Vorlesungsskript

Theoretische Informatik III

Sommersemester 2004

Prof. Dr. Johannes Köbler
Humboldt-Universität zu Berlin
Lehrstuhl Komplexität und Kryptografie

22. Juli 2004

Inhaltsverzeichnis

1 Relationalstrukturen

Sei A eine nichtleere Menge, R_i eine k_i-stellige Relation auf A, d.h. $R_i \subseteq A^{k_i}$ für $i = 1, \ldots, n$. Dann heißt $(A; R_1, \ldots, R_n)$ ***Relationalstruktur***. Die Menge A heißt der ***Individuenbereich*** oder die ***Trägermenge*** der Relationalstruktur.

Wir werden hier hauptsächlich den Fall $n = 1$, $k_1 = 2$, also (A, R) mit $R \subseteq A \times A$ betrachten. Man nennt dann R eine ***(binäre) Relation auf*** A.

Beispiel 1

- (F, M) *mit* $F := \{f \mid f$ *ist Fluss in Europa*$\}$ *und*
  $M := \{(f, g) \in F \times F \mid f$ *mündet in* $g\}$.
- (U, B) *mit* $U := \{x \mid x$ *ist Berliner* $\}$ *und*
  $B := \{(x, y) \in U \times U \mid x$ *ist Bruder von* $y\}$.
- $(\mathcal{P}(M), \subseteq)$*, wobei* $\mathcal{P}(M)$ *die Potenzmenge einer beliebigen Menge* M *und* $\subseteq$ *die Inklusionsbeziehung auf den Teilmengen von* M *ist.*
- (A, Id_A)*, wobei* $Id_A = \{(x, x) \mid x \in A\}$ *die* ***Identität auf*** A *ist.*
- $(\mathbb{R}, \leq)$.
- $(\mathbb{Z}, |)$*, wobei* $|$ *die "teilt"-Relation bezeichnet.*
- $(\mathcal{F}ml, \Rightarrow)$ *mit* $\mathcal{F}ml := \{F \mid F$ *ist aussagenlogische Formel*$\}$ *und*
  $\Rightarrow = \{(F, G) \in \mathcal{F}ml \times \mathcal{F}ml \mid G$ *ist aussagenlogische Folgerung von* $F\}$.

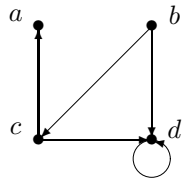
Oft wird für $(a, b) \in R$ auch die ***Infix***-Schreibweise aRb benutzt.

1.1 Darstellung endlicher Relationen

1.1.1 Graphische Darstellung

Eine Relation R auf einer endlichen Menge A kann durch einen ***gerichteten Graphen*** $G = (A, R)$ mit ***Knotenmenge*** A und ***Kantenmenge*** R veranschaulicht werden. Hier-

zu stellen wir jedes Element $x \in A$ als einen Knoten dar und verbinden jedes Knotenpaar $(x, y) \in R$ durch eine gerichtete Kante (Pfeil). Zwei durch eine Kante verbundene Knoten heißen ***adjazent*** (benachbart).



Der ***Ausgangsgrad*** eines Knotens $x \in V$ ist $d_{out}(x) = \|R(x)\|$, wobei $R(x) = \{y \in V \mid xRy\}$ der ***Nachbereich*** von x ist. Entsprechend ist $d_{in}(x) = \|\{y \in V \mid yRx\}\|$ der ***Eingangsgrad*** von x. Falls R symmetrisch (siehe unten) ist, können die Pfeilspitzen auch weggelassen werden. In diesem Fall ist $d(x) = d_{in}(x) = d_{out}(x)$ der ***Grad*** von x. Ist die zugrundeliegende Relation R zudem irreflexiv, so erhalten wir einen (schleifenfreien) ***Graphen***.

1.1.2 Matrixdarstellung (Adjazenzmatrix)

Eine Relation R auf einer endlichen (geordneten) Menge $A = \{a_1, \ldots, a_n\}$ lässt sich durch eine boolesche $n \times n$-Matrix $M_R = (m_{ij})$ mit

$$m_{ij} := \begin{cases} 1, & a_i R a_j, \\ 0, & \text{sonst} \end{cases}$$

darstellen. Beispielsweise hat die Relation

$$R = \{(b, c), (b, d), (c, a), (c, d), (d, d)\}$$

auf der Menge $A = \{a, b, c, d\}$ die Matrixdarstellung

$$M_R = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

1.1.3 Tabellendarstellung (Adjazenzliste)

Eine weitere Möglichkeit besteht darin, eine endliche Relation R in Form einer Tabelle darzustellen, die jedem Element $x \in A$ seinen Nachbereich $R(x)$ in Form einer Liste

zuordnet:

x	$R(x)$
a	-
b	c, d
c	a, d
d	d

1.2 Operationen auf Relationen

Da Relationen Mengen sind, sind auf ihnen die mengentheoretischen Operationen ***Durchschnitt***, ***Vereinigung***, ***Komplement*** und ***Differenz*** definiert. Seien R und S Relationen auf A, dann ist

$$\begin{array}{rcl} R \cap S & = & \{(x,y) \in A \times A \mid xRy \wedge xSy\}, \\ R \cup S & = & \{(x,y) \in A \times A \mid xRy \vee xSy\}, \\ R - S & = & \{(x,y) \in A \times A \mid xRy \wedge \neg xSy\} \\ \overline{R} & = & (A \times A) - R \end{array}$$

und $R \subseteq S$ bedeutet

$$\forall x, y : xRy \Rightarrow xSy.$$

Weiterhin ist für eine Menge $\mathcal{M} \subseteq \mathcal{P}(A \times A)$ von Relationen auf A der ***Schnitt über*** $\mathcal{M}$ und die ***Vereinigung über*** $\mathcal{M}$ wieder eine Relation auf A.

$$\bigcap \mathcal{M} = \{(x,y) \mid \forall R \in \mathcal{M} : xRy\}$$

$$\bigcup \mathcal{M} = \{(x,y) \mid \exists R \in \mathcal{M} : xRy\}$$

Die ***transponierte (konverse) Relation*** zu R ist

$$R^T := \{(y,x) \mid xRy\}.$$

R^T wird oft auch mit R^{-1} bezeichnet. Zum Beispiel ist $(\mathbb{R}, \leq^T) = (\mathbb{R}, \geq)$.

