Lösungsvorschlag Arbeitsheft 2

1 Kern-Mengen in gerichteten Graphen

Eine Kernmenge $K \subseteq V$ eines gerichteten Graphen $G = (V, E)$ erfüllt folgende Eigenschaften:

$$K^2 \cap E = \varnothing \qquad \text{bzw.} \qquad \forall k \in K : Succ(k) \cap K = \varnothing \tag{1}$$

$$\forall v \in V \setminus K : Succ(v) \cap K \neq \varnothing \tag{2}$$

Dabei ist $Succ(v) := \{w \in V : (v, w) \in E\}$ die Menge Nachfolger eines Knotens.

a)

Sei also $n \in \mathbb{N}$, $V = [1, n], E = \{(i, j) \in V^2 \mid i \neq j\}$ und $G = (V, E)$ gerichtet.
Setze $K = \{1\}$. Es gilt:

$$(1, 1) \notin E \implies (1)$$

$$(V \setminus K) \times \{1\} \subset E \implies (2)$$

Folglich erfüllt K beide Eigenschaften einer Kernmenge.

b)

Im folgenden Schreiben wir $[a]_n$ anstatt $k \pmod n$ für Lesbarkeit.

Sei $C_n := ([1, n], \{(i, [i + 1]_n) \mid i \in [1, n]\})$ der gerichtete Kreis mit n Knoten.

Sei nun $n = 2\ell \geq 3$ für $\ell \in \mathbb{N}$. Setze $K := 2[1, \ell] = \{2, 4, 6, \cdots, 2\ell\} \subset V_{C_n}$. Dann gilt:

$$\forall k \in K : Succ(k) = \{(k, [k + 1]_n)\} \not\subset K^2 \implies (1)$$

$$\forall v \in V_{C_n} \setminus K : Succ(v) = \{(v, v + 1)\} \in V_{C_n} \times K \implies (2)$$

Damit ist K eine Kernmenge von C_n.

Sei nun $n = 2\ell + 1 \geq 3$ für $\ell \in \mathbb{N}$. Angenommen $K \subseteq V_{C_n}$ wäre eine Kernmenge.

Offensichtlich muss dann $K \neq \varnothing$. Sei nun $v \in V_{C_n}$. Dann:

$$v \in K \implies (1) \wedge Succ(v) = \{[v + 1]_n\} \implies [v + 1]_n \notin K$$

$$v \notin K \implies (2) \wedge Succ(v) = \{[v + 1]_n\} \implies [v + 1]_n \in K$$

Sei nun $k \in K$. Wenn man obige Resultate endlich oft iteriert anwendet, so erhält man

$$k \in K \implies [k + 1]_n \notin K \implies [k + 2]_n \in K \implies \cdots \implies [k + 2\ell]_n \in K$$

Jedoch ist $[k + 2\ell]_n = [k - 1]_n$. Ferner gilt $Succ([k - 1]_n) = \{k\}$, damit ist aber (1) verletzt, und K kann keine Kernmenge sein.

Insgesamt haben genau die gerichteten Kreise mit einer geraden Anzahl an Knoten, $C_{2\ell}, \ell \in \mathbb{N}$ eine Kernmenge.

c)

Wir beweisen dies durch Angabe eines Algorithmus und dessen Korrektheitsbeweis.

Der Algorithmus wird die Knoten des Baumes in V färben mit $c : V \to \{0, 1, 2\}$, wobei 0 für nicht gefärbt steht. $c(v) = 1$ soll dabei $v \in K$ bedeuten, und $c(v) = 2$ dann $v \notin K$, wobei K die gesuchte Kernmenge ist.

1. Solange $c^{-1}(0) \neq \varnothing$, es also ungefärbte Knoten gibt:

   (a) Setze $c(k) := 1$ für alle $k \in \{v \in c^{-1}(0) \mid Succ(v) = \varnothing\}$.
   Knoten ohne Nachfolger müssen in K liegen.

   (b) Setze $c(v) := 2$ für alle $v \in c^{-1}(0)$ mit $Succ(v) \cap c^{-1}(1) \neq \varnothing$.
   Knoten mit Nachfolgern in K können wegen (1) nicht in K sein.

   (c) Setze $c(k) := 1$ für $k \in \{v \in c^{-1}(0) \mid \forall w \in Succ(v) : c(w) = 2\}$.
   Knoten die nur Nachfolger haben, welche (schon festgelegt) nicht in K liegen, dürfen in K liegen.

