Universität Paderborn
Fakultät für Elektrotechnik, Informatik und Mathematik
Arbeitsgruppe Algorithmen und Komplexität

Seminar
„Perlen der theoretischen Informatik“
WS 2004/05

Graphenfärbung in erwartet konstanter Zeit

von

Dennis Hannwacker
hanni@uni-paderborn.de
Matr.-Nr. 6146900

betreut durch

Dr. rer. nat. Christian Schindelhauer

vorgelegt bei

Prof. Dr. math. Friedhelm Meyer auf der Heide

Paderborn, 3. Januar 2005

Inhaltsverzeichnis

1 Einleitung

Versetzen Sie sich in eine Zeit zurück, in der die Drucktechnologie noch nicht so ausgereift war, wie das heute der Fall ist. Sie haben eine Anstellung als Drucker gefunden und haben gerade die Aufgabe bekommen, die in Abb. 1.1 dargestellte Landkarte zu vervielfältigen.

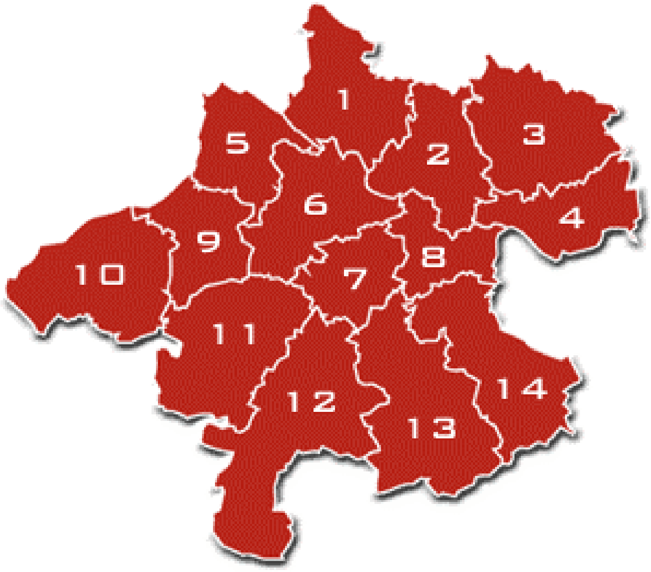

Abb. 1.1: Landkarte

Damit jedes Land sich von seinen Nachbarländern klar abhebt, sollen angrenzende Länder unterschiedlich eingefärbt werden. Leider stehen ihnen nur vier Farben zur Verfügung. Das sind zwar doppelt so viele wie Gutenberg zur Auswahl standen, aber reichen sie aus, um jedes Land einzufärben, ohne die eben erwähnte Nachbarschaftsregel zu verletzen?

Diese Problemstellung lässt sich sehr einfach und anschaulich in die Sprache der Graphentheorie übersetzen. Eine Landkarte mit n Ländern wird dabei durch einen Graphen G mit n Knoten modelliert, wobei die Knoten des Graphen die Länder der Landkarte repräsentieren. Sind zwei Länder benachbart, so werden die entsprechenden Knoten im Graphen durch eine Kante verbunden. Die Kantenmenge modelliert somit die Nachbarschaftsrelationen zwischen den Ländern.

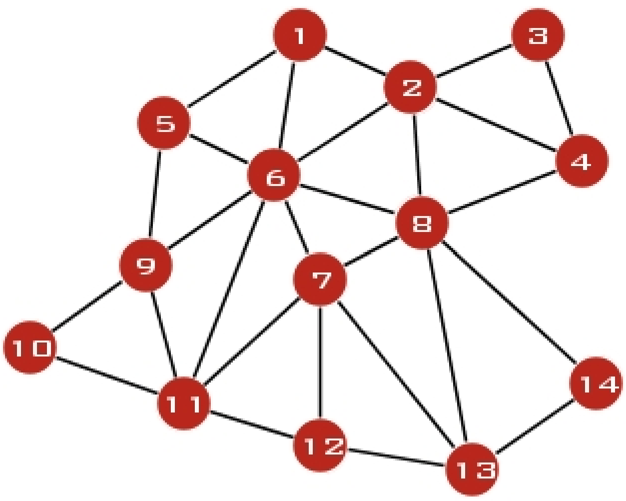

Abb. 1.2: Modellierung der Landkarte durch einen Graphen

Da die Kanten eine Relation beschreiben, an der die verbundenen Knoten in gleicher Art und Weise beteiligt sind, werden zur Beschreibung des Problems ungerichtete Kanten verwendet. Die zusätzliche Information einer gerichteten Kante wäre überflüssig und ließe sich nicht anhand des Ausgangsproblems begründen.

Die Frage, ob eine Landkarte mit k Farben einfärbbar ist, ohne dass Nachbarländer dieselbe Farbe erhalten, formuliert sich in der Sprache der Graphentheorie somit wie folgt:
Gibt es eine Färbung der Knoten des Graphen G mit k Farben, so dass alle Knoten, die durch eine Kante verbunden sind, unterschiedlich gefärbt sind?
Dieses Entscheidungsproblem formuliert man auch kurz: Ist der Graph G k-färbbar?