Sei R eine Relation auf A. Dann heißt R

reflexiv,	falls $\forall x \in A : xRx$ (also $Id_A \subseteq R$),
irreflexiv,	falls $\forall x \in A : \neg xRx$ (also $Id_A \subseteq \overline{R}$),
symmetrisch,	falls $\forall x, y \in A : xRy \Rightarrow yRx$ (also $R \subseteq R^T$),
asymmetrisch,	falls $\forall x, y \in A : xRy \Rightarrow \neg yRx$ (also $R \subseteq \overline{R^T}$),
antisymmetrisch,	falls $\forall x, y \in A : xRy \wedge yRx \Rightarrow x = y$ (also $R \cap R^T \subseteq Id$),
konnex,	falls $\forall x, y \in A : xRy \vee yRx$ (also $A \times A \subseteq R \cup R^T$),
semikonnex,	falls $\forall x, y \in A : x \neq y \Rightarrow xRy \vee yRx$ (also $\overline{Id} \subseteq R \cup R^T$),
transitiv,	falls $\forall x, y, z \in A : xRy \wedge yRz \Rightarrow xRz$ (also $R^2 \subseteq R$, s. Übungen)

gilt.

Beispiel 2

- *Die Relation "ist Schwester von" ist zwar in einer reinen Damengesellschaft symmetrisch, i.a. jedoch weder symmetrisch noch asymmetrisch noch antisymmetrisch.*
- $(\mathbb{R}, <)$ *ist irreflexiv, asymmetrisch, transitiv und semikonnex.*
- $(\mathbb{R}, \leq)$ *und* $(\mathcal{P}(M), \subseteq)$ *sind reflexiv, antisymmetrisch und transitiv.* $(\mathbb{R}, \leq)$ *ist konnex,* $(\mathcal{P}(M), \subseteq)$ *ist im Fall* $\|M\| \leq 1$ *konnex, aber im Fall* $\|M\| \geq 2$ *weder semikonnex noch konnex.*

Eine wichtige zweistellige Operation auf der Menge $\mathcal{P}(A \times A)$ aller Relationen auf A ist das Relationenprodukt (auch Komposition genannt).

Das ***Produkt*** zweier Relationen R und S auf A ist

$$R \circ S := \{(x, z) \mid \exists y : xRy \wedge ySz\}.$$

Übliche Bezeichnungen für das Relationenprodukt sind auch $R;S$ und $R \cdot S$ oder einfach RS. Für $\underbrace{R \circ \cdots \circ R}_{\text{n-mal}}$ wird auch R^n geschrieben.

Vorsicht: Das n-fache Relationenprodukt von R sollte nicht mit dem n-fachen kartesischen Produkt der *Menge* R verwechselt werden. Wir vereinbaren, dass R^n das n-fache Relationenprodukt bezeichnen soll, falls R eine Relation ist.

Beispiel 3 *Ist B die Relation "ist Bruder von", V "ist Vater von", M "ist Mutter von" und $E = V \cup M$ "ist Elternteil von", so ist $B \circ E$ die Relation "ist Onkel von".*

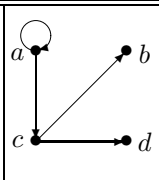
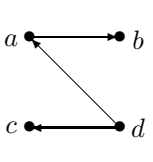
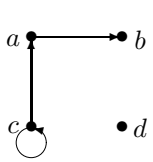
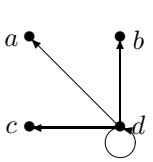
Sind $M_R = (r_{ij})$ und $M_S = (s_{ij})$ boolesche $n \times n$-Matrizen für R und S, so erhalten wir für $T = R \circ S$ die Matrix $M_T = (t_{ij})$ mit

$$t_{ij} := \bigvee_{k=1,\ldots,n} (r_{ik} \wedge s_{kj})$$

Der Nachbereich $R \circ S\,(x)$ von x bzgl. $R \circ S$ berechnet sich zu

$$R \circ S\,(x) = \bigcup \{S(y) \mid y \in R(x)\}.$$

Beispiel 4 *Wir betrachten auf der Menge $A = \{a, b, c, d\}$ die Relationen $R = \{(a, a), (a, c), (c, b), (c, d)\}$ und $S = \{(a, b), d, a), d, c)\}$.*

Relation	R	S	$R \circ S$	$S \circ R$
Graph				
Adjazenz-matrix	$\begin{pmatrix} 1&0&1&0\\ 0&0&0&0\\ 0&1&0&1\\ 0&0&0&0 \end{pmatrix}$	$\begin{pmatrix} 0&1&0&0\\ 0&0&0&0\\ 0&0&0&0\\ 1&0&1&0 \end{pmatrix}$	$\begin{pmatrix} 0&1&0&0\\ 0&0&0&0\\ 1&0&1&0\\ 0&0&0&0 \end{pmatrix}$	$\begin{pmatrix} 0&0&0&0\\ 0&0&0&0\\ 0&0&0&0\\ 1&1&1&1 \end{pmatrix}$
Adjazenz-liste	$a:\ a, c$ $b:$ - $c:\ b, d$ $d:$ -	$a:\ b$ $b:$ - $c:$ - $d:\ a, c$	$a:\ b$ $b:$ - $c:\ a, c$ $d:$ -	$a:$ - $b:$ - $c:$ - $d:\ a, b, c, d$

Beobachtung: Das Relationenprodukt ist nicht kommutativ, d.h. i.a. gilt nicht $R \circ S = S \circ R$.

Satz 5 *Seien Q, R, S Relationen auf A. Dann gilt*

(i) $(Q \circ R) \circ S = Q \circ (R \circ S)$, *d.h. $\circ$ ist assoziativ,*

(ii) $Id \circ R = R \circ Id = R$, *d.h. Id ist neutrales Element.*

Beweis:

$$
\begin{array}{lll}
i) \quad x\ (Q \circ R) \circ S\ y & \Leftrightarrow & \exists u \in A : x\ (Q \circ R)\ u\ \wedge\ u\ S\ y \\
& \Leftrightarrow & \exists u \in A : (\exists v \in A : x\ Q\ v\ R\ u)\ \wedge\ u\ S\ y \\
& \Leftrightarrow & \exists u, v \in A : x\ Q\ v\ R\ u\ S\ y \\
& \Leftrightarrow & \exists v \in A : x\ Q\ v\ \wedge\ (\exists u \in A : v\ R\ u\ \wedge\ u\ S\ y) \\
& \Leftrightarrow & \exists v \in A : x\ Q\ v\ (R \circ S)\ y \\
& \Leftrightarrow & x\ Q \circ (R \circ S)\ y
\end{array}
$$

$$ii) \quad x\ Id \circ R\ y \Leftrightarrow x = x\ R\ y \Leftrightarrow x\ R\ y \Leftrightarrow xRy = y \Leftrightarrow x\ R \circ Id\ y.$$

□

Es lässt sich leicht nachprüfen, dass der Schnitt über eine Menge transitiver Relationen wieder transitiv ist, d.h. es existiert zu jeder Relation R auf einer Menge A eine kleinste, R umfassende transitive Relation.