Wir zeigen dass zu jeder Zeit $c^{-1}(1)$ eine Kernmenge von $c^{-1}(\{1, 2\})$ bildet. Ferner sagen wir, dass $\varnothing$ eine korrekte Kernmenge von $\varnothing$ ist.

Sei nun $c^{-1}(1)$ eine Korrekte Kernmenge von $c^{-1}(\{1, 2\})$.
Durch anwenden von (a) werden beide Eigenschaften (1) und (2) einer Kernmenge erhalten, denn:

- Für (1): Sei $k \in V$ einer der gerade gefärbten Knoten, also $c(k) = 1$ und $Succ(k) = \varnothing$. Wegen letzterem, gilt schonmal $Succ(k) \cap K = \varnothing$, also kann k nicht (1) verletzen.

  Betrachte nun $v \in \{w \in c^{-1}(\{1, 2\}) \mid k \in Succ(w)\}$ insofern nichtleer. Wäre $c(v) = 1$, so müsste v durch die Schritte (a) oder (c) des Algorithmus gefärbt worden sein. Wegen $Succ(v) \neq \varnothing$ kann es nicht durch (a) gewesen sein. Da k bis vor kurzem ungefärbt und nun mit 1, also insbesondere nie $c(k) = 2$ gegolten hat, kann dies auch nicht durch Schritt (c) passiert sein. Folglich muss $c(v) = 2$, und damit ist Bedingung (1) nicht verletzt.

- Für (2) spielt das 1-färben (hinzufügen von Knoten zur Kernmenge) keine Rolle.

Folglich ist nach anwenden von (a) $c^{-1}(1)$ immernoch eine korrekte Kernmenge von $c^{-1}(\{1, 2\})$.

Wir untersuchen nun Schritt (b). Sie wieder $c^{-1}(1)$ eine korrekte Kernmenge von $c^{-1}(\{1,2\})$. Es werden durch anwenden von (b) wieder die Bedingungen erhalten:

- Für (1) spielt das 2-färben (hinzufügen zu $V \setminus K$) keine Rolle.
- Für (2): Da jeder der gerade 2-gefärbten Knoten per Vorraussetzung von (b) einen Nachfolger in $c^{-1}(1)$ hat, bleibt die Bedingung (2) für $c^{-1}(\{1,2\})$ erhalten.

Folglich ist nach anwenden von (b) $c^{-1}(1)$ immernoch eine korrekte Kernmenge von $c^{-1}(\{1,2\})$.

Wir untersuchen nun Schritt (c). Sie wieder $c^{-1}(1)$ eine korrekte Kernmenge von $c^{-1}(\{1,2\})$. Es werden durch anwenden von (c) wieder die Bedingungen erhalten:

- Für (1): Sei $k \in V$ einer der gerade gefärbten Knoten, also $c(k) = 1$ und $\forall w \in Succ(k) : c(w) = 2$. Wegen letzterem kann k nicht selbst die Bedingung (1) verletzen.

  Betrachte nun $v \in \{w \in c^{-1}(\{1,2\}) \mid k \in Succ(w)\}$ insofern nichtleer. Wäre $c(v) = 1$, so müsste v durch die Schritte (a) oder (c) des Algorithmus gefärbt worden sein. Wegen $Succ(v) \neq \varnothing$ kann es nicht durch (a) gewesen sein. Da k bis vor kurzem ungefärbt und nun mit 1, also insbesondere nie $c(k) = 2$ gegolten hat, kann dies auch nicht durch Schritt (c) passiert sein. Folglich muss $c(v) = 2$, und damit ist Bedingung (1) nicht verletzt.
- Für (2) spielt das 1-färben (hinzufügen von Knoten zur Kernmenge) keine Rolle.

Folglich ist nach anwenden von (c) $c^{-1}(1)$ immernoch eine korrekte Kernmenge von $c^{-1}(\{1,2\})$.

Wir zeigen nun dass in jeder Iteration mindestens ein Knoten gefärbt werden kann:

Es ist klar, dass jeder (endliche) orientierte Baum mindestens einen Knoten besitzt, welcher keine Nachfolger hat. Andererseits hätte man einen Pfad, in welchem jeder Knoten einen Nachfolger hat. Da Bäume per Definition aber azyklisch sind, hätte man damit per trivialer Induktion einen unendlich langen Pfad in dem orientiertem Baum, was grundsätzlich nicht möglich ist.

