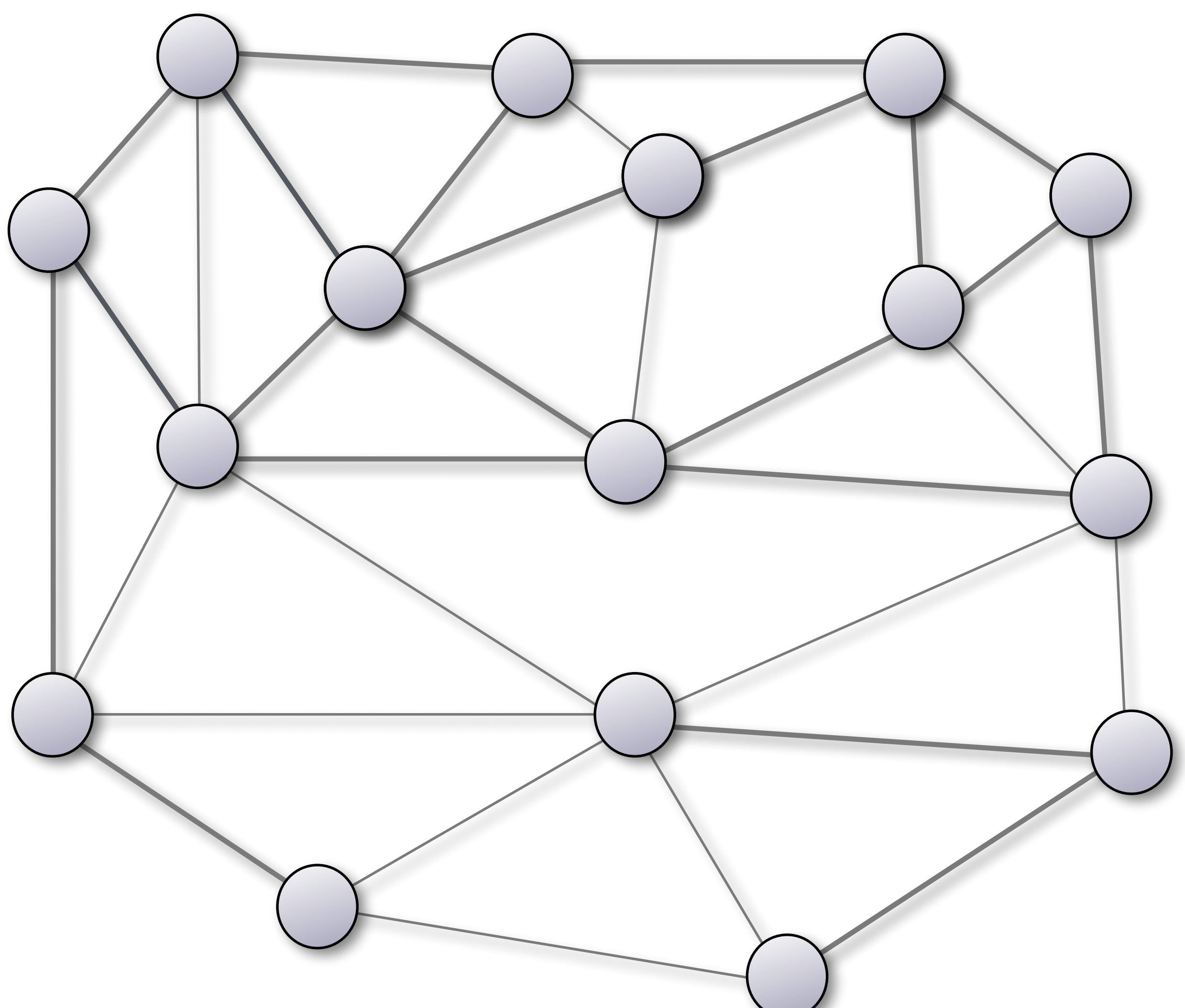
Kapitel 1.5

Färbungen

k-Färbung, chromatische Zahl

Eine (Knoten-)Färbung eines Graphen $G = (V, E)$ mit k Farben ist eine Abbildung $c : V \to [k]$, so dass gilt