Die praktische Relevanz von der Reduzierung von Farben bei der Einfärbung von Landkarten dürfte heutzutage eher eine untergeordnete Rolle spielen. Doch diese anschauliche Problemstellung ist nicht die einzige, die sich als Graphfärbungsproblem formulieren lässt. Vielmehr ist das Problem der k-Färbung eines Graphen nicht nur eines der ältesten graphentheoretischen Probleme, sondern auch eines der ersten, die als NP-vollständig klassifiziert wurden.

Schon aufgrund der Definition der NP-Vollständigkeit gilt, dass sich jedes andere beliebige Problem aus NP in polynomieller Zeit auf das Graphfärbungsproblem reduzieren lässt.
Es lassen sich also auch Probleme in ein Färbungsproblem übersetzen, die auf den ersten Blick gar keine Gemeinsamkeiten mit diesem aufweisen.

Zu diesen umformulierbaren Problemen gehören beispielsweise die Stundenplanprobleme. Ein konkretes Stundenplanproblem ist zum Beispiel die Erstellung des Stundenplans einer Schule, bei der Randbedingungen beachtet werden müssen, wie etwa dass jeder Lehrer pro Zeitfenster nur eine Klasse unterrichten oder dass jede Klasse pro Zeiteinheit nur an einer Unterrichtsveranstaltung teilnehmen kann.

Jeder Knoten des Graphen steht dabei stellvertretend für eine im Stundenplan unterzubringende Lehrveranstaltung, die von einem bestimmten Lehrer gehalten und von einer festgelegten Klasse besucht werden muss. Veranstaltungen, die aus den bereits genannten Gründen nicht gleichzeitig stattfinden können, werden dabei durch Kanten verbunden. Die möglichen Farben entsprechen den zur Verfügung stehenden Zeitfenstern. Wenn es einem jetzt gelingt, diesen Graphen so zu färben, dass durch Kanten verbundene Knoten unterschiedliche Farben erhalten, so hat man einen Stundenplan gefunden, der keine Konflikte aufweist.

Die Frage nach der k-Färbbarkeit eines Graphen bleibt also interessant und es lohnt sich, sich weiterhin mit dem Problem zu befassen. Zentraler Bestandteil dieser Ausarbeitung wird dabei die Vorstellung und Analyse eines Backtracking-Algorithmus sein, der für ein festes k und einen beliebigen Graphen G die Frage nach der k-Färbbarkeit von G mit einer Laufzeit beantworten kann, die im average case durch eine Konstante beschränkt ist. Ein Algorithmus also, von dem man im Normalfall erwartet, dass er das Problem für ein festes k in konstanter Zeit lösen kann.

Dies wurde von Herbert Wilf in seinem Buch [Wil87] im letzten Abschnitt gezeigt und von Christian Schindelhauer in seinem Vorlesungsskript [Sch04] aufgegriffen. Beide Arbeiten dienten als Grundlage dieser Ausarbeitung.

2 Definitionen

2.1 Definition: Ungerichteter Graph

Ein ungerichteter Graph ist ein Tupel $G = (V,E)$ mit den wie folgt definierten Komponenten:
V ist eine Menge von Knoten
$E \subseteq \{\{v,w\} \mid v,w \in V \text{ mit } v \neq w\}$ ist eine Menge von Kanten

Im Folgenden betrachten wir immer ungerichtete Graphen, deren Knotenmengen von der Form $V=\{1,\ldots,n\}$, $n \in \mathbb{N}$ sind, also aus Knoten bestehen, die mit den natürlichen Zahlen von 1 bis n beschriftet sind.

Das in der Einleitung bereits angesprochene Problem der k-Färbbarkeit eines Graphen definiert sich mathematisch nun wie folgt, wobei die Farben durch die natürlichen Zahlen von 1 bis k repräsentiert werden:

2.2 Definition: *k*-Färbbarkeit

Sei $k \in \mathbb{N}$ bel.
Ein ungerichteter Graph $G = (V,E)$ heißt k-färbbar :⇔
$\exists f : V \to \{1,\ldots,k\} : \forall \{v,w\} \in E : f(v) \neq f(w)$

Eine Färbung f des Graphen G mit k Farben, die über eine Kante verbundene Knoten unterschiedlich färbt und somit ein Beleg für die k-Färbbarkeit des Graphen G ist, nennen wir auch gültige oder konfliktfreie Färbung von G mit k Farben.

Nun kann es für einen Graphen G mehrere Möglichkeiten geben, ihn gültig mit k Farben zu färben. Die Funktion, die uns für einen gegebenen Graphen G in Abhängigkeit von k die Anzahl der gültigen Färbungen von G zurückgibt, nennen wir chromatisches Polynom von G.

2.3 Definition: Chromatisches Polynom

Sei G ein ungerichteter Graph.
Die Funktion $P_G: \mathbb{N} \to \mathbb{N}$ mit $P_G(k)$ = Anzahl der gültigen Färbungen von G mit k Farben heißt chromatisches Polynom von G.
Dass es sich bei der Funktion P_G tatsächlich um ein Polynom in k handelt, wurde von Birkhoff 1912 gezeigt und lässt sich in [Bir12] nachlesen.