Die Relation

$$R^+ := \bigcap \{S \subseteq A \times A \mid R \subseteq S,\ S \text{ transitiv}\}$$

heißt die ***transitive Hülle*** von R und

$$R^* := \bigcap \{S \subseteq A \times A \mid R \subseteq S,\ S \text{ reflexiv und transitiv}\}$$

heißt die ***reflexiv-transitive Hülle*** von R.

Satz 6

(i) $R^+ = \bigcup_{i \geq 1} R^i$,

(ii) $R^* = \bigcup_{i \geq 0} R^i$, *wobei* $R^0 := Id$.

Beweis: siehe Übungen.

1.3 Äquivalenz- und Ordnungsrelationen

Die nachfolgende Tabelle gibt einen Überblick über die in diesem Abschnitt behandelten Relationalstrukturen.

	refl	sym.	trans.	asym.	antisym.	konnex	semikon.
Äquivalenzrel.	√	√	√				
(Halb-)Ordnung	√		√		√		
Striktordnung			√	√			
lineare Ord.			√		√	√	
lin. Striktord.			√	√			√
schwache Ord.			√			√	
Quasiordnung	√		√				

In der Tabelle sind nur die definierenden Eigenschaften durch ein "√" gekennzeichnet. Das schließt nicht aus, dass gleichzeitig auch noch weitere Eigenschaften vorliegen.

Als erstes wenden wir uns den ***Äquivalenzrelationen***, d.h. reflexiven, symmetrischen und transitiven Relationen zu.

Beispiele für Äquivalenzrelationen

a) Auf der Menge aller Geraden im $\mathbb{R}^2$ die Parallelität, deren Äquivalenzklassen aus jeweils allen Geraden mit derselben Richtung (oder Steigung) bestehen.

b) Auf der Menge aller Menschen "im gleichen Jahr geboren wie", d.h. alle Menschen eines Jahrgangs stehen in Relation zueinander.

c) Auf $\mathbb{Z}$ die Restgleichheit bei Division durch eine feste ganze Zahl m. Hierdurch werden die ganzen Zahlen in die m Restklassen modulo m unterteilt.

d) Auf der Menge aller aussagenlogischen Formeln die semantische Äquivalenz.

Den bzgl. einer Äquivalenzrelation E zu x gehörigen Nachbereich $E(x)$ nennt man ***die von*** x ***repräsentierte Äquivalenzklasse*** und bezeichnet sie mit $[x]_E$. Die durch E auf

M induzierte Partition $\{[x]_E \mid x \in M\}$ wird ***Quotienten- oder Faktormenge*** genannt und mit M/E bezeichnet.

Die kleinste Äquivalenzrelation auf M ist die Identität Id_M, die größte die Allrelation $M \times M$. Im Falle der Identität enthält jede Äquivalenzklasse nur ein Element, d. h. $M/Id_M = \{\{x\} \mid x \in M\}$. Im Falle der Allrelation $M \times M$ gibt es nur eine Äquivalenzklasse M, d.h. $M/(M \times M) = \{M\}$. Sind allgemein E_1, E_2 zwei Äquivalenzrelationen auf M mit $E_1 \subseteq E_2$, so ist jede Äquivalenzklasse von E_1 in einer Äquivalenzklasse von E_2 enthalten, d.h. jede Äquivalenzklasse von E_2 ist die Vereinigung von Äquivalenzklassen von E_1. Man sagt auch, E_1 bewirkt eine "feinere" Klasseneinteilung als E_2. Demnach ist die Identität die feinste und die Allrelation die gröbste Äquivalenzrelation.

Da der Schnitt über eine Menge von Äquivalenzrelationen wieder eine Äquivalenzrelation ist, können wir für eine beliebige Relation R auf einer Menge A die kleinste R umfassende Äquivalenzrelation

$$h_{\text{äq}}(R) := \bigcap\{E \subseteq A \times A \mid R \subseteq E,\ E \text{ ist eine Äquivalenzrelation}\}$$

definieren.

Als nächstes betrachten wir ***Ordnungen***, d.h. reflexive, antisymmetrische und transitive Relationen; auch ***(reflexive) Halbordnungen*** oder ***partielle Ordnungen*** genannt.

Beispiele für Ordnungen

a) $(\mathcal{P}(M); \subseteq),\ (\mathbb{Z}, \leq),\ (\mathbb{N}, |)$.

b) Auf der Menge $\mathcal{A}(M)$ aller Äquivalenzrelationen auf M die Relation "feiner als". Dabei ist, wie wir gesehen haben, E_1 feiner als E_2, falls E_1 in E_2 enthalten ist. In diesem Fall bewirkt E_1 eine feinere Klasseneinteilung auf M als E_2, da jede Äquivalenzklasse von E_1 in einer Äquivalenzklasse von E_2 enthalten ist.

c) Ist R eine Ordnung auf A und $B \subseteq A$, so ist $R \cap (B \times B)$ eine Ordnung auf B (die ***Einschränkung*** von R auf B). Beispielsweise ist $(\mathcal{A}(M); \subseteq)$ die Einschränkung von $(\mathcal{P}(M \times M); \subseteq)$ auf $\mathcal{A}(M)$.

Wir untersuchen nun den Bezug zu ***Striktordnungen***, d.h. transitiven und asymmetrischen Relationen.

Zu den Ordnungen $(\mathcal{P}(M); \subseteq)$ und $(\mathbb{Z}; \leq)$ gehören die Striktordnungen $(\mathcal{P}(M); \subsetneq)$ und $(\mathbb{Z}; <)$. Der nächste Satz zeigt, dass ganz allgemein zu jeder Ordnung eine Striktordnung und umgekehrt zu jeder Striktordnung eine Ordnung gehört.

Satz 7 *Falls R eine Striktordnung ist, ist $R \cup Id$ eine Ordnung. Falls R eine Ordnung ist, ist $R \setminus Id$ eine Striktordnung.*

Bemerkung 8 *Der leichteren Lesbarkeit willen verwenden wir das Symbol $<$ für eine Striktordnung und das Symbol $\leq$ für die zugehörige Ordnung.*

Beweis: Sei zunächst $<$ eine Striktordnung. Wir zeigen, dass dann $\leq := < \cup Id$ eine Ordnung ist.

reflexiv: Klar.

antisymmetrisch: Gelte $x \leq y \wedge y \leq x$. Aus der Annahme $x \neq y$ folgt $x < y \wedge y < x$ im Widerspruch zur Asymmetrie von $<$.

transitiv: Sei $x \leq y \wedge y \leq z$.
1. Fall: $x = y \vee y = z$: Klar.
2. Fall: $x \neq y \wedge y \neq z$. Nun folgt $x < y < z$, also $x < z$ und damit erst recht $x \leq z$.