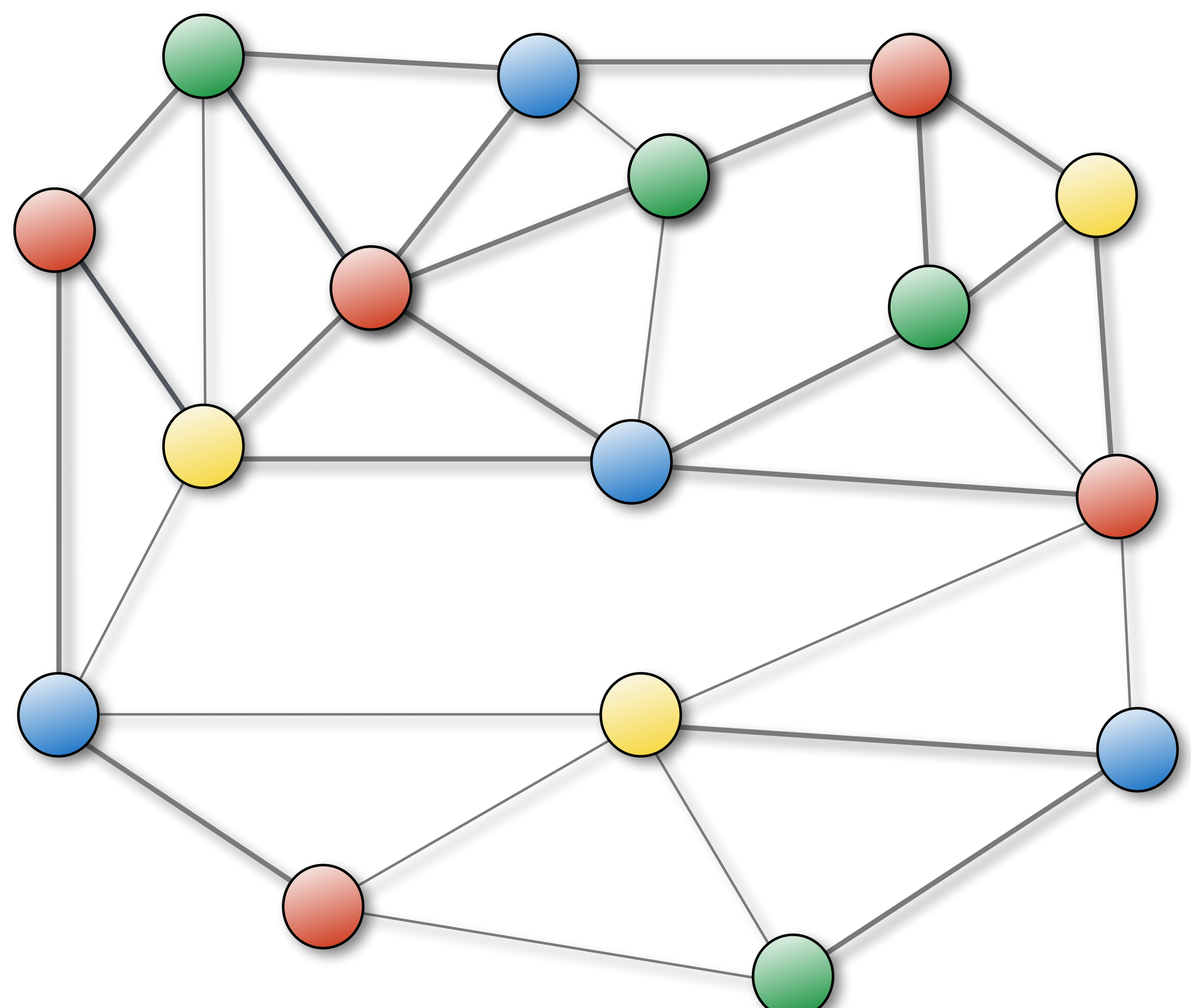
$$c(u) \neq c(v) \text{ für alle Kanten } \{u, v\} \in E.$$