Ein graphentheoretischer Begriff, der uns im Folgenden noch häufig begegnen wird, ist der des induzierten Teilgraphen, deshalb will ich ihn an dieser Stelle noch einmal in Erinnerung rufen. Ausgehend von einem Graphen G und einer Teilmenge seiner Knoten L nennt sich der eindeutig bestimmte Teilgraph, der aus der Knotenmenge L und genau den Kanten aus G besteht, die zwischen den Knoten aus L verlaufen, der durch L induzierte Teilgraph von G.

2.4 Definition: Induzierter Teilgraph

Sei $G = (V,E)$ ein ungerichteter Graph und $L \subseteq V$.
$E' := \{\{v,w\} \mid v,w \in L \wedge \{v,w\} \in E\}$
Der Graph $H_L(G) := (L, E')$ heißt der durch L induzierte Teilgraph von G.

Sei $G = (V,E)$ ein ungerichteter Graph mit $V=\{1,\ldots,n\}$, $n \in \mathbb{N}$.
Für eine natürliche Zahl $d \in \mathbb{N}$ mit $d \leq n$ bezeichne $H_d(G)$ den Teilgraphen von G, der durch die Knotenmenge $\{1,\ldots,d\}$ induziert wird.

3 Backtracking-Algorithmus

Eine gültige Färbung mit *k* Farben für einen Graphen lässt sich in folgender Weise finden: Man beginnt mit dem ersten Knoten und färbt ihn mit Farbe eins. Knoten zwei wird ebenfalls in der ersten Farbe eingefärbt, es sei denn, Knoten eins und zwei sind durch eine Kante verbunden. Sollte dies der Fall sein, so verwendet man Farbe zwei.

Beim dritten Knoten versucht man zuerst, genauso wie bei jedem anderen Knoten auch, erneut die erste Farbe zu verwenden. Sollte dabei ein Konflikt mit den zwei zuvor bereits eingefärbten Knoten entstehen, so wird schrittweise die Farbnummer erhöht, bis keine Konflikte mehr vorliegen.

Es wird also für jeden Knoten die niedrigste verfügbare Farbe verwendet, die keinen Konflikt mit bereits eingefärbten Knoten verursacht. Dieses Vorgehen kann uns in eine Sackgasse führen, in der für den aktuell zu färbenden Knoten keine Farbe mehr existiert, die ihn konfliktfrei zu den bereits verarbeiteten Knoten einfärben würde.

Betrachten wir dazu einfach mal folgendes Beispiel. Wir versuchen den in Abb. 3.1 dargestellten Graphen mit zwei Farben zu färben.

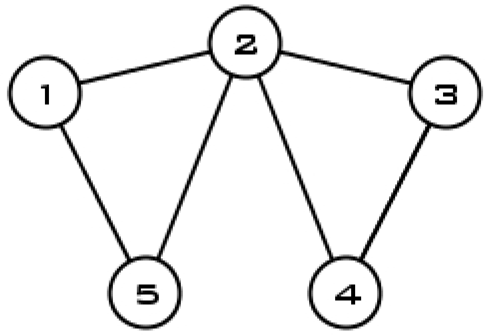

Abb. 3.1: Beispielgraph

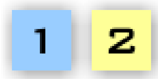

Abb. 3.2: Farbpalette mit 2 Farben

Nachdem die ersten drei Knoten in der oben beschriebenen Vorgehensweise eingefärbt wurden, sieht der Graph so aus, wie es Abb. 3.3 illustriert. Der als nächstes zu bearbeitende vierte Knoten, lässt sich jetzt aber weder mit der blauen noch mit der gelben Farbe einfärben, ohne einen Konflikt zu erzeugen.

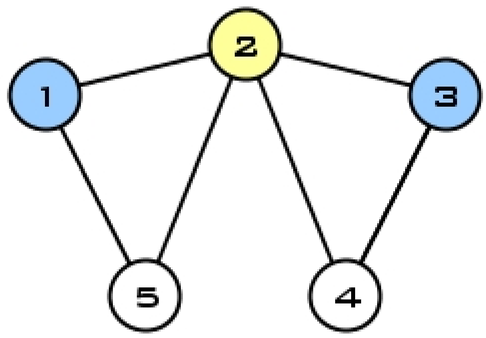

Abb. 3.3: Sackgasse Nr.1

Steckt man in einer solchen Sackgasse, so werden alle Schritte bis zu dem zuletzt gefärbten Knoten rückgängig gemacht, für den noch andere Farben zur Verfügung stehen und ersetzt seine aktuelle Farbe durch die nächst niedrigste. Ausgehend von dieser veränderten Situation versucht man erneut die weiteren Knoten einzufärben.

In unserem Beispiel würde dies bedeuten, zu Knoten eins zurückzugehen und statt der blauen die gelbe Farbe zu verwenden. Von dieser neuen Situation ausgehend bleibt für Knoten zwei nur die blaue und für den dritten Knoten nur die gelbe Farbe. Wir würden erneut in einer Sackgasse landen, wie sie sich uns in Abb.3.4 darstellt.

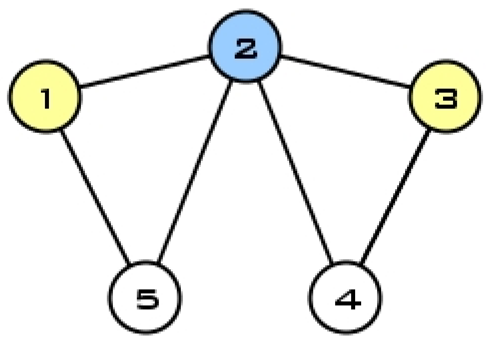

Abb. 3.4: Sackgasse Nr.2

Ein Verfahren, das in dieser Art und Weise die Möglichkeiten durchspielt, nennt man auch Backtracking-Verfahren.