Folglich können wir zu beginn mindestens einen Knoten 1-färben (zur Kernmenge hinzufügen).

Sei $c^{-1}(0) \neq \varnothing$, da wir sonst fertig sind, und $Succ(v) \neq \varnothing$ für $v \in c^{-1}(0)$, da Knoten ohne Nachfolger beim ersten (a) gefärbt werden.

Dann gibt es ein $v \in c^{-1}(0)$ mit $Succ(v) \subseteq c^{-1}(\{1,2\})$, also einen ungefärbten Knoten, welcher nur gefärbte Nachfolger hat, da sonst jeder Knoten einen ungefärbten Nachfolger hat und wir damit einen Zykel oder einen unendlichen Pfad in $c^{-1}(0)$ hätten.

Falls nun $Succ(v) \subseteq c^{-1}(2)$, so lässt sich (c) anwenden, andernfalls (b).

Insgesamt lässt sich mindestens ein Knoten in jeder Iteration färben. Da wir von endlichen Eingabebäumen ausgegangen sind, wird der Algorithmus also nach endlich vielen Schritten terminieren. Dann ist aber auch $V = c^{-1}(\{1,2\})$ und wir haben die Kernmenge $c^{-1}(1)$ von V.

d)

Das Zertifikat ist einfach eine Kodierung der Kernmenge selbst. Dies ist offensichtlich kürzer als die Eingabe da $K \subseteq V$. Ferner lässt sich dies in polynomieller Zeit verifizieren, indem man für alle Knoten alle Nachfolger betrachtet und je nach Knoten die Bedingung (1) oder (2) überprüft. Dies geht in quadratischer Zeit.

e) Nein, dies würde Bedingung (1) widersprechen.

f) Nein, dies würde Bedingung (2) verletzen, da $A(x_i)$ und $A(\overline{x_i})$ nur einen ausgehenden Pfeil zum jeweils anderen haben. Wenn beide nicht in der Kernmenge wären, hätten sie keinen Pfeil zu einem Knoten in der Kernmenge.

g) k.

h) Bei m Klauseln werden $3m$ der $B_j(c_i)$-Knoten eingeführt. Für n Variablen werden $2n$ Knoten eingeführt (1 pro Literal). Folglich ist $|V_{G(\Phi)}| = 3m + 2n \leq 3(m+n)$ also polynomiell in $m+n$.

i) Wir haben $2n$ ausgehende Pfeile der Literal-Knoten (1 pro Literal). Jeder der $B_j(c_i)$-Knoten hat pro Literal in Klausel c_i genau 1 ausgehenden Pfeil zum entsprechendem Literal, also 3 Pfeile pro $B_j(c_i)$-Knoten, von denen es $3m$ gibt, was $9m$ ausgehende Pfeile der $B_j(c_i)$-Knoten macht. Insgesamt haben wir $9m + 2n$ Pfeile, was polynomiell in $m+n$ beschränkt ist.

j) Die Anzahl der Kanten / Pfeile in einem Graphen $G = (V, E)$ ist beschränkt durch $|V|^2$. Wenn also $|V|$ schon polynomiell beschränkt in der Eingabe ist, so ist dies auch $|V|^2$.

k) Da jeder Literal-Knoten genau einen ausgehenden Pfeil hat, folgt mit e) und f) schon, dass die Bedingung (1) einer Kernmenge erfüllt ist, da diese nur durch die Knoten der Kernmenge (also hier die Literalknoten) verletzt werden kann.

Der jeweils andere Literalknoten, welcher nicht in der Kernmenge liegt, verletzt die (2) Bedingung nicht, da sein einziger ausgehender Pfeil eben auf sein entsprechendes gegenüber in der Kernmenge zeigt.