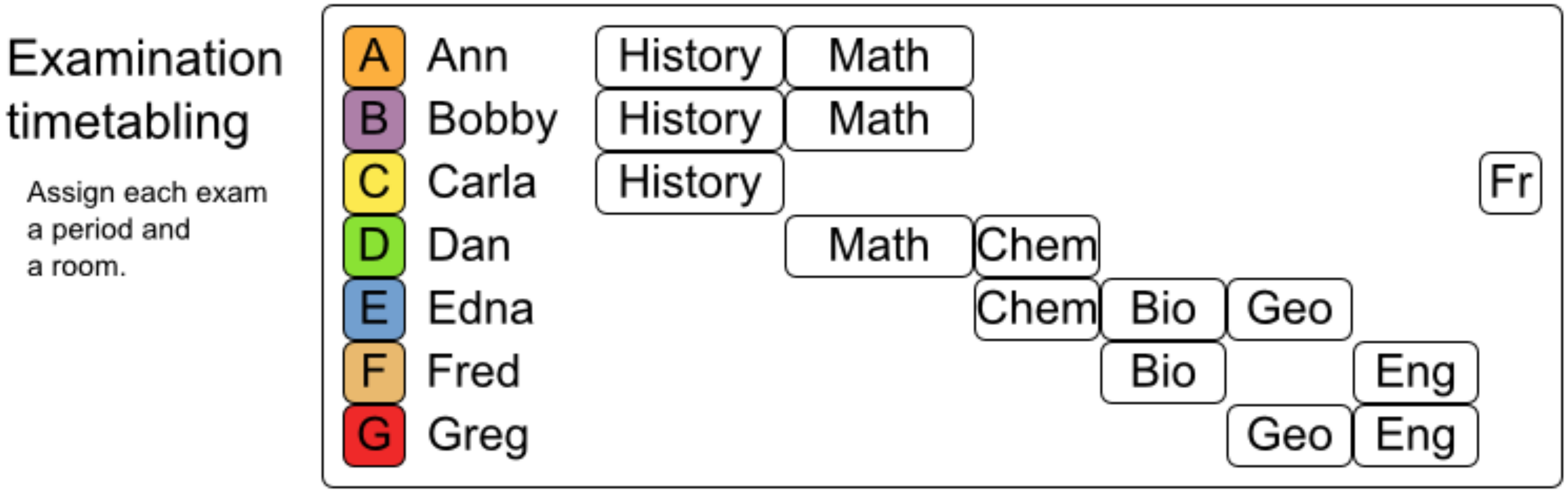
k-Färbung, chromatische Zahl

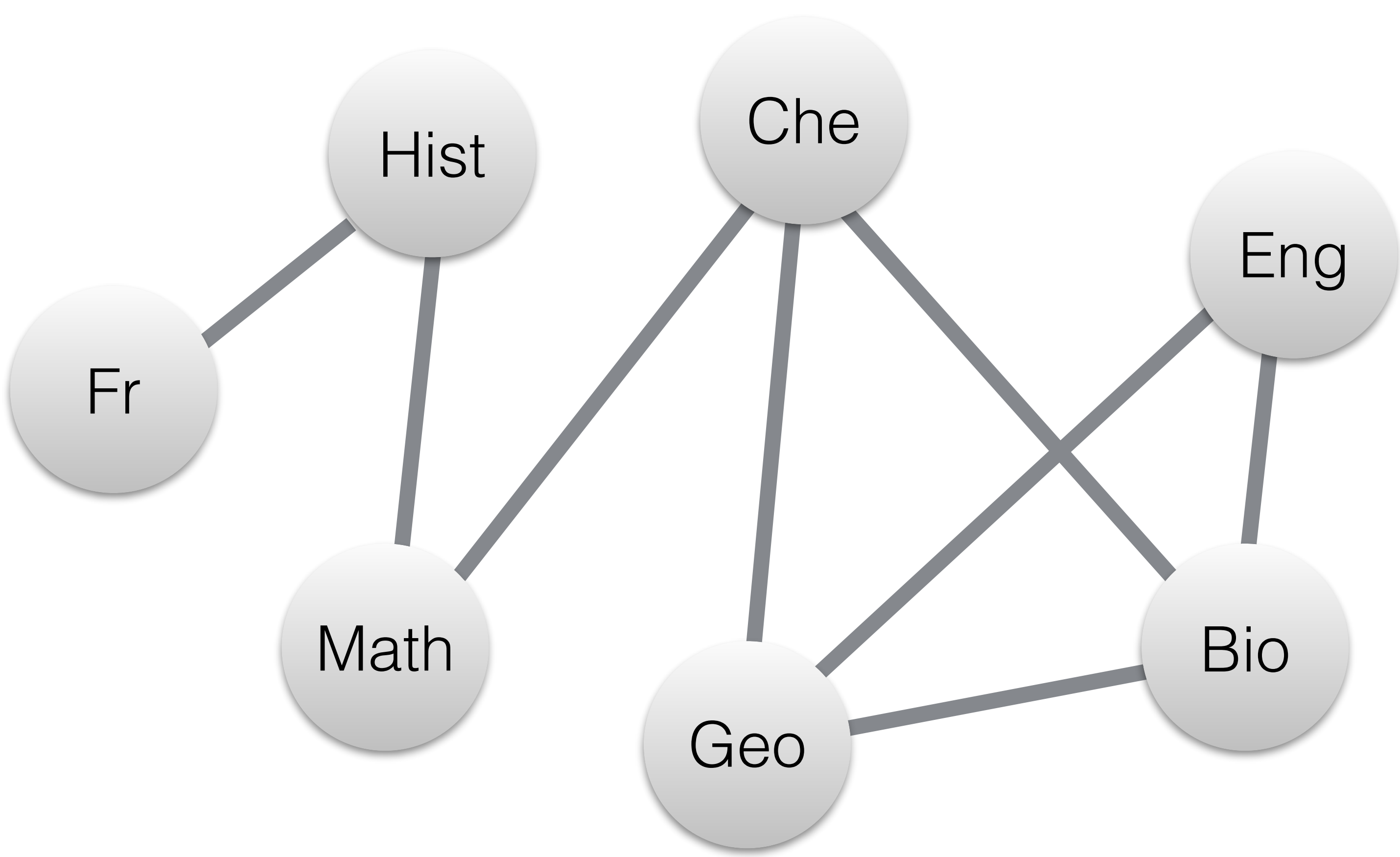
Eine (Knoten-)Färbung eines Graphen $G = (V, E)$ mit k Farben ist eine Abbildung $c : V \to [k]$, so dass gilt