Der Versuch, den Graphen aus Abb. 3.1 mit zwei Farben zu färben, lässt sich auch in Form eines Suchbaumes darstellen, der von dem Backtracking-Algorithmus durchlaufen wird.

Die Suche nach einer gültigen Färbung beginnt dabei in der Wurzel des Baumes. Sie repräsentiert die Ausgangssituation, in der noch kein Knoten des Graphen gefärbt wurde. Dagegen stehen Knoten unterhalb der Wurzel für Situationen, in denen der Algorithmus bereits Knoten des Graphen gefärbt hat und zwar jeweils mit den in der Beschriftung angegebenen Farben.

Die nach ihren Nummern aufsteigend sortierten Knoten des Graphen werden dabei mit den in der Beschriftung angegebenen Farben gefärbt, indem diese von links nach rechts gelesen der Reihe nach zugeordnet werden. Bei einer Beschriftung mit nur drei Farbnummern erhalten daher auch nur die ersten drei Knoten eine Farbe.

Gültige Färbungen eines Graphen mit *n* Knoten findet man im zugehörigen Suchbaum folglich als Blätter wieder, die mit *n* Farbnummern beschriftet sind. Wie man also leicht am Suchbaum in Abb. 3.5 erkennen kann, ist der Graph aus Abb. 3.1 nicht 2-färbbar.

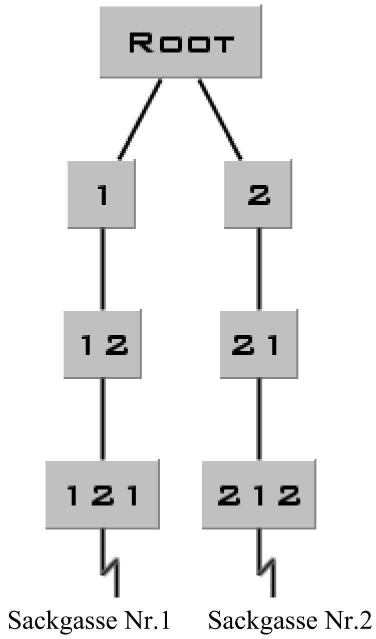

Abb. 3.5: Suchbaum für 2 Farben

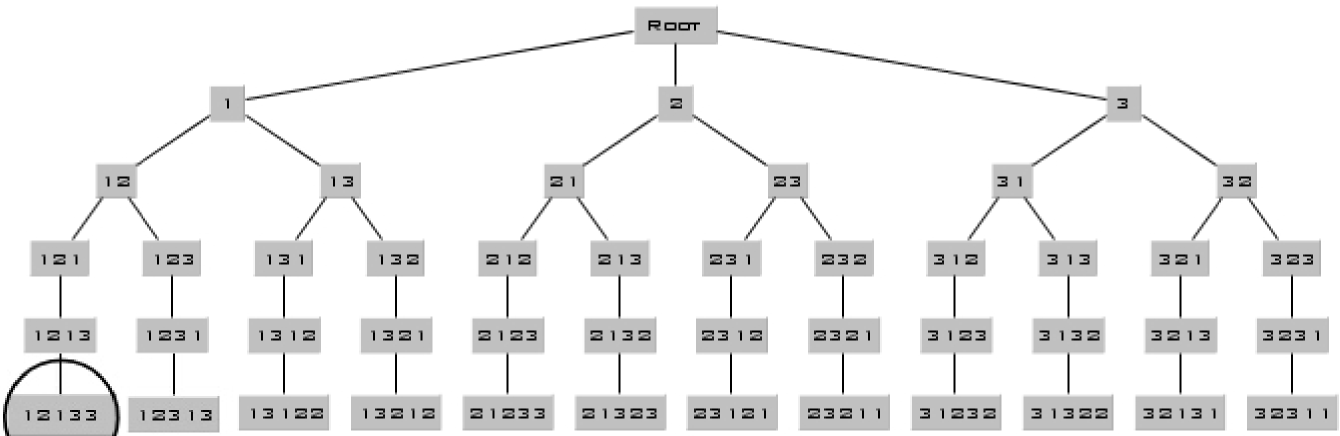

Abb. 3.6: Suchbaum für 3 Farben

Erweitert man aber die Farbpalette auf drei Farben, so existieren zwölf gültige Färbungen für unseren Graphen, wie der entsprechende Suchbaum aus Abb. 3.6 aufzeigt. Eine dieser Möglichkeiten, den Graphen gültig zu färben, wird in Abb. 3.7 dargestellt.