Sei nun $\leq$ eine Ordnung. Wir zeigen, dass dann $< = \leq \setminus Id = \{(x, y) \mid x \leq y, x \neq y\}$ eine Striktordnung ist. Man beachte, dass $<$ irreflexiv ist.

asymmetrisch: Aus der Annahme $x < y$ und $y < x$ folgt zunächst $x \leq y$ und $y \leq x$. Da $\leq$ antisymmetrisch ist, folgt $x = y$ und somit $x < x$ im Widerspruch zur Irreflexivität von $<$.

transitiv: Sei $x < y < z$. Dann folgt $x \leq y \leq z$ und somit $x \leq z$, da $\leq$ transitiv ist. Aus der Annahme $x = z$ folgt $z \leq y$, also $z = y$ (wegen $y \leq z$ und der Antisymmetrie von $\leq$), was im Widerspruch zur Irreflexivität von $<$ steht.

■

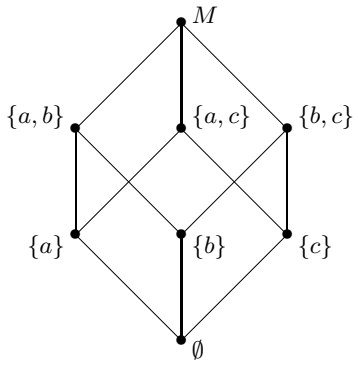
Zur graphischen Darstellung von Striktordnungen $<$ bzw. der zugehörigen Ordnungen $\leq$ werden gerne so genannte ***Hasse–Diagramme*** verwendet. Dabei wird nur der Graph der Nachbarrelation $\lhd := < \setminus <^2$, d.h.

$$x \lhd y \;:\Leftrightarrow\; x < y \wedge \neg\exists z : x < z < y$$

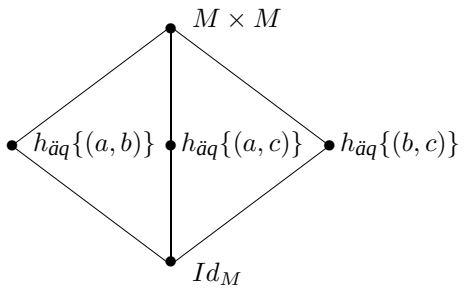
gezeichnet. Für $x \lhd y$ sagt man auch, y ist ***oberer Nachbar*** von x. Weiterhin wird im Fall $x \lhd y$ der Knoten y oberhalb vom Knoten x gezeichnet, so dass auf Pfeilspitzen verzichtet werden kann.

Beispiel 9

- *Sei* $M = \{a, b, c\}$.

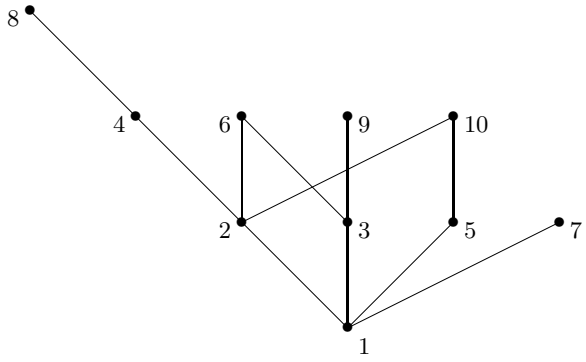


$(\mathcal{P}(M);\subseteq)$



$(\mathcal{A}(M);\subseteq)$

- *Die Einschränkung der "teilt"-Relation auf* $\{1,2,\ldots,10\}$*:*



Sei $\leq$ eine Ordnung auf A und sei $H \subseteq A$. Ein Element $a \in H$ heißt ***kleinstes Element*** oder ***Minimum*** von H, falls $a \leq b$ für alle $b \in H$ gilt. Gilt entsprechend $a \geq b$ für alle $b \in H$, so heißt $a \in H$ ***größtes Element*** oder ***Maximum*** von H.

Aufgrund der Antisymmetrie kann es in H höchstens ein kleinstes und höchstens ein größtes Element geben. Besteht H beispielsweise aus zwei ***unvergleichbaren*** Elementen a und b, d.h. gilt weder $a \leq b$ noch $b \leq a$, so besitzt $H = \{a,b\}$ weder ein kleinstes noch ein größtes Element.

Ein Element $a \in H$ heißt **minimal** in H, falls es in H keine kleineren Elemente gibt, d.h. es gilt für alle $b \in H$,

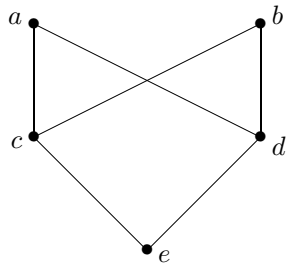
$$b \leq a \Rightarrow b = a.$$

Entsprechend heißt ein Element $a \in H$ ***maximal***, wenn

$$\forall b \in H : a \leq b \Rightarrow a = b$$

gilt, also keine echt größeren Elemente in H existieren.

In folgender Ordnung



besitzt beispielsweise die Teilmenge $\{a, b, c\}$ das kleinste Element c, aber kein größtes Element. Die Menge $\{a, b, c, e\}$ besitzt die beiden maximalen Elemente a und b, sowie ein minimales Element e.

Jedes Element $a \in A$ mit $a \leq b$ für alle $b \in H$ heißt ***untere*** und jedes $c \in A$ mit $b \leq c$ für alle $b \in H$ heißt ***obere Schranke*** von H. Eine Teilmenge H von A heißt ***nach oben beschränkt***, wenn (mindestens) eine obere Schranke für H existiert, und ***nach unten beschränkt***, wenn es eine untere Schranke für H gibt. Eine nach oben und unten beschränkte Teilmenge heißt ***beschränkt***.

In obigem Beispiel hat die Menge $H = \{a, b, c\}$ die beiden unteren Schranken c und e; obere Schranken existieren nicht.

Besitzt eine Teilmenge H eine größte untere Schranke, d.h. besitzt die Menge aller unteren Schranken von H ein größtes Element, so heißt dieses das ***Infimum von*** H ($\inf H$). Gilt entsprechend für ein Element $s \in A$

$$(\forall b \in H : b \leq s) \wedge (\forall s' \in A : (\forall b \in H : b \leq s') \Rightarrow s \leq s'),$$

d.h. ist s die kleinste obere Schranke von H, so heißt s das ***Supremum von*** H ($\sup H$). Aufgrund der Definition ist klar, dass es höchstens ein Supremum und höchstens ein Infimum für jede Teilmenge gibt.

Im vorigen Beispiel besitzt die Teilmenge $H = \{a, b, c\}$ das Infimum $\inf H = c$, aber kein Supremum, da keine oberen Schranken für H existieren. Das Infimum von $\{a, b\}$ existiert ebenfalls nicht, da die Menge $\{c, d, e\}$ der unteren Schranken von $\{a, b\}$ kein größtes Element besitzt. Das Infimum von $\{c, d\}$ ist e.