$$c(u) \neq c(v) \text{ für alle Kanten } \{u, v\} \in E.$$

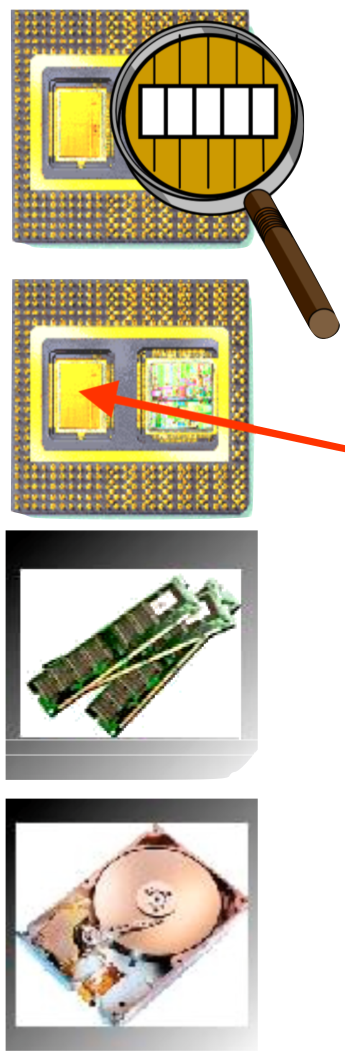


Graphfärbung - Prüfungsplan





Graphfärbung - Speicherallokation



Registers

Cache

Main memory

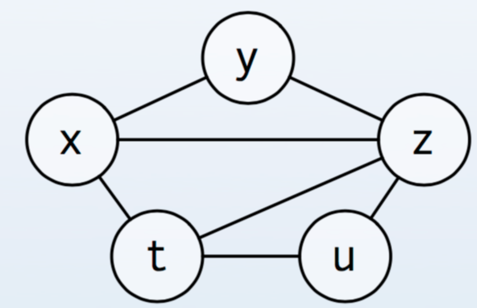
Disk

Interference graph example

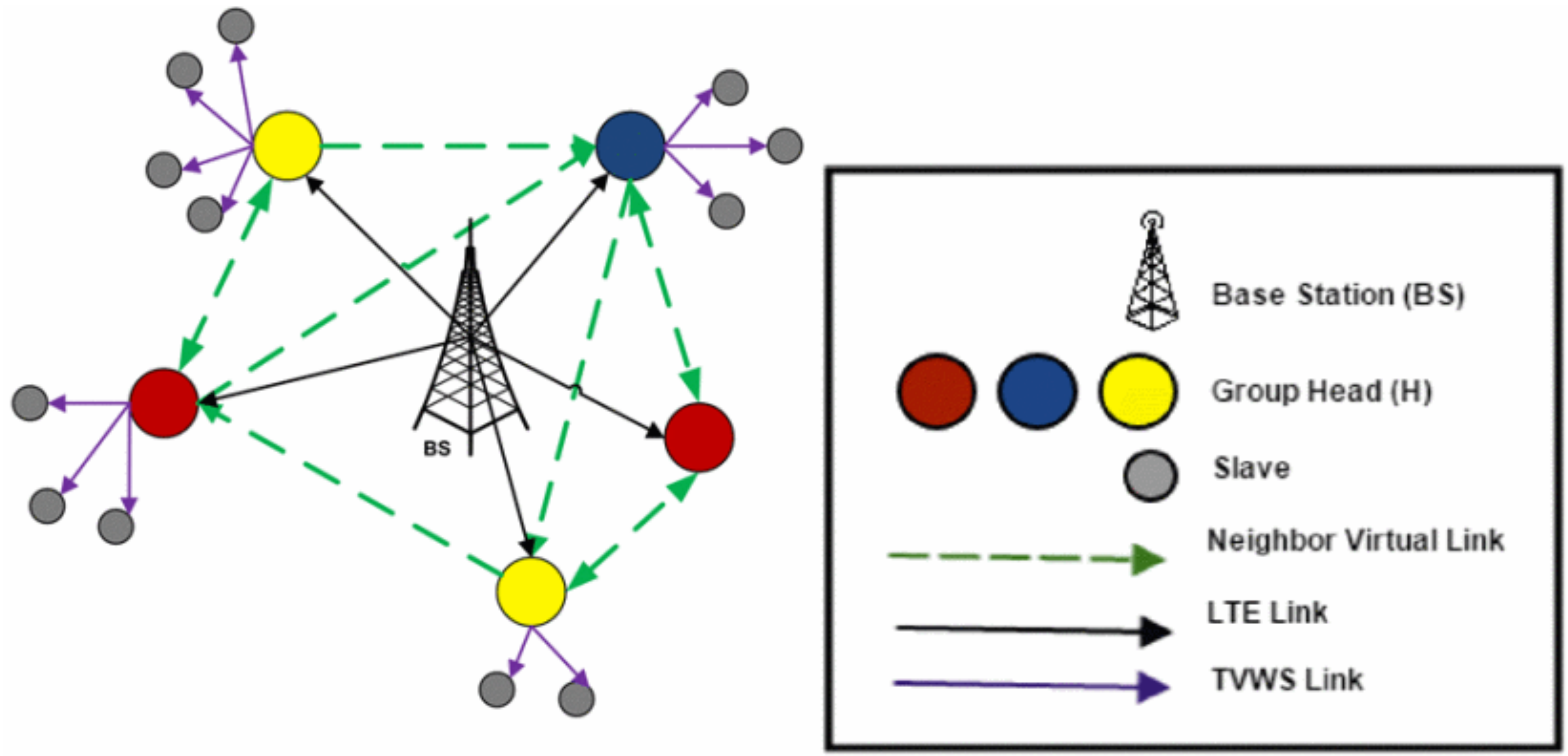
Live ranges

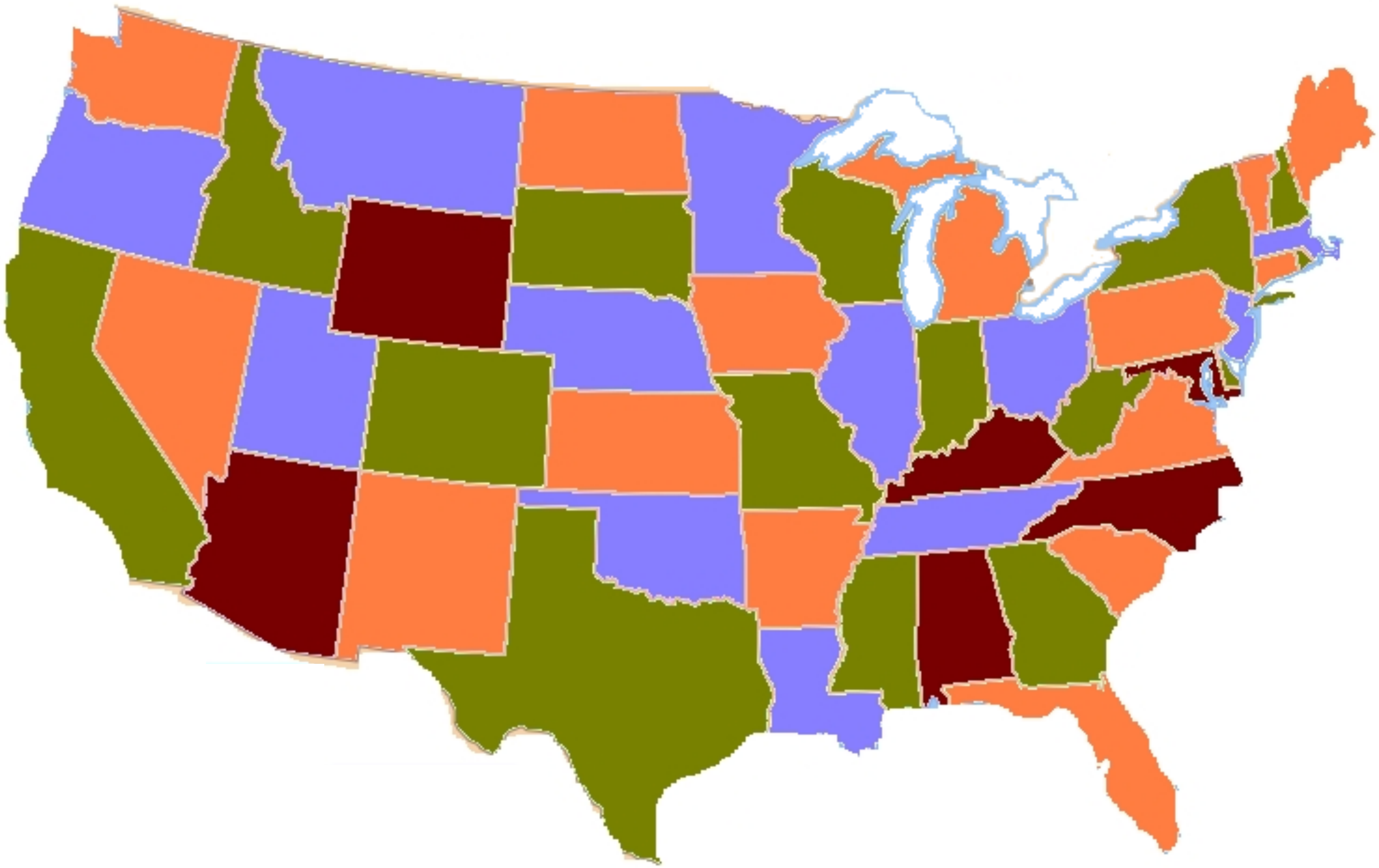
	x	y	z	t	u
x←1					
y←2					
z←x+y					
t←y					
u←x+t					
print z					
print t					
print u					