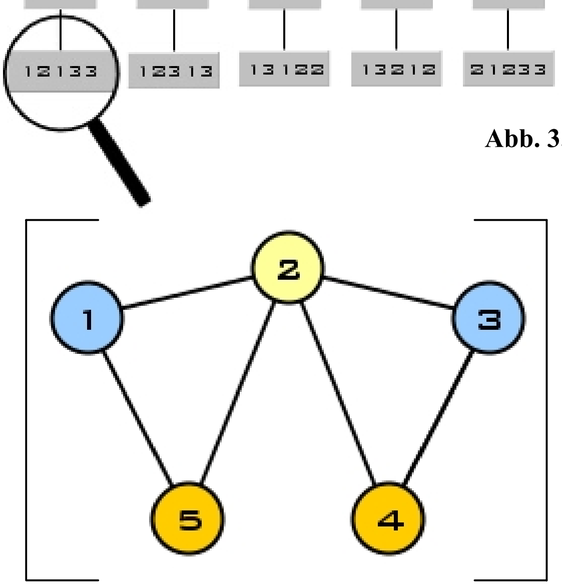

Abb. 3.7: Gültige Färbung mit 3 Farben

Abb. 3.8: Farbpalette mit 3 Farben

```
Proc valid (G, f, node)
    begin
        for every {i, node} ∈ E with i < node do
            if f [i] = f [node] then return false
        od
        return true
    end
```

```
Proc coloring (G, k)
    begin
        node ← 1
        color ← 1
        while (1 ≤ node ≤ n) do
            f [node] ← color
            if valid (G, f, node) then
                node ← node + 1
                color ← 1
            else
                if color < k then
                    color ← color + 1
                else
                    repeat
                        node ← node – 1
                        if node ≥ 1 then
                            color ← f [node] + 1
                        fi
                    until (color ≤ k ∨ node < 1)
                fi
            fi
        od
        if node = n + 1 then
            return G is k-colorable
        else
            return G is not k-colorable
        fi
    end
```

Abb. 3.9: Der Backtracking-Algorithmus

Der Algorithmus aus Abb. 3.9 löst das Entscheidungsproblem der *k*-Färbbarkeit eines ungerichteten Graphen $G = (V,E)$ und nutzt dafür das bereits besprochene Backtracking-Verfahren. Die Hilfsprozedur valid überprüft dabei, ob der aktuelle Knoten node konfliktfrei zu den bereits eingefärbten Knoten gefärbt wurde.

4 Laufzeitanalyse

Für manche Graphen kann das Entscheidungsproblem der k-Färbbarkeit sehr schnell gelöst werden, da die zugehörigen Backtracking-Suchbäume sehr klein ausfallen. Es existieren aber auch Graphen, deren Suchbäume ganz im Gegenteil sehr viele Knoten besitzen.

Wir werden nun zeigen, dass die durchschnittliche Größe eines solchen Suchbaumes, und damit die Laufzeit im average case, für ein festes k durch eine Konstante beschränkt ist.

Betrachten wir beispielsweise den Fall $k = 3$. Schon die Frage nach der 3-Färbbarkeit eines Graphen ist bereits ein NP-vollständiges Problem. Trotzdem werden wir zeigen, dass die durchschnittliche Knotenanzahl des Backtracking-Suchbaumes für beliebig große Graphen konstant ist. Das ist kein Widerspruch zu der NP-Vollständigkeit des Problems, denn es wird nicht behauptet, dass die Laufzeit für alle Graphen konstant ist, sondern, dass dies im average case der Fall ist. Man kann also für einen zufälligen Graphen eine konstante Laufzeit erwarten, sie wird einem aber keinesfalls garantiert.

Unser Vorhaben stellt sich mathematisch formuliert in folgender Weise dar:
Sei $\mathcal{G}_n$ die Menge aller ungerichteten Graphen mit n Knoten.
Sei $A(n,k)$ die durchschnittliche Anzahl von Knoten im Backtracking-Suchbaum für das Problem der k-Färbbarkeit eines Graphen mit n Knoten.
Es gilt also:

$$A(n,k) = \frac{1}{|\mathcal{G}_n|} \sum_{G \in \mathcal{G}_n} \left(\text{Anzahl der Knoten im Suchbaum für } G\right)$$

Es bleibt noch zu zeigen, dass $A(n,k)$ durch eine Konstante beschränkt ist, die nur von k abhängig ist und somit für alle $n \in \mathbb{N}$ gilt.
Dazu benötigen wir aber noch ein paar vorbereitende Lemmas, die uns bei der Abzählung der Knoten im Suchbaum helfen werden.

Lemma 4.1: Für alle $s_1, \ldots, s_k \in \mathbb{N}$ mit $\sum_{i=1}^{k} s_i = d$ gilt: $\sum_{i=1}^{k} s_i^2 \geq \frac{d^2}{k}$

Beweis:

$$\begin{aligned}
0 &\leq \sum_{i=1}^{k} \left(s_i - \tfrac{d}{k}\right)^2 && \Big|\ \text{Bem.}: \forall x \in \mathbb{R} : x^2 \geq 0 \\
&= \sum_{i=1}^{k} \left(s_i^2 - 2\, s_i \tfrac{d}{k} + \tfrac{d^2}{k^2}\right) \\
&= \sum_{i=1}^{k} s_i^2 - 2 \tfrac{d}{k} \sum_{i=1}^{k} s_i + \sum_{i=1}^{k} \tfrac{d^2}{k^2} && \Big|\ \text{Bem.}: \sum_{i=1}^{k} s_i = d \ \text{(Vor.)} \\
&= \sum_{i=1}^{k} s_i^2 - 2 \tfrac{d^2}{k} + k \tfrac{d^2}{k^2} \\
&= \sum_{i=1}^{k} s_i^2 - \tfrac{d^2}{k}
\end{aligned}$$

q.e.d.