Alle Klauselknoten haben Pfeile zu den Literalen, welche die Klausel wahrmachen (und damit in der Kernmenge liegen). Insgesamt ist also Bedingung (2) auch im Graphen erfüllt. Daher bildet die gegebene Menge der Literalknoten eine Kernmenge für $G(\Phi)$.

l) Eine Wahrheitsbelegung weißt jeder Variable x einen Wert zu. Dann hat genau eines der Literale x und $\overline{x}$ den Wahrheitswert 1, sodass per Definition der Kernmenge genau einer der Literalknoten $A(x), A(\overline{x})$ in dieser liegt.

m) Die 3 Klauselknoten $B_1(c), B_2(c), B_3(c)$ einer Klausel c bilden einen C_3. Wären mindestens 2 dieser in der Kernmenge, also B_i, B_j mit $i \neq j$, so hätten wir wegen $B_1 \to B_2 \to B_3 \to B_1$ eine Verletzung der Bedingung (1) einer Kernmenge. Folglich darf höchstens einer der 3 Klauselknoten in K liegen.

n) Sei also $c = (\ell_1 \vee \ell_2 \vee \ell_3)$ eine der Klausel. Angenommen $A(\ell_i) \notin K$ für $i = 1, 2, 3$. Nach m) existiert ein $j \in \{1, 2, 3\}$ sodass $B_j(c) \notin K$ keinen Pfeil auf einen Klauselknoten in K hat. Da $B_j(c)$ aber sonst nur ausgehende Pfeile zu $A(\ell_i)$ für $i = 1, 2, 3$ hat, wäre dann Bedingung (2) der Kernmenge verletzt. Folglich $\exists j \in \{1, 2, 3\} : A(\ell_j) \in K$.

o) Die Wahrheitsbelegung $\varphi : X \to \{0, 1\}$ ist $\varphi(x) := \begin{cases} 1 & , A(x) \in K \quad (\overset{\text{e}}{\iff} A(\overline{x}) \notin K) \\ 0 & , \text{sonst} \end{cases}$

2 Drei-Färbbarkeit von Graphen

Im folgenden betrachten wir ungerichtete Graphen $G = (V, E)$. Sei für $v \in V$ dann

$$Adj(v) := \{w \in V \mid (v, w) \in E \quad \vee \quad (w, v) \in E\}$$

a)

Wir betrachten C_n für $n \geq 3$. Wir schreiben wieder $[k]_n$ für k (mod n).
Entsprechend ist $V_{C_n} = [0, n-1]$.

Fall 1: $n = 2\ell$ für $\ell \in \mathbb{N}$:
Hier lässt sich C_n sogar 2-färben, indem man $c(1+2k) := 1$ für $k \in [0, \ell-1]$ setzt, also alle ungeraden Knoten in V_{C_n} mit 1 färbt. Alle anderen färbt man mit 2, also $c(2k) := 2$ für $k \in [0, \ell-1]$. Dann folgt für $k \in V_{C_n}$ sofort, dass $c([k-1]_n) \neq c(k) \neq c([k+1]_n)$, und da $Adj(k) = \{[k-1]_n, [k+1]_n\}$, damit c eine 2-Färbung von C_n ist.

Fall 2: $n = 2\ell + 1$ für $\ell \in \mathbb{N}$:
Wir setzen $c(0) := 3$. Dann analog zum oberen Fall färben wir die restlichen ungeraden Knoten mit 1, die geraden mit 2, also

$$c(1+2k) := 1 \quad \text{für } k \in [0, \ell-1] \qquad \text{sowie} \qquad c(2k) := 2 \quad \text{für } k \in [1, \ell-1]$$

Dann ist für $c(n) \neq c(0) \neq c(1)$ und $Adj(0) = \{n, 1\}$. Ferner gilt für $k \in [1, n]$, dass $c([k-1]_n) \neq c(k) \neq c([k+1]_n)$ und $Adj(k) = \{[k-1]_n, [k+1]_n\}$, also Insgesamt c eine 3-Färbung von C_n ist.

Folglich lässt sich C_n in jedem Fall 3-Färben.

b)

Wir betrachten einen endlichen Baum $T = (V, E)$ mit maximaler Tiefe $d \in \mathbb{N}$. Sei $D(v) \in [0, d]$ die Ebene bzw Tiefe eines Knotens $v \in V$. T lässt sich 2-färben, indem man

$$\forall v \in \{w \in V \mid D(v) \text{ gerade}\} : c(v) := 1 \quad \text{und} \quad \forall v \in \{w \in V \mid D(v) \text{ ungerade}\} : c(v) := 2$$