Interference graph



Graphfärbung - Funknetzwerke





4-Farben-Theorem

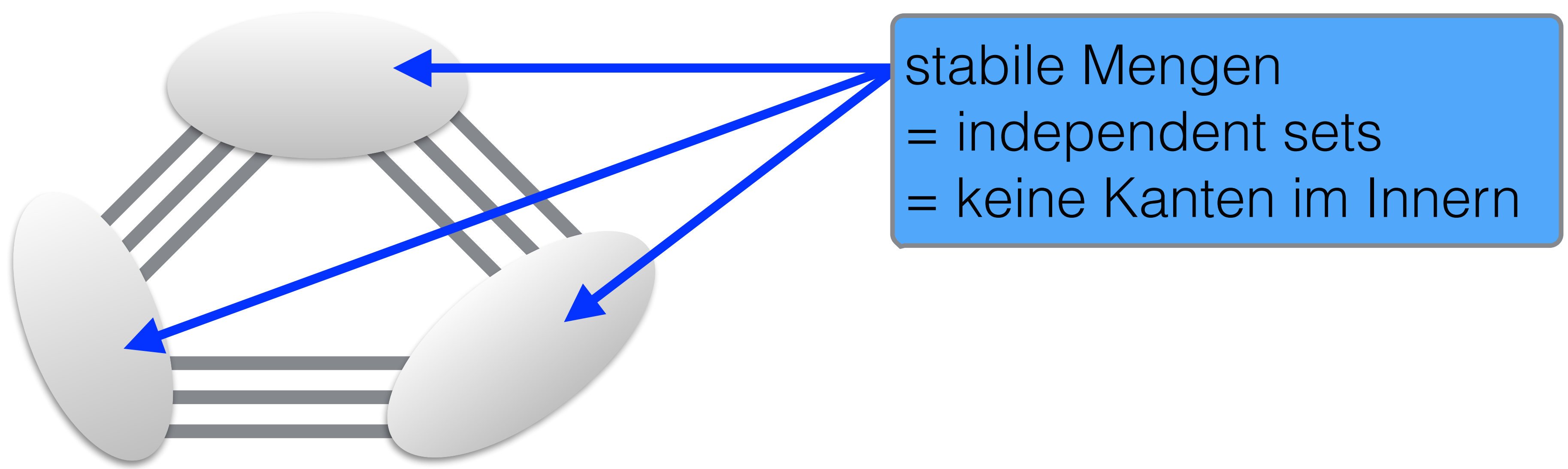
k-Färbung, chromatische Zahl

Eine (Knoten-)Färbung eines Graphen $G = (V, E)$ mit k Farben ist eine Abbildung $c : V \to [k]$, so dass gilt

$$c(u) \neq c(v) \text{ für alle Kanten } \{u, v\} \in E.$$

Die chromatische Zahl $\chi(G)$ ist die minimale Anzahl Farben, die für eine Knotenfärbung von G benötigt wird.

Äquivalente Formulierung: $\chi(G) \leq k$ gdw. G k-partit



Spezialfall: $\chi(G) \leq 2$ gdw. G bipartit

“Ist G bipartit?” kann man in Zeit $O(|E|)$ mit Breiten- oder Tiefensuche beantworten.

Färbbarkeit ist sehr schwer

Satz: Für jedes $k \geq 3$ ist das Problem

„Gegeben ein Graph $G = (V, E)$, gilt $\chi(G) \leq k$?“

NP-vollständig.

Alternativen?