Bis jetzt haben wir uns bezüglich des Färbungsproblems eigentlich immer die Frage gestellt, wie viele gültige Färbungen ein vorgegebener Graph haben kann. Das folgende Lemma dreht diese Fragestellung um, und beschäftigt sich demzufolge mit der Anzahl der Graphen, für die eine vorgegebene Färbung gültig ist.

Lemma 4.2:
Sei V=$\{1,\ldots,d\}$, $d \in \mathbb{N}$ eine Menge von Knoten.
Sei $f : \mathrm{V} \to \{1,\ldots,k\}$, $k \in \mathbb{N}$ eine Färbung dieser Knoten.

Dann gibt es höchstens $2^{\frac{d^2}{2}(1-\frac{1}{k})}$ Graphen, die V als Knotenmenge haben und für die *f* eine gültige Färbung ist.

Beweis:
Sei s_i die Anzahl der Knoten, die von *f* mit der Farbe i gefärbt werden.
$\forall i \in \{1,\ldots,k\}: \; s_i := |T_i|$ mit $T_i = \{\mathrm{node} \in \mathrm{V} \mid f(\mathrm{node}) = i\}$

Es gilt selbstverständlich: $\sum_{i=1}^{k} s_i = d$

Da für die Graphen, die wir zählen wollen, *f* eine gültige Färbung sein muss, können in diesen Graphen nur Kanten zwischen Knoten existieren, die durch *f* unterschiedlich gefärbt werden. Die Färbung *f* zerlegt also die Knotenmenge V in *k* Teilmengen T_i, in denen jeweils die Knoten enthalten sind, die von *f* mit der Farbe i gefärbt werden. Kanten dürfen jetzt nur noch zwischen Knoten existieren, die aus unterschiedlichen Teilmengen stammen. Die maximale Anzahl von Kanten, die ein Graph haben darf, berechnet sich mithilfe der Kardinalitäten s_i somit wie folgt: $s_1 s_2 + s_1 s_3 + \ldots + s_1 s_k + s_2 s_3 + \ldots + s_2 s_k + \ldots + s_{k-1} s_k$

Für diesen Ausdruck haben wir folgende Abschätzung:

$$
\begin{aligned}
\sum_{\substack{i,j=1 \\ i<j}}^{k} s_i s_j &= \tfrac{1}{2} \sum_{\substack{i,j=1 \\ i \neq j}}^{k} s_i s_j \\
&= \tfrac{1}{2} \Big(\sum_{i,j=1}^{k} s_i s_j - \sum_{i=1}^{k} s_i s_i \Big) \\
&= \tfrac{1}{2} \Big(\sum_{i=1}^{k} \sum_{j=1}^{k} s_i s_j - \sum_{i=1}^{k} s_i^2 \Big) \\
&= \tfrac{1}{2} \Big(\sum_{i=1}^{k} \big(s_i \sum_{j=1}^{k} s_j \big) - \sum_{i=1}^{k} s_i^2 \Big) \\
&= \tfrac{1}{2} \Big(\big(\sum_{j=1}^{k} s_j \big) \big(\sum_{i=1}^{k} s_i \big) - \sum_{i=1}^{k} s_i^2 \Big) \\
&= \tfrac{1}{2} \Big(\sum_{i=1}^{k} s_i \Big)^2 - \tfrac{1}{2} \sum_{i=1}^{k} s_i^2 \quad \Big|\ \text{Bem.}: \sum_{i=1}^{k} s_i = d \ \ (\text{Vor.}) \wedge \sum_{i=1}^{k} s_i^2 \geq \tfrac{d^2}{k} \ \ (\text{Lemma 4.1}) \\
&\leq \tfrac{1}{2} d^2 - \tfrac{1}{2} \tfrac{d^2}{k} \\
&= \tfrac{d^2}{2} \big(1 - \tfrac{1}{k} \big)
\end{aligned}
$$

Ein Graph, für den *f* eine gültige Färbung ist, kann also maximal $\frac{d^2}{2}\left(1-\frac{1}{k}\right)$ Kanten besitzen. Wenn man jetzt einen dieser Graphen konstruieren möchte, so hat man für jede Kante genau 2 Möglichkeiten, sie entweder zu dem Graphen hinzuzufügen oder sie wegzulassen.

Man kann folglich maximal $2^{\frac{d^2}{2}(1-\frac{1}{k})}$ verschieden Graphen erzeugen, für die *f* eine gültige Färbung ist. **q.e.d.**

Lemma 4.3:
Die Gesamtanzahl der gültigen Färbungen mit *k* Farben *aller* Graphen mit *d* Knoten beträgt höchstens $k^d\ 2^{\frac{d^2}{2}(1-\frac{1}{k})}$.
Diese Aussage formuliert sich mithilfe des chromatischen Polynoms *P* auch wie folgt:
Sei $\mathcal{G}_d$ die Menge aller ungerichteten Graphen mit *d* Knoten,
dann gilt:

$$\sum_{H \in \mathcal{G}_d} P_H(k) \leq k^d\ 2^{\frac{d^2}{2}(1-\frac{1}{k})}$$

Beweis:
In diesem Lemma sollen alle gültigen Färbungen *f* für alle Graphen *H* mit *d* Knoten gezählt werden. Also alle Kombinationen bzw. Tupel (*H*, *f*), für die gilt, dass *f* eine gültige Färbung des Graphen *H* ist.
Wenn wir jetzt einmal *f* festhalten, also alle Tupel zählen, die ein und dieselbe Färbung *f* besitzen, so zählen wir damit alle Graphen *H*, für die *f* eine gültige Färbung ist.