Auch in linearen Ordnungen muss nicht jede Teilmenge ein Supremum oder Infimum besitzen. So hat in der linear geordneten Menge $(\mathbb{Q}, \leq)$ die Teilmenge $H := \{x \in \mathbb{Q} \mid x^2 \leq 2\}$ weder ein Supremum noch ein Infimum. Natürlich hat in linearen Ordnungen jede endliche Teilmenge ein kleinstes und ein größtes Element und somit erst recht ein Supremum und ein Infimum.

Eine Teilmenge H von A heißt ***Kette***, falls je zwei Elemente vergleichbar sind, d.h. es gilt für alle $a, b \in H$,

$$a \leq b \vee b \leq a.$$

Im Beispiel bilden $\{a, c, e\}$, $\{a, d, e\}$, $\{b, c, e\}$ und $\{b, d, e\}$ sowie alle Teilmengen hiervon Ketten.

Eine Teilmenge H von A heißt ***Antikette***, falls je zwei Elemente unvergleichbar sind, d.h. es gilt für alle $a, b \in H$,

$$a \not< b \wedge b \not< a.$$

Im Beispiel bilden $\{a, b\}$, $\{c, d\}$ und $\{e\}$ sowie alle Teilmengen hiervon Antiketten.

1.4 Abbildungen

Eine binäre Relation R auf einer Menge M heißt ***rechtseindeutig***, falls

$$xRy \wedge xRz \Rightarrow y = z$$

und ***linkseindeutig***, falls

$$xRz \wedge yRz \Rightarrow x = y$$

für alle $x, y, z \in M$ gilt.

Der ***Nachbereich*** $N(R)$ und der ***Vorbereich*** $V(R)$ von R ist

$$N(R) := \bigcup_{x \in M} R(x) \text{ und } V(R) := \bigcup_{x \in M} R^T(x).$$

Eine rechtseindeutige Relation R mit $V(R) = A$ und $N(R) \subseteq B$ heißt ***Abbildung*** oder ***Funktion*** von A nach B (kurz $R : A \to B$). Wie üblich werden wir Abbildungen meist mit kleinen Buchstaben $f, g, h, ...$ bezeichnen und für $(x, y) \in f$ nicht xfy sondern $f(x) = y$ oder $f : x \mapsto y$ schreiben.

Ist $f : A \to B$ eine Abbildung, so wird der Vorbereich $V(f) = A$ der ***Definitionsbereich*** und die Menge B der ***Wertebereich*** oder ***Wertevorrat*** von f genannt. Der Nachbereich $N(f)$ wird als ***Bild*** von f bezeichnet. Im Fall $N(f) = B$ heißt f ***surjektiv***. Ist f linkseindeutig, so heißt f ***injektiv***. In diesem Fall impliziert $f(x) = f(y)$ die Gleichheit $x = y$. Eine injektive und surjektive Abbildung heißt ***bijektiv***. Für eine injektive Abbildung $f : A \to B$ ist auch f^T eine Abbildung, die mit f^{-1} bezeichnet und die ***inverse Abbildung*** zu f genannt wird. Man beachte, dass der Definitionsbereich $V(f^{-1}) = N(f)$ nur dann gleich B ist, wenn f auch surjektiv, also eine Bijektion ist.

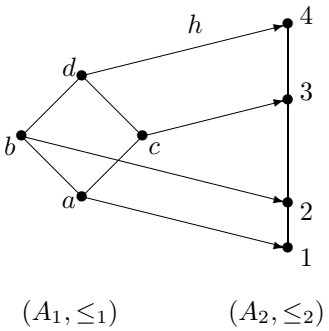
1.5 Homo- und Isomorphismen

Seien (A_1, R_1) und (A_2, R_2) Relationalstrukturen. Eine Abbildung $h : A_1 \to A_2$ heißt ***Homomorphismus***, falls für alle $a, b \in A_1$ gilt:

$$aR_1b \Rightarrow h(a)R_2h(b).$$

Sind (A_1, R_1) und (A_2, R_2) Ordnungen, so spricht man von ***Ordnungshomomorphismen*** oder einfach von ***monotonen*** Abbildungen. Injektive Ordnungshomomorphismen werden auch ***streng monotone*** Abbildungen genannt.

Beispiel 10 *Folgende Abbildung $h : A_1 \to A_2$ ist ein bijektiver Homomorphismus. (Das Bild zeigt die Hasse-Diagramme der beiden Ordnungen $(A_1, \leq_1)$ und $(A_2, \leq_2)$).*



$(A_1, \leq_1)$ $(A_2, \leq_2)$

Bemerkung 11 *Die Umkehrung eines bijektiven Homomorphismus muss kein Homomorphismus sein (siehe Beispiel: wegen $2 \leq_2 3$ und $h^{-1}(2) = b \not\leq_1 c = h^{-1}(3)$ ist h^{-1} nicht monoton). Ist dies jedoch der Fall, so spricht man von einem Isomorphismus.*

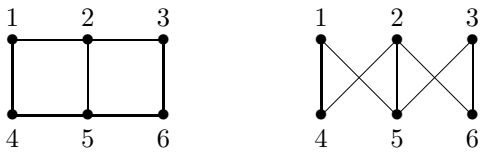
Definition 12 (***Isomorphismus***)

Ein bijektiver Homomorphismus $h : A_1 \to A_2$, bei dem auch h^{-1} ein Homomorphismus ist, d.h. es gilt

$$\forall a, b \in A_1 : aR_1b \Leftrightarrow h(a)R_2h(b).$$

heißt ***Isomorphismus***. In diesem Fall heißen die Strukturen (A_1, R_1) und (A_2, R_2) ***isomorph*** (kurz: $(A_1, R_1) \cong (A_2, R_2)$).

Beispiel 13

- 

x	1	2	3	4	5	6
$h_1(x)$	1	5	3	4	2	6
$h_2(x)$	4	5	6	1	2	3

- *Die Bijektion $h : x \to e^x$ ist ein Ordnungsisomorphismus zwischen $(\mathbb{R}, \leq)$ und $((0, \infty), \leq)$.*

- *Für $n \in \mathbb{N}$ sei*
$$T_n := \{k \in \mathbb{N} \mid k \text{ teilt } n\}$$
die Menge aller Teiler von n und $P_n := T_n \cap$ Prim die Menge aller Primteiler von n. Dann ist für quadratfreies n, d.h. es gibt kein $k \in \mathbb{N}$, so dass k^2 die Zahl n teilt, die Abbildung $h : k \mapsto P_k$ ein Ordnungsisomorphismus zwischen $(T_n, |)$ und $(\mathcal{P}(P_n), \subseteq)$.