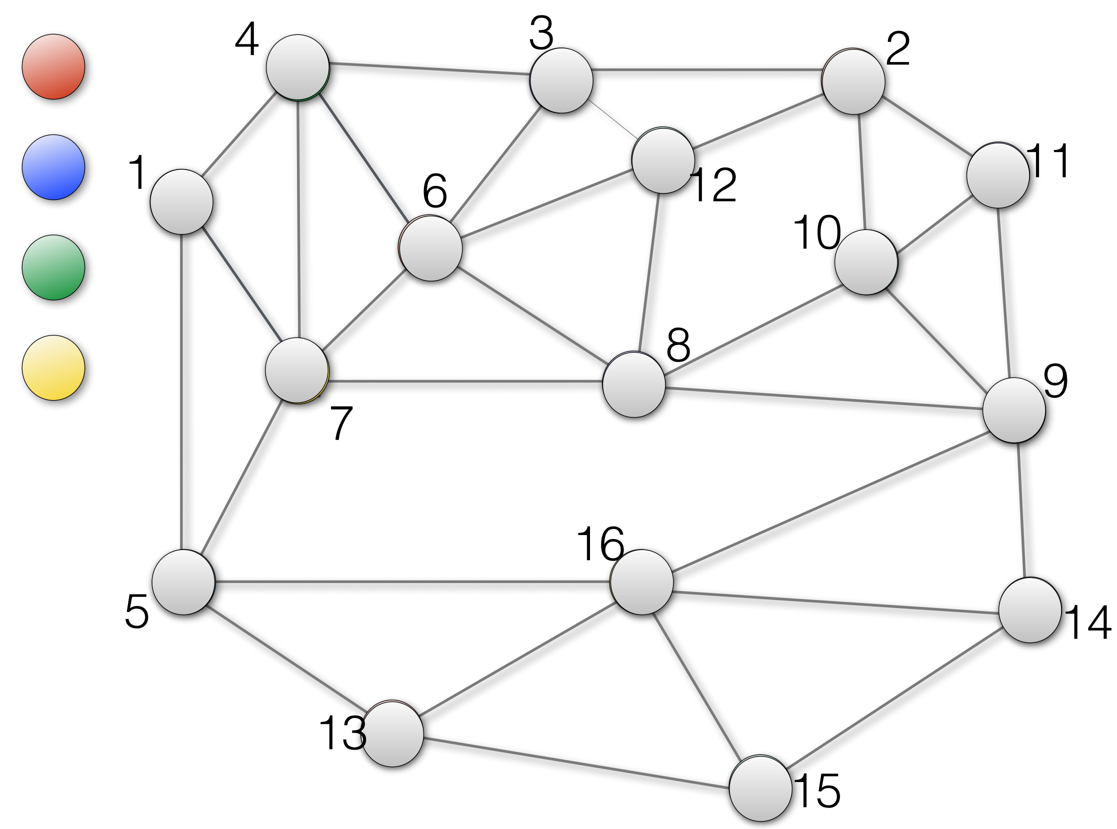
Exponentieller Algorithmus?	**Ja**, mit Inklusion/Exklusion. (Polynomieller Speicher und Zeit $O(2.3^n)$)
Approximationen?	**Nein.** Für jedes $\varepsilon > 0$ ist es NP-schwer, eine $n^{1-\varepsilon}$-Approximation zu finden.
Spezialfälle?	**Ja.** Wir werden einige Arten von Graphen sehen, für die es gute Algorithmen gibt.

Greedy-Algorithmus

GREEDY-FÄRBUNG (G)

wähle eine beliebige Reihenfolge der Knoten: $V = \{v_1, \ldots, v_n\}$
$c[v_1] \leftarrow 1$
for $i = 2$ **to** $i = n$ **do**
  $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c(u)$ für alle $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

Beobachtung:
Für jede Reihenfolge $V = \{v_1, \ldots, v_n\}$ der Knoten benötigt der Greedy-Algorithmus höchstens $\Delta(G)+1$ viele Farben.

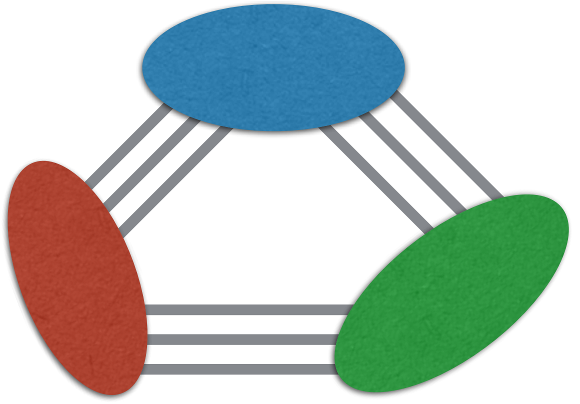


Notation:
$\Delta(G)$:= maximaler Grad in G

GREEDY-FÄRBUNG (G)

1: wähle eine beliebige Reihenfolge der Knoten: $V = \{v_1, \ldots, v_n\}$
2: $c[v_1] \leftarrow 1$
3: **for** $i = 2$ **to** $i = n$ **do**
4: $\quad c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c(u)$ für alle $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

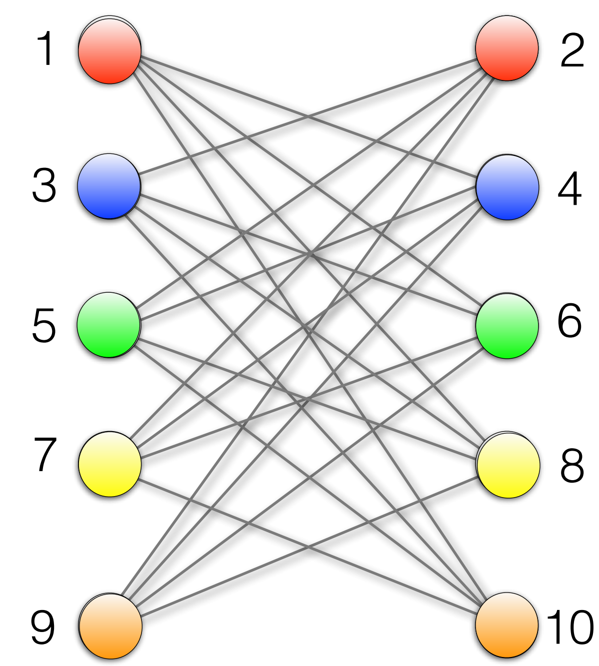
Beobachtung:
Es gibt eine Reihenfolge
$V = \{v_1, \ldots, v_n\}$ der Knoten,
für die der Greedy-Algorithmus
nur $\boldsymbol{\chi}(G)$ viele Farben benötigt.