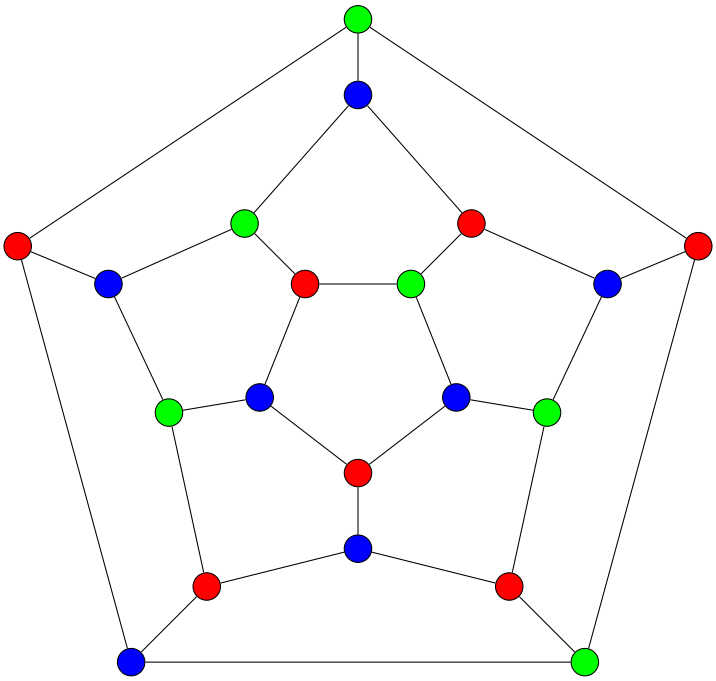
setzt. Dank der Struktur des Baumes gilt

$$\forall v \in V : \forall w \in Adj(v) : [D(w)]_2 \neq [D(v)]_2$$

also dass alle zu v adjazenten Knoten auf ebenen mit anderer Parität als v liegen.

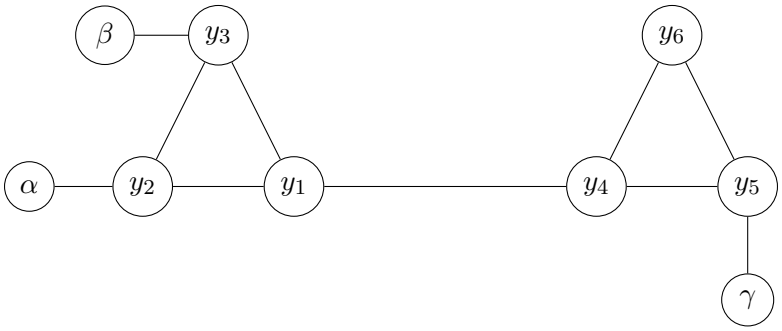
Seien nun $u, v \in V$. Wir fügen die Kante (u, v) zu E hinzu. Falls $c(u) \neq c(v)$, so muss nichts geändert werden. Falls jedoch $c(u) = c(v)$ so setzen wir $c(u) := 3$, und benutzen unsere dritte Farbe. Dann ist c eine korrekte 3-Färbung des veränderten T.

c)



d) Das Zertfikat ist einfach die 3-partition der Knoten hintereinandergeschrieben, welches offensichtlich als permutation der Knoteneingabe der Länge her polynomiell in der Eingabelänge beschränkt ist. Zuerst überprüft man das Zertifikat auf Vollständigkeit, dass auch wirklich jeder Knoten genau 1 mal vorkommt. Dann betrachtet man der Reihe nach die Knoten einer Farbe und überprüft für jeden, dass sie nicht adjazent zu einem der Knoten mit der selben Farbe sind. Dies dauert maximal $\mathcal{O}(|V|^2)$ Operationen.

e)



f)

Sei also $c(\alpha) = c(\beta) = c(\gamma) = 1$. Dann muss $c(y_1) = 1$, da $\{c(y_2), c(y_3)\} = \{2, 3\}$. Folglich ist $c(y_4) \in \{2, 3\}$, wodurch mit $c(\gamma) = 1$ dann auch $\{c(y_4), c(y_5)\} = \{2, 3\}$ folgt. Also bleibt nur die Wahl $c(y_6) = 1$.

g) Betrachte einfach alle Fälle:

α	β	γ	y_1	y_2	y_3	y_4	y_5	y_6
1	1	1	1	2	3	2	3	1
1	1	2	1	2	3	2	3	1
1	1	3	1	2	3	3	2	1
1	2	1	3	2	1	2	3	1
1	2	2	3	2	1	2	3	1
1	2	3	3	2	1	3	2	1
1	3	1	2	3	1	2	3	1
1	3	2	2	3	1	2	3	1
1	3	3	2	3	1	3	2	1
2	1	1	3	1	2	2	3	1

