

Algorithmische Graphentheorie

9. Übungsblatt

Jasper Gude

Pia Röttgers

22. Juni 2026

8 / 34

Aufgabe 1 – Dreifärbbarkeit

a) Siehe Abbildung 1.

1 / 1

b) Da die Knoten, die zwei Türme miteinander verbinden, die selbe Farbe haben, kann man zwischen zwei Türme einen weiteren Turm einfügen, der die Färbbarkeitsregeln nicht verletzt.

Die Knoten mit Grad 2 im Sterngraphen sind die Spitzen der Türme. Da die Turmverbindungsknoten alle die selbe Farbe haben müssen und die zu ihnen adjazenten Knoten jeweils unterschiedlich gefärbt sein müssen, da sie selbst adjazent zueinander sind, müssen die Spitzen die letzte freie Farbe bekommen.

Könnte klarer argumentiert sein, aber mit "Konstruktionsanleitung" aus dem Absatz obendrüber kann mans so stehen lassen.

2 / 2

c) Um das Problem 3COL auf das Problem 3COL4 zu reduzieren, müssen wir dafür sorgen, dass Knoten mit Grad größer 4 so aufgelöst werden, dass sie maximal Grad 4 haben.

Dazu machen wir uns zu nutze, dass der Sterngraph beliebig erweiterbar ist und somit beliebig viele Knoten mit Grad 2 hat, die alle die selbe Farbe haben müssen. ... und im Sterngraphen der Maximalgrad 4 ist.

Einen Knoten v mit Grad $m > 4$ ersetzen wir durch einen Stern mit m Zacken. Die zu v adjazenten Kanten verbinden wir mit den Spitzen des Sterns.

Jetzt haben wir einen Graphen mit Maximalgrad 4 auf dem wir TEST3COL4 anwenden können.

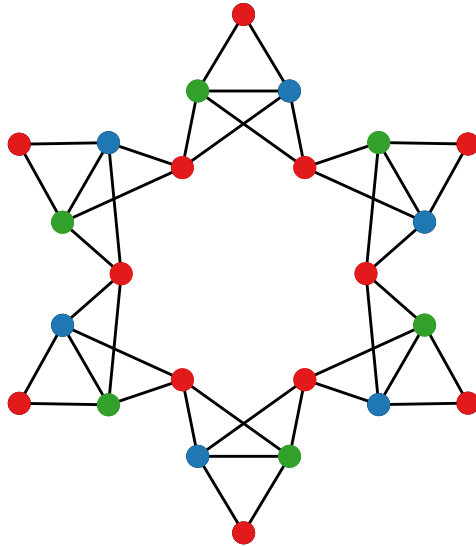


Abbildung 1: Dreifärbung des Sterngraphs.

Laufzeitbegründung für
Polynomialzeit-Reduktion fehlt.

Beim Rückübersetzen können wir die Sterne wieder durch einen einzigen Knoten ersetzen, der die Farbe der Spitzen hat. Da diese die selbe Farbe haben, verletzt das nicht die Färbbarkeitsregel.

Da 3COL NP-vollständig ist, muss also 3COL4 und insbesondere auch TEST3COL4 auch NP-vollständig sein.

3 / 4

Aufgabe 2 – Chromatische Zahl

- a) Angenommen der Graph G ist vollständig. So hat jeder Knoten den maximalen Knotengrad $\Delta(G)$. Für alle Knoten v gilt, dass jeder adjazente Knoten und v eine eigene Farbe haben muss. Das sind also $\Delta(G) + 1$ viele. Jeder Graph (mit weniger Kanten) hat also eine chromatische Zahl $\chi(G) \leq \Delta(G) + 1$.
- b) Für alle Anzahlen an Knoten $n \in \mathbb{N}$ und Maximalgrad $\Delta < n$ existiert ein Graph $G_{n,\Delta}$ mit $\chi(G_{n,\Delta}) = \Delta + 1$. Dieser Graph hat eine Clique der Größe Δ .

Nicht-vollständige Graphen haben ggf. einen anderen Maximalgrad. Diese Begründung hier argumentiert ausschließlich, dass die chromatische Zahl von Graphen G mit $n = |V(G)|$ kleiner oder gleich n ist.

0 / 2

0 / 2

Okay, ich betrachte einen Graphen mit 3 Knoten und 2 Kanten.

Dieser Graph hat Maximalgrad 2 und enthält Cliques der Größe 2, entspricht also eurer Beschreibung. Seine chromatische Zahl sollte also laut eurer Beschreibung 3 sein, er hat aber eine 2-Färbung.
-> Widerlegt durch Gegenbeispiel.