- *Während auf der Menge $V = \{1, 2, 3\}$ genau $2^3 = 8$ (allgemein $2^{\binom{n}{2}}$) verschiedene Graphen existieren, gibt es auf dieser Menge nur 4 verschiedene nichtisomorphe Graphen:*

Da auf der Knotenmenge $V = \{1, \ldots, n\}$ maximal $n!$ Graphen zueinander isomorph sein können, muss es mindestens $2^{\binom{n}{2}}/n!$ verschiedene nichtisomorphe Graphen auf V geben. Tatsächlich sind es $2^{\binom{n}{2}} \cdot \frac{1+o(1)}{n!}$, also auch nicht sehr viel mehr.

- *Es existieren genau 5 nichtisomorphe Ordnungen mit 3 Elementen:*

Anders ausgedrückt: Die Klasse aller dreielementigen Ordnungen zerfällt unter der Äquivalenzrelation $\cong$ in fünf Äquivalenzklassen, die durch obige fünf Hasse-Diagramme repräsentiert werden.

2 Graphen

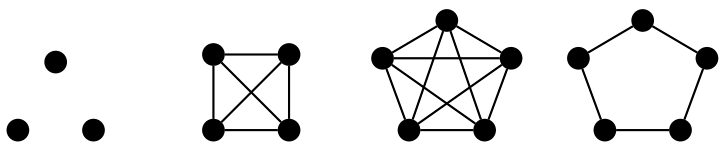
Wir betrachten zunächst nur ungerichtete Graphen $G = (V, E)$, d.h. E ist eine symmetrische und irreflexive Relation auf V. Da in diesem Fall für jedes Paar $(x, y) \in R$ auch (y, x) in R enthalten ist und $x \neq y$ gilt, können wir die geordneten Paare (x, y) bzw. (y, x) durch ungeordnete Paare $\{x, y\}$ ersetzen. Dies führt uns auf folgende Definition.

Definition 14 (***Graph***)

> Sei V eine endliche Menge. Ein (endlicher, ungerichteter) ***Graph*** ist ein Paar $G = (V, E)$ mit $E \subseteq \binom{V}{2}$, wobei $\binom{V}{2} := \{e \subseteq V \mid \|e\| = 2\}$ aus allen zweielementigen Teilmengen von V besteht.

Die Anzahl $\|V\|$ der Knoten wird die ***Ordnung*** von G genannt und mit $n(G)$ oder einfach mit n bezeichnet. Als Knotenmenge verwenden wir meist die Menge $V = \{1, \ldots, n\}$, für die wir auch kurz $[n]$ schreiben. Die Anzahl $\|E\|$ der Kanten wird die ***Größe*** von G genannt und mit $m(G)$ oder einfach mit m bezeichnet.

Beispiel 15 *Der kleinste Graph auf der Knotenmenge $V = [n]$ ist $E_n = ([n], \emptyset)$ und der größte ist $K_n = ([n], \binom{[n]}{2})$. Jeder zu E_n isomorphe Graph heißt **leer** und jeder zu K_n isomorphe Graph heißt **vollständig**. Ein Beispiel für einen Graphen mit $m = n$ Kanten ist $C_n = ([n], E)$ für $n \geq 3$, wobei $E = \{\{i, i+1\} \mid 1 \leq i \leq n-1\} \cup \{\{1, n\}\}$ ist.*



E_3 K_4 K_5 C_5

Eine Kante $e = \{u, v\} \in E$ ***verbindet*** die beiden ***Endpunkte*** u und v. Man sagt auch, e ***inzidiert*** mit den Knoten u und v, e und u bzw. e und v sind ***inzident***, oder die Knoten u und v sind ***adjazent*** (benachbart).

Der zu $G = (V, E)$ ***komplementäre*** Graph ist $\hat{G} = (V, \bar{E})$ mit $\bar{E} = \binom{V}{2} \setminus E$. Beispielsweise sind der leere Graph E_n und der vollständige Graph K_n komplementär.

Wie bereits im vorigen Abschnitt bemerkt, gibt es auf einer n-elementigen Knotenmenge genau $2^{\binom{n}{2}}$ verschiedene Graphen, wovon $2^{\binom{n}{2}} \cdot \frac{1+o(1)}{n!}$ paarweise nichtisomorph sind.

Die ***Nachbarschaft*** eines Knotens $u \in V$ ist die Knotenmenge $\Gamma(u) = \{v \in V \mid \{u, v\} \in E\}$. Die Anzahl $\|\Gamma(u)\|$ der Nachbarn ist der ***Grad*** von u und wird mit $d(u)$ bezeichnet. Da jede Kante mit zwei Knoten inzidiert, gilt

$$\sum_{u \in V} d(u) = 2m(G).$$

Der ***Maximalgrad*** von G ist

$$\Delta(G) = \max_{u \in V} d(u)$$

und der ***Minimalgrad*** von G ist

$$\delta(G) = \min_{u \in V} d(u).$$

Im Fall $\Delta(G) = \delta(G)$ heißt G ***regulär***.

Ein ***Weg*** ist eine Folge $W = v_0, v_1, \ldots, v_l$ von Knoten mit $l \geq 0$ und $\{v_i, v_{i+1}\} \in E$ für $i = 0, \ldots, l-1$. v_0 heißt der ***Anfangs-*** und v_l der ***Endknoten*** von W. Man sagt auch, W ist ein Weg von v_0 nach v_l. Die Anzahl l der dabei durchlaufenen Kanten wird als ***Länge*** $l(W)$ von W bezeichnet. Existiert zwischen je zwei Knoten $u, v \in V$ ein Weg, so heißt G ***zusammenhängend***.

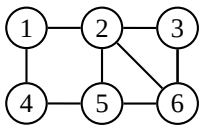
Ein ***Pfad*** oder ***einfacher Weg*** ist ein Weg $P = v_0, v_1, \ldots, v_l$, bei dem alle Knoten v_i paarweise verschieden sind. Ein ***Kreis*** ist ein Weg $K = v_0, v_1, \ldots, v_l$ mit $l \geq 3$, $v_0 = v_l$ und $v_i \neq v_j$ für $1 \leq i < j \leq l$, d.h. $v_1, \ldots, v_l$ ist ein Pfad. Ein Graph, der keinen Kreis besitzt, heißt ***kreisfrei*** oder ***azyklisch***.

$G' = (V', E')$ ist ***Teilgraph*** von $G = (V, E)$, falls $V' \subseteq V$ und $E' \subseteq E \cap \binom{V'}{2}$ gilt. Im Fall $V' = V$ wird G' auch ein ***(auf)spannender*** Teilgraph von G genannt. Gilt dagegen $V' \subseteq V$ und $E' = E \cap \binom{V'}{2}$, so wird G' der von V' in G ***induzierte*** Teilgraph genannt und mit $G[V']$ bezeichnet. Der Subgraph $G[V - U]$ von G, der aus G durch Entfernung aller Knoten $u \in U$ sowie aller mit diesen Knoten inzidenten Kanten entsteht, bezeichnen wir auch kurz mit $G - U$ bzw. mit $G - u$, falls $U = \{u\}$ nur einen Knoten enthält.