α	β	γ	y_1	y_2	y_3	y_4	y_5	y_6
2	1	2	3	1	2	2	3	1
2	1	3	2	1	3	3	2	1
2	2	1	2	1	3	3	2	1
2	3	1	1	3	2	2	3	1
3	1	1	2	1	3	3	2	1
3	1	2	2	1	3	2	3	1
3	1	3	2	1	3	3	2	1
3	2	1	1	2	3	2	3	1
3	3	1	3	1	2	2	3	1

h) Offensichtlich und auch schon öfters benutzt.

i) Folgt aus h), da $A(x_i), A(\overline{x_i}), DUMMY$ ein Dreieck bilden.

j) k.

k) Es gibt $2n$ Variablenknoten sowie m Gadgets mit jeweils 6 Knoten (α, β, γ werden verschmolzen). Dazu haben wir die 3 Knoten WAHR,FALSCH,DUMMY. Insgesamt gibt es also $2n + 6m + 3 \in \mathcal{O}(m + n)$ Knoten.

l) Es gibt pro Variable 3 Kanten; zwischen den entsprechenden Literalknoten und von beiden zu DUMMY. Ein Gadget hat 10 Kanten an sich. Ferner kommen pro Gadget 2 Kanten hinzu; von y_6 zu FALSCH und DUMMY. Damit haben wir insgesamt $3n + 12m \in \mathcal{O}(m + n)$ Kanten.

m) Sei also $c = (\ell_1 \vee \ell_2 \vee \ell_3)$ eine Klausel aus Φ und φ die erfüllende Variablenbelegung.
Dann ex. $i \in \{1, 2, 3\} : \varphi(\ell_i) = 1$. Da dann $A(\ell_i)$ mit einem der α, β, γ Knoten aus $G_9(c)$ verschmolzen ist, folgt mit g), dass es eine 3-Färbung gibt in der y_6 die Farbe WAHR hat. Da verschiedene Gadgets abgesehen von den Literalknoten nicht miteinander verbunden sind, haben wir damit eine korrekte 3-Färbung von $G(\Phi)$.

n) Dies folgt aus h), da die Knoten y_6, DUMMY, FALSCH ein Dreieck bilden.

o) Da die Knoten α, β, γ eines Gadgets $G_9(c)$ mit den Literalknoten $A(\ell_1), A(\ell_2), A(\ell_3)$ verschmolzen sind, können diese nicht die Farbe DUMMY annehmen. Wenn also keiner der 3 WAHR ist, so müssen alle 3 FALSCH sein. Nach f) müsste dann aber y_6 ebenfalls FALSCH sein, Widerspruch. Folglich muss mindestens einer der 3 Literalknoten WAHR sein.

p) Wenn also $G(\Phi)$ eine 3-Färbung hat, so ist $G_9(c)$ für jede Klausel c der Eingabeformel korrekt gefärbt. Nach o) ist dann mindestens einer der 3 Knoten α, β, γ, welche mit den entsprechenden Literalknoten der Klausel c verschmolzen sind, WAHR gefärbt. Die Wahrheitsbelegung $\varphi : X \to \{0, 1\}$ ist dann

$$\varphi(x) := \begin{cases} 1 & , A(x) \text{ hat Farbe WAHR} \\ 0 & , \text{sonst} \end{cases}$$

Dann ist nämlich pro Klausel mindestens eines der vorkommenden Literale erfüllt und damit Φ erfüllt.

q) Wir gehen von endlichen Graphen aus. Man kann sich pro Zusammenhangskomponente des Graphen einen beliebigen "Startknoten" $v \in V$ wählen. Wir setzen o.B.d.A $c(v) := 0$. Sei nun $k \in V$ gefärbt. Dann setzen wir $c(\ell) := [c(k) + 1]_2$ für alle noch nicht gefärbten Knoten $\ell \in Adj(k)$. Nach endlich vielen Schritten werden alle Knoten gefärbt sein. Nun kann man in polynomieller Zeit testen, ob der Graph korrekt gefärbt ist. Da das vertauschen von Farben semantisch keine Rolle spielt, gibt es praktisch nur diese eine Möglichkeit den Graphen zu 2-färben. Damit lässt sich also in polynomieller Zeit entscheiden, ob ein endlicher Graph 2-färbbar ist.