Greedy-Färbung (G)

1: wähle eine beliebige Reihenfolge der Knoten: $V = \{v_1, \ldots, v_n\}$
2: $c[v_1] \leftarrow 1$
3: **for** $i = 2$ **to** $i = n$ **do**
4: $\quad c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c(u)$ für alle $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

Beobachtung:
Es gibt bipartite Graphen und eine Reihenfolge $V = \{v_1, \ldots, v_n\}$ der Knoten, für die der Greedy-Algorithmus |V|/2 viele Farben benötigt.



vollständig bipartiter Graph ohne ein perfektes Matching

GREEDY-FÄRBUNG (G)

1: wähle eine beliebige Reihenfolge der Knoten: $V = \{v_1, \ldots, v_n\}$
2: $c[v_1] \leftarrow 1$
3: **for** $i = 2$ **to** $i = n$ **do**
4: $\quad c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c(u)$ für alle $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

Beobachtung:
Gilt für die (gewählte) Reihenfolge $|N(v_i) \cap \{v_1, \ldots, v_{i-1}\}| \leq k \quad \forall\, 2 \leq i \leq n$,
dann benötigt der Greedy-Algorithmus höchstens **k+1** viele Farben.

Heuristik:
v_n := Knoten vom kleinsten Grad. Lösche v_n.
v_{n-1} := Knoten vom kleinsten Grad im Restgraph. Lösche v_{n-1}.
Iteriere.

Falls $G=(V,E)$ erfüllt:

In jedem Subgraphen gibt es einen Knoten mit Grad $\leq k$

$\Rightarrow$ Heuristik liefert Reihenfolge $v_1, \ldots, v_n$ für die der Greedy-Algorithmus höchstens $k+1$ Farben benötigt

Greedy-Algorithmus

Greedy-Färbung (G)

1: **wähle eine beliebige Reihenfolge der Knoten:** $V = \{v_1, \ldots, v_n\}$
2: $c[v_1] \leftarrow 1$
3: **for** $i = 2$ **to** $i = n$ **do**
4: $\quad c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c(u)$ **für alle** $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

Falls G=(V,E) erfüllt:

In jedem Subgraphen gibt es einen Knoten mit Grad $\leq k$

$\Rightarrow$ Heuristik liefert Reihenfolge $v_1,\ldots,v_n$ für die der Greedy-Algorithmus höchstens $k+1$ Farben benötigt

Korollar:
Die Heuristik findet immer eine Färbung mit 2 Farben für Bäume.

Satz: (ohne Beweis)
Ist ein Graph planar (kann überkreuzungsfrei in der Ebene gezeichnet werden), so gibt es immer einen Knoten vom Grad ≤ 5.

Korollar:
Die Heuristik findet eine Färbung mit ≤ 6 Farben für planare Graphen.

GREEDY-FÄRBUNG (G)

1: wähle eine beliebige Reihenfolge der Knoten: $V = \{v_1, \ldots, v_n\}$
2: $c[v_1] \leftarrow 1$
3: **for** $i = 2$ **to** $i = n$ **do**
4: $\quad c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c(u)$ für alle $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

Beobachtung:
Gilt für die (gewählte) Reihenfolge $|N(v_i) \cap \{v_1, \ldots, v_{i-1}\}| \leq k \quad \forall\, 2 \leq i \leq n$,
dann benötigt der Greedy-Algorithmus höchstens **k+1** viele Farben.

Heuristik:
v_n := Knoten vom kleinsten Grad. Lösche v_n.
v_{n-1} := Knoten vom kleinsten Grad im Restgraph. Lösche v_{n-1}.
Iteriere.

Korollar:

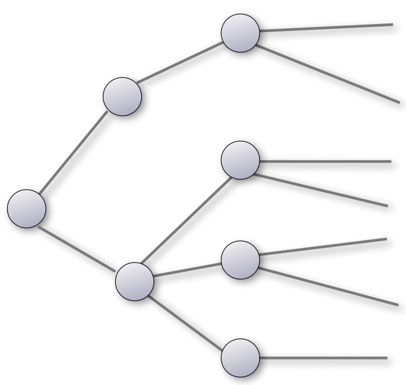
$G=(V,E)$ zshgd. und es gibt $v \in V$ mit $\deg(v) < \Delta(G)$

$\Rightarrow$ Heuristik (oder Breiten-/Tiefensuche) liefert Reihenfolge,
für die der Greedy-Algorithmus höchstens $\Delta(G)$ Farben benötigt

Warum ist Färbbarkeit so schwer?

Satz: ∀ k ∈ N, ∀ r ∈ N : Es gibt Graphen ohne einen Kreis mit Länge ≤ k, aber mit chromatischer Zahl ≥ r.

(Beweis im Skript für k=3)



Lokal sieht der Graph aus
wie ein Baum (alle Knoten,
die man von einem v aus
in k/2 Schritten erreichen kann).

Satz: Einen 3-färbbaren Graphen kann man in Zeit $O(|V|+|E|)$ mit $O(\sqrt{|V|})$ Farben färben.