Nach Lemma 4.2 wissen wir, dass es höchstens $2^{\frac{d^2}{2}(1-\frac{1}{k})}$ solcher Graphen geben kann.
Damit haben wir für eine feste Färbung *f* eine Abschätzung für die Anzahl der Tupel gefunden.
Eine Abschätzung für alle Färbungen *f* erhält man folglich, indem man die gefundene Abschätzung für ein festes *f* mit der Anzahl der möglichen Färbungen multipliziert.
Da es k^d Möglichkeiten gibt, *d* Knoten mit *k* Farben zu färben, ist der Beweis erbracht.
q.e.d.

Wenn wir den Suchbaum aus Abb. 3.6 nochmal etwas genauer betrachten, so erkennen wir, dass die Knoten, die auf einer Ebene im Suchbaum liegen oder anders formuliert, die dieselbe Tiefe *d* haben, gültige Färbungen des durch die ersten *d* Knoten induzierten Teilgraphen sind.

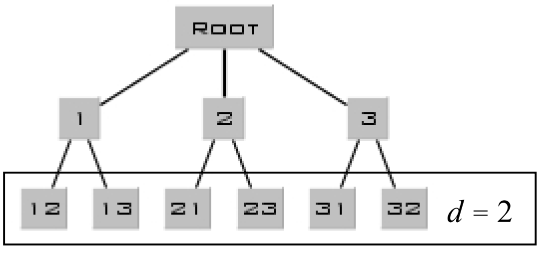

Abb. 4.1: Ausschnitt des Suchbaumes aus Abb. 3.6

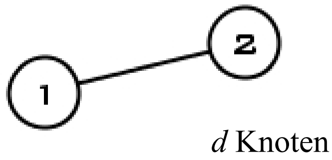

Abb. 4.2: Durch die ersten 2 Knoten induzierter Teilgraph des Graphen aus Abb. 3.1

Lemma 4.4:
Sei *P* das chromatische Polynom.
Die Anzahl der Knoten in Tiefe *d* im Backtracking-Suchbaum für das Problem der *k*-Färbbarkeit eines Graphen *G* stimmt mit der Anzahl der gültigen Färbungen des induzierten Teilgraphen $H_d(G)$ mit *k* Farben überein und beträgt somit also $P_{H_d(G)}(k)$.

Lemma 4.5:
Seien $n, d \in \mathbb{N}$ mit $d \leq n$ bel.

Es gibt $2^{\binom{n}{2}-\binom{d}{2}}$ ungerichtete Graphen *G* mit *n* Knoten, für die der induzierte Teilgraph $H_d(G)$ identisch ist.

Beweis:
Sei *H* ein beliebiger ungerichteter Graph mit *d* Knoten.

Der Graph *H* hat maximal $\binom{d}{2}$ Kanten, denn $\binom{d}{2}$ ist die Anzahl der Möglichkeiten, aus einer *d*-elementigen Menge 2-elementige Teilmengen zu bilden. Die *d*-elementige Menge ist hier natürlich die Knotenmenge des Graphen. Eine 2-elementige Teilmenge entspricht somit einer ungerichteten Kante. Folglich ist $\binom{d}{2}$ die Anzahl der möglichen Kanten in *H*.

Die Anzahl der Graphen *G*, die *H* als induzierten Teilgraphen besitzen ($H_d(G) = H$), erhält man, indem man die Möglichkeiten zählt, den Graphen *H* auf einen Graphen *G* mit *n* Knoten zu erweitern.

Von den $\binom{n}{2}$ Kanten, die man in diesen erweiterten Graphen *G* einzeichnen könnte, stehen die Kanten zwischen den ersten *d* Knoten nicht mehr zur Auswahl, da sie durch den Graphen *H* bereits festgelegt sind. Es bleiben also nur noch $\binom{n}{2}-\binom{d}{2}$ Kanten übrig, die man zu *G* hinzufügen oder eben weglassen kann.

Folglich lassen sich $2^{\binom{n}{2}-\binom{d}{2}}$ verschiedene Graphen *G* erzeugen, die *H* als induzierten Teilgraphen haben, für die somit also der induzierte Teilgraph $H_d(G)$ identisch ist.
q.e.d.