2.1 Cliquen und Stabilität

Definition 16 (***Clique***)

Sei $G = (V, E)$ ein Graph. Dann heißt $C \subseteq V$ ***Clique*** in G, falls $G[C]$ vollständig ist. Ist dagegen $G[S]$ leer, so heißt $S \subseteq V$ ***stabil*** oder ***unabhängig*** in G. Eine Knotenmenge $K \subseteq V$ heißt ***Knotenüberdeckung***, falls für alle $e \in E$ gilt: $e \cap K \neq \emptyset$.



$C = \{2, 5, 6\}$
$S = \{1, 3, 5\}$

Die Cliquenzahl von G ist

$$\omega(G) = \max\{\|C\| \mid C \text{ ist Clique in } G\}$$

und die Stabilitätszahl von G ist

$$\alpha(G) = \max\{\|S\| \mid S \text{ ist stabil in } G\}.$$

Beispiel 17

$$\omega(K_n) = n, \quad \omega(C_n) = \begin{cases} 3, & n = 3 \\ 2, & n \geq 4, \end{cases}$$

$$\alpha(K_n) = 1, \quad \alpha(C_n) = \left\lfloor \frac{n}{2} \right\rfloor.$$

Lemma 18 *(i)* $\omega(G) = \alpha(\bar{G})$,

(ii) $1 \leq \omega(G) \leq \Delta(G) + 1$,

(iii) $\frac{n}{\Delta(G)+1} \leq \alpha(G) \leq n$.

Beweis:

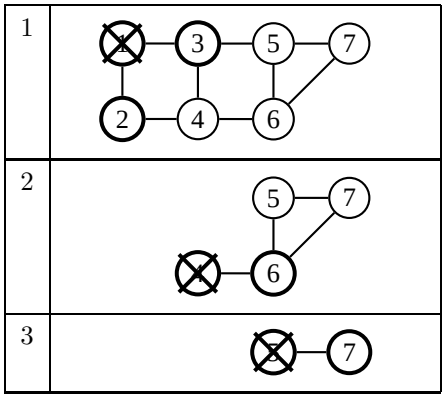
(i) Da C genau dann eine Clique in G ist, wenn C stabil in $\bar{G}$ ist, folgt

$$\max\{\|C\| \mid C \text{ ist Clique in } G\} = \max\{\|S\| \mid S \text{ ist stabil in } \bar{G}\}.$$

(ii) $\omega(G) \geq 1$ ist klar. $\omega(G) \leq \Delta(G) + 1$ folgt aus der Tatsache, dass der Grad jedes Knotens u in einer Clique C mindestens $\|C\| - 1$ beträgt.

(iii) $\alpha(G) \leq n$ ist klar. Für $\alpha(G) \geq \frac{n}{\Delta(G)+1}$ betrachten wir folgenden Algorithmus.

Eingabe: $G = (V, E)$
  $S \leftarrow \emptyset$
  repeat wähle Knoten $u \in V$
    $S \leftarrow S \cup \{u\}$
    $G \leftarrow G - (\Gamma(u) \cup \{u\})$
  until $V = \emptyset$
Ausgabe: S



Da G zu Beginn n Knoten hat und da in jedem Schleifendurchlauf höchstens

$$\|\Gamma(u) \cup \{u\}\| = 1 + \|\Gamma(u)\| \leq 1 + \Delta(G)$$

Knoten entfernt werden, ist die Anzahl der Schleifendurchläufe (und damit die Größe der stabilen Menge S) mindestens $\frac{n}{\Delta(G)+1}$.

■

Für einen gegebenen endlichen Graphen G und eine Zahl k betrachten wir nun die folgenden graphentheoretischen Fragestellungen:

Clique: Besitzt G eine Clique der Größe k?

IndependentSet (IS): Besitzt G eine stabile Knotenmenge der Größe k?

NodeCover (NC): Besitzt G eine Knotenüberdeckung der Größe k?

Satz 19 *IS ist $\mathcal{NP}$-vollständig.*

Beweis: Wir zeigen 3-SAT $\leq_p$ IS. Sei

$$F = \bigwedge_{i=1}^{m} \underbrace{C_i}_{\bigvee_{j=1}^{k_i} l_{ij}}$$

mit $k_i \leq 3$ und $l_{ij} \in \{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$. Betrachte den folgenden Graphen $G = (V, E)$ mit

$$\begin{aligned} V &= \{(i,j) \mid 1 \leq i \leq m, 1 \leq j \leq k_i\} \\ E &= \{\{(i,j),(i,j')\} \mid 1 \leq i \leq m, 1 \leq j < j' \leq k_i\} \\ &\quad \cup \{\{(s,t),(u,v)\} \mid l_{st} \text{ und } l_{uv} \text{ sind komplementär}\}. \end{aligned}$$

Dabei heißen zwei Literale **komplementär**, wenn das eine die Negation des anderen ist. Nun gilt

$$\begin{aligned} F \in \text{3-SAT} &\Leftrightarrow \text{es gibt eine Belegung, die in jeder Klausel } C_i \text{ mindestens ein Literal wahr macht} \\ &\Leftrightarrow \text{es gibt } m \text{ Literale } l_{1,j_1}, \ldots, l_{m,j_m}, \text{ die paarweise nicht komplementär sind} \\ &\Leftrightarrow \text{es gibt } m \text{ Knoten } (1, j_1), \ldots, (m, j_m), \text{ die nicht durch Kanten verbunden sind} \\ &\Leftrightarrow G \text{ besitzt eine stabile Knotenmenge der Größe } m. \end{aligned}$$

■

Korollar 20 *Clique ist $\mathcal{NP}$-vollständig.*

Beweis: Siehe Übungen. ■

Korollar 21 *NC ist $\mathcal{NP}$-vollständig.*

Beweis: Da I genau dann stabil ist, wenn $V \setminus I$ eine Knotenüberdeckung ist, können wir mittels der Reduktionsfunktion

$$f(G, k) = (G, n - k)$$

IS auf NC reduzieren, wobei $n = \|V\|$ die Anzahl der Knoten in G ist. ■

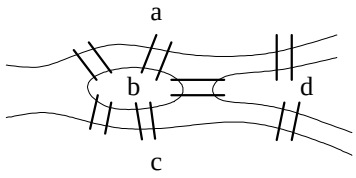
2.2 Euler- und Hamiltonkreise

Sei $G = (V, E)$ ein zusammenhängender Graph. Ein Weg $W = v_0, v_1, \ldots, v_l$ heißt ***Eulerweg*** in G, falls jede Kante in E genau einmal durchlaufen wird, d.h. es gilt

$$\{\{v_i, v_{i+1}\} \mid 0 \leq i \leq l - 1\} = E \text{ und } l(W) = m(G).$$

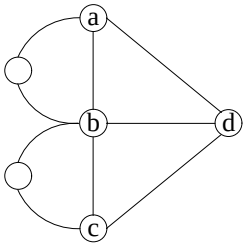
Gilt zudem $v_l = v_0$, so heißt W ***Eulerkreis***. Man beachte, dass ein Eulerkreis kein Kreis sein muss, da er einzelne Knoten mehrmals durchlaufen kann.