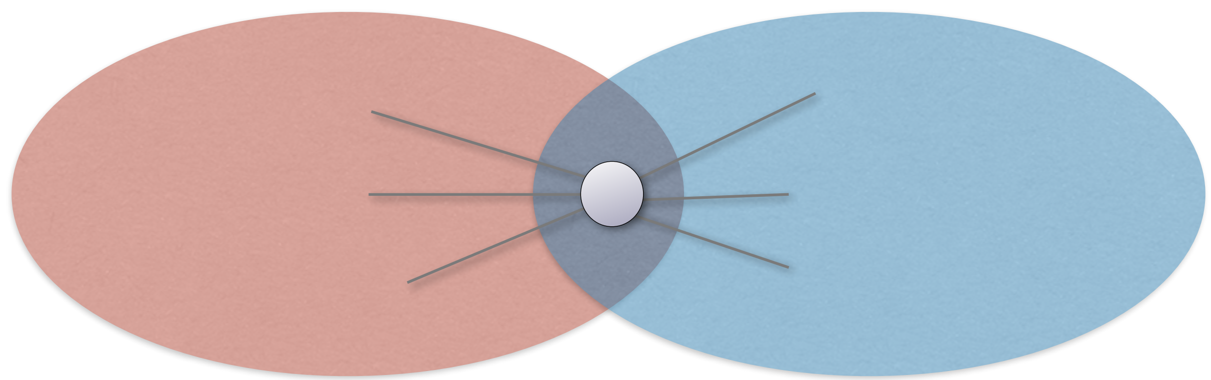
Algorithmus:

While es gibt Knoten v, der $> \sqrt{|V|}$ ungefärbte Nachbarn hat:

Färbe v mit neuer Farbe und seine Nachbarn mit 2 weiteren neuen Farben.

Lösche alle gefärbten Knoten. Der Restgraph hat Maximalgrad $\Delta \leq \sqrt{|V|}$.

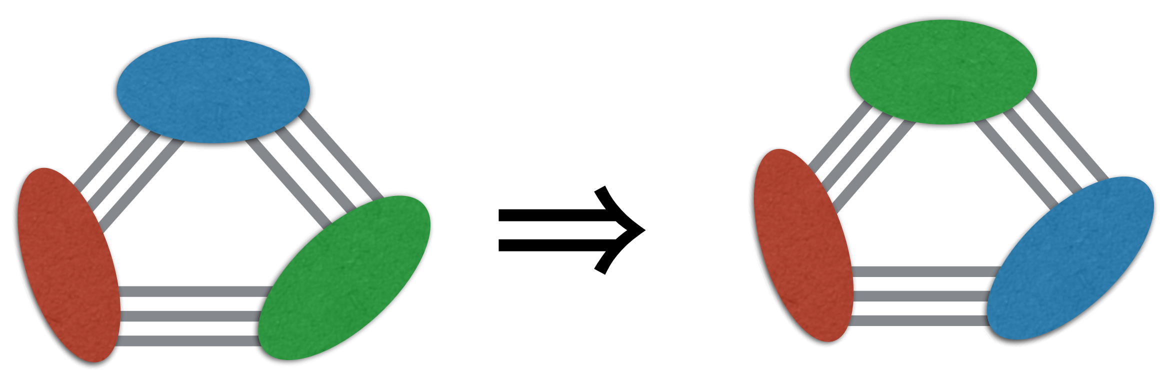
Färbe verbleibende Knoten mit Greedy-Algorithmus mit $\Delta + 1$ neuen Farben.



Ist G ein Graph, in dem man jeden Block mit k Farben färben kann.

Dann kann man auch G mit k Farben färben.

Farbklassen tauschen



Abhängigkeit vom maximalen Grad

GREEDY-FÄRBUNG (G)

1: wähle eine beliebige Reihenfolge der Knoten: $V = \{v_1, \ldots, v_n\}$
2: $c[v_1] \leftarrow 1$
3: **for** $i = 2$ **to** n **do**
4: $\quad c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c(u)$ für alle $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

Jeder Graph kann in Zeit $O(|E|)$ mit $\Delta(G)+1$ Farben gefärbt werden

Satz von Brooks

$G \neq K_n$, $G \neq C_{2n+1}$, G zshgd:

$\Rightarrow$ G kann in Zeit $O(|E|)$ mit $\Delta(G)$ Farben gefärbt werden

Satz von Brooks

$G \neq K_n$, $G \neq C_{2n+1}$, G zshgd:

$\Rightarrow$ G kann in Zeit $O(|E|)$ mit $\Delta(G)$ Farben gefärbt werden

Algorithmus:

- Falls $\Delta(G)=2$: färbe G mit zwei Farben

  (da G zshgd und kein ungerader Kreis, ist G ein Pfad oder ein gerader Kreis)
- Falls $\exists\, v \in V$ mit $deg(v) < \Delta(G)$: färbe G mit Greedy-Algorithms + Heuristik

  (benötigt nur $\Delta(G)$ Farben)
- Falls es einen Artikulationsknoten v gibt: färbe alle Blöcke (jeweils inkl. Knoten v) mit Heuristik; ggf. Farbtausch, damit v in allen Graphen einheitlich gefärbt

  (in allen diesen Blöcken hat v Grad < $\Delta(G)$, Heuristik funktioniert also wie oben argumentiert)
- Bestimme Knoten $v,x,y \in V$ mit $x,y \in N(v)$ und $\{x,y\} \notin E$

  (diese existieren, da G zshgd und kein vollständiger Graph)
- Betrachte $G' := G[V \setminus \{x,y\}]$:
  - *Falls G' zshgd:* färbe G mit Greedy-Alg: Erst x und y, danach Heuristik für G'
  - *Falls G' nicht zshgd:* …