Wir sind jetzt bestens ausgerüstet, um endlich unser ursprüngliches Vorhaben in Angriff nehmen zu können:

$$\begin{aligned}
A(n,k) &= \frac{1}{|\mathcal{G}_n|} \sum_{G \in \mathcal{G}_n} \left(\text{Anzahl der Knoten im Suchbaum für } G\right) \\
&= 2^{-\binom{n}{2}} \sum_{G \in \mathcal{G}_n} \left(\sum_{d=0}^{n} \text{Anzahl der Knoten in Tiefe } d \text{ für } G\right) \\
&= 2^{-\binom{n}{2}} \sum_{G \in \mathcal{G}_n} \sum_{d=0}^{n} P_{H_d(G)}(k) \qquad \text{(nach Lemma 4.4)} \\
&= 2^{-\binom{n}{2}} \sum_{d=0}^{n} \Big(\sum_{G \in \mathcal{G}_n} P_{H_d(G)}(k) \Big)
\end{aligned}$$

Betrachten wir einmal für ein festes *d* die innere Summe. Diese läuft über alle Graphen *G* mit *n* Knoten hinweg, während für jeden Graphen *G* nur der induzierte Teilgraph $H_d(G)$ in die Berechnung der Summe einfließt. Für die Graphen *G*, die denselben induzierten Teilgraphen besitzen, ist der Summand $P_{H_d(G)}(k)$ folglich identisch. Nach Lemma 4.5 ist dies für jeweils $2^{\binom{n}{2}-\binom{d}{2}}$ Graphen der Fall. Lässt man jetzt stattdessen die Summe über alle induzierten Graphen, also alle Graphen *H* mit *d* Knoten laufen, so gilt:

$$A(n,k) = 2^{-\binom{n}{2}} \sum_{d=0}^{n} 2^{\binom{n}{2}-\binom{d}{2}} \Big(\sum_{H \in \mathcal{G}_d} P_H(k) \Big)$$

$$= \sum_{d=0}^{n} 2^{-\binom{n}{2}} \, 2^{\binom{n}{2}-\binom{d}{2}} \Big(\sum_{H \in \mathcal{G}_d} P_H(k) \Big)$$

$$= \sum_{d=0}^{n} 2^{-\binom{d}{2}} \Big(\sum_{H \in \mathcal{G}_d} P_H(k) \Big)$$

$$\leq \sum_{d=0}^{n} 2^{-\binom{d}{2}} \Big(k^d \, 2^{\frac{d^2}{2}(1-\frac{1}{k})} \Big) \qquad \text{(nach Lemma 4.3)}$$

$$= \sum_{d=0}^{n} k^d \, 2^{-\frac{d(d-1)}{2}} \, 2^{\frac{d^2}{2}(1-\frac{1}{k})}$$

$$= \sum_{d=0}^{n} k^d \, 2^{-\frac{d^2}{2}+\frac{d}{2}+\frac{d^2}{2}-\frac{d^2}{2k}}$$

$$= \sum_{d=0}^{n} k^d \, 2^{-\frac{d^2}{2k}+\frac{d}{2}}$$

$$\leq \sum_{d=0}^{\infty} k^d \, 2^{-\frac{d^2}{2k}+\frac{d}{2}} =: \mathrm{const}(k)$$

Diese Reihe konvergiert für alle $k \in \mathbb{N}$. Das bedeutet, dass der Grenzwert dieser Reihe für jedes *k* existiert. Der Grenzwert selbst ist dabei natürlich für jedes *k* verschieden und daher also eine von *k* abhängige Konstante.

Bezeichne const(*k*) den von *k* abhängigen Grenzwert der Reihe, so gilt wie bereits gesehen:

$$A(n,k) \leq \mathrm{const}(k)$$

Wir haben also gezeigt, dass *A*(*n,k*) durch eine Konstante beschränkt ist, die nur von *k* abhängig ist und somit für alle $n \in \mathbb{N}$ gilt.
q.e.d.

Für ein festes *k* ist daher die Laufzeit im average case konstant, lässt man dagegen aber ebenfalls das *k* laufen, so entwickelt sich unsere Abschätzung exponentiell.

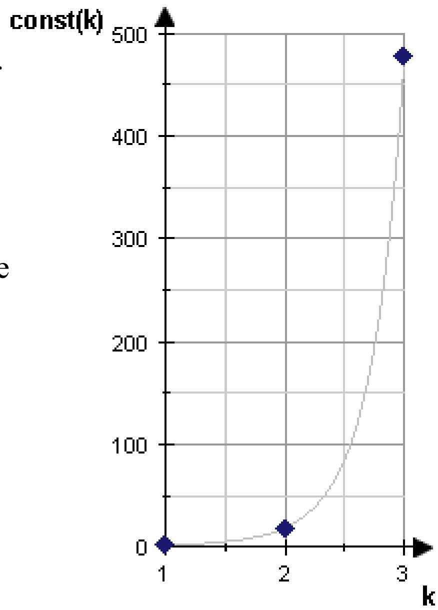

Abb. 4.3: Entwicklung von const(k)

5 Literaturverzeichnis

[Bir12] G.D. Birkhoff: A determinant formula for the number of ways of coloring a map Annals of Math. 14 (1912), 42-46.

[Sch04] C. Schindelhauer: Vorlesungsskript Average-Komplexitätstheorie und Average-Analyse von Algorithmen, Universität Paderborn (2004). http://wwwcs.upb.de/cs/ag-madh/vorl/Average03/skript/skript-average.pdf

[Wil87] H.S. Wilf: Algorithms and Complexity, Prentice Hall (1987). http://www.cis.upenn.edu/%7Ewilf/AlgComp3.html