(Generell ist die Aussage korrekt, wenn ihr die geforderte Größe der Clique um 1 erhöht, aber auch damit wäre die Beschreibung für die Aufgabenstellung zu knapp. Es sollte zu jedem Delta und jedem n ein Graph angegeben werden, dafür reicht eine Beschreibung der notwendigen Eigenschaft nicht aus)
Sowwy :(

Aufgabe c) bekommt einen neuen Graphen, bezieht sich nicht mehr auf vorherige Aussagen. (Delta zu verwenden war hier nicht vorgesehen, aber ich mein.... falsch ist es nicht)
Zumindest klarer beschreiben, dass für jeden Knoten eine zugeordnete Variable für die Farbe angelegt wird (sonst fallen x,a,b einfach vom Himmel), und dass sich die Nebenbedingung auf Kanten in $E(G)$ bezieht. Ohne diese Einschränkung kommt immer eine n-Färbung raus!

- c) Die Summe über alle Farben zu bestimmen ist eine andere Aussage als die höchste verwendete Farbe zu finden.

Variablen:

$$x_1, \dots, x_n \in \{1, \dots, \Delta + 1\}$$

Jeder Knoten hat eine Farbe

Zielfunktion:

$$\arg \min \sum_{i=1}^n x_i$$

Es werden so wenig Farben wie möglich benutzt

Nebenbedingung:

$$\forall ab: x_a \neq x_b$$

Die Knoten einer Kante haben nicht die selbe Farbe

0.5 / 2

- d) Wir führen eine binäre Hilfsvariable ein:

$$y = \begin{cases} 1 & \text{falls } x_1 \leq x_2 - 1 \text{ oder } x_1 - 1 \geq x_2 \\ 0 & \text{sonst} \end{cases}$$

Ein so definiertes y können wir im ILP nicht bauen. (Und was tut M hier?)
Es müssen Constraints formuliert werden, die der binären Hilfsvariable einen Wert zuordnen, sodass sie das gewollte Verhalten zeigt.

Die Konstante M stellt sicher, dass wir nicht in die Situation $\infty - 1$ kommen, was nicht definiert ist.

Nun können wir $x_1 \neq x_2$ durch $y = 1$ ersetzen.

0 / 3

Aufgabe 3 – Perfektes Eliminationsschema

1. Falls $|V(G)| = 0$, gib das Eliminationsschema zurück.
2. Suche nach nicht erweiterbaren U , sodass $G[U]$ zusammenhängend und $U \cup N(U) \neq V(G)$. Das geht mit Breitensuche mit einem beliebigen Startknoten s .
3. Wähle aus $W = V(G) \setminus U \cup N(U)$ einen Knoten aus und füge ihn zum Eliminationsschema hinzu. Lösche diesen Knoten dann aus G .
4. Gehe zurück zu Punkt 1.

✓ Das die Menge U nicht maximal sondern nur nicht erweiterbar ist, macht die Folgerung, dass jeder Knoten in W mit jedem in $N(U)$ verbunden ist, nicht kaputt, da, wenn es eine Kante zwischen einem Knoten in $N(U)$ und W nicht existiert, U erweitert werden kann.

Ebenfalls bleibt die Eigenschaft, dass $N(U)$ eine Clique ist erhalten. Die Argumentation ist die selbe, wie zuvor.

Jeder Knoten in W ist ein simplizialer Knoten in $G[W]$, da, wenn es eine Kante zwischen zwei Knoten in W nicht gibt, dann U erweitert werden kann.

2. Das geht, aber es muss auch da stehen wie es umgesetzt wird.
3. Wir haben nur gezeigt, dass es in W einen simplizialen Knoten gibt, der auch auf G simplizial ist. Wir haben nicht gezeigt, dass das für alle Knoten in W gilt.

Durch Magie ✓ wird nicht der gesamte Breitensuchbaum erstellt und die Breitensuche läuft in $o(|V| + |E|) = o(n + n^2)$. Durch die n rekursiven Aufrufe, in denen jeder Knoten einmal zum Eliminationsschema hinzugefügt wird, läuft der ganze Algorithmus dadurch in $o(n^3)$

1.5 / 4

Was ist hier rekursiv?

Wenn man die Rekursion und BFS geschickt zusammenbaut, ist jeder Knoten max. 1x Startpunkt einer BFS und Kanten werden höchstens 2x betrachtet. $O(V)$ viele Breitensuchen über $O(V^2)$ Kanten kriegen wir in $O(V^3)$ hin.