Königsberger Brückenproblem:



Frage: Gibt es einen Spaziergang über alle 7 Brücken, bei dem keine Brücke mehrmals überquert werden muss und der zum Ausgangspunkt zurückführt?

Gelöst von Euler (1707 – 1783) durch Betrachtung des folgenden Graphen, der offenbar genau dann einen Eulerkreis hat, wenn die Antwort auf obige Frage „ja“ ist.



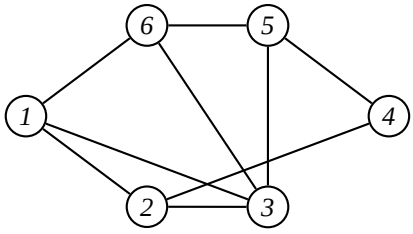
Satz 22 (Euler, 1736) *Ein zusammenhängender Graph $G = (V, E)$ besitzt genau dann einen Eulerkreis, wenn all seine Knoten geraden Grad haben.*

(Beweis siehe Übungen.)

Ein Kreis $K = v_0, v_1, \ldots, v_l$ heißt ***Hamiltonkreis*** in G, falls jeder Knoten in V genau einmal durchlaufen wird, d.h. es gilt

$$\{v_i \mid 0 \leq i \leq l\} = V \text{ und } l(W) = n(G).$$

Beispiel 23 *Der Graph*



besitzt den Hamiltonkreis $K = 1, 6, 3, 5, 4, 2, 1$.

Aufgrund des Satzes von Euler bereitet es keine Schwierigkeiten, für einen gegebenen Graphen zu entscheiden, ob er einen Eulerkreis besitzt. Auch das Auffinden eines Eulerkreises ist nicht schwierig (siehe Übungen). Dagegen ist das Problem, die Existenz eines Hamiltonkreises zu entscheiden, $\mathcal{NP}$-vollständig.

2.3 Bäume

Definition 24 (***Baum***)

Ein ***Baum*** ist ein zusammenhängender und azyklischer Graph (V, E). Jeder Knoten $u \in V$ vom Grad $d(u) \leq 1$ heißt ***Blatt*** (oder Endknoten) und die übrigen Knoten (vom Grad ≥ 2) heißen ***innere Knoten***.

Lemma 25 *Jeder Baum mit $n \geq 2$ Knoten hat mindestens 2 Blätter.*

Beweis: Ausgehend von einem beliebigen Knoten u_0 lässt sich ein Weg $P = u_0, u_1, \ldots, u_l$ konstruieren, indem man für $i = 0, 1, \ldots, l-1$ jeweils eine noch nicht benutzte Kante $\{u_i, u_{i+1}\}$ wählt. Da T ein Baum ist, gilt $u_i \neq u_j$ für $i \neq j$. Da T endlich ist, muss daher nach endlich vielen Schritten ein Knoten u_l erreicht werden, der keine Fortsetzung von P erlaubt, da er nur mit der Kante $\{u_{l-1}, u_l\}$ inzidiert. Also ist u_l ein Blatt und ein weiteres Blatt lässt sich dadurch finden, dass man dieses Verfahren mit dem Startknoten u_l wiederholt. ■

Ein Baum $T = (V, E)$ mit einem ausgezeichneten Knoten $r \in V$ (der ***Wurzel*** genannt wird) heißt ***Wurzelbaum*** und wird mit (T, r) bezeichnet.

2.4 Bipartite Graphen

Definition 26 (***bipartit***)

Ein Graph $G = (V, E)$ heißt ***bipartit***, falls sich V so in zwei Teilmengen U und W partitionieren läßt, dass alle Kanten zwischen U und W verlaufen.

Einen Spezialfall bildet der ***vollständig bipartite*** Graph $K_{u,w} = (V, E)$ mit $V = [u + w]$ und $E = \{\{i, j\} \mid 1 \leq i \leq u < j \leq u + w\}$, der alle Kanten zwischen $U = \{1, \ldots, u\}$ und $W = \{u + 1, \ldots, u + v\}$ enthält.

Satz 27 *Ein Graph ist genau dann bipartit, wenn er keine Kreise ungerader Länge besitzt.*

(Beweis siehe Übungen.)

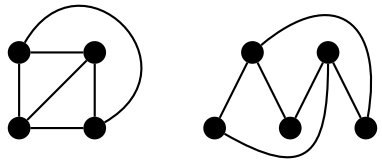
2.5 Planare Graphen

Ein planarer Graph kann so in der Ebene gezeichnet werden, dass sich keine Kanten überschneiden.

Definition 28 (***planar***)

> Ein Graph G heißt ***planar***, wenn er so in die Ebene einbettbar ist, dass sich zwei verschiedene Kanten höchstens in ihren Endpunkten berühren. Dabei werden die Knoten von G als Punkte und die Kanten von G als Verbindungslinien zwischen den zugehörigen Endpunkten dargestellt. Ein ***ebener*** Graph ist ein in die Ebene eingebetteter Graph.

Wie die folgenden Einbettungen von K_4 und $K_{2,3}$ in die Ebene zeigen, sind K_4 und $K_{2,3}$ planar.

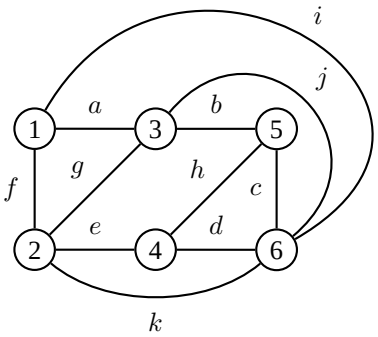


K_4 $K_{2,3}$

Durch die Kanten eines ebenen Graphen wird die Ebene in so genannte ***Gebiete*** unterteilt, von denen genau eines unbeschränkt ist und als ***äußeres*** Gebiet bezeichnet wird. Die Anzahl der Gebiete von G bezeichnen wir mit $r(G)$ oder kurz mit r. Der ***Rand*** eines Gebiets ist die Menge aller Kanten, die an dieses Gebiet grenzen. Bezeichnen wir die Anzahl der an ein Gebiet g grenzenden Kanten mit $d(g)$, wobei von g eingeschlossene Kanten doppelt gezählt werden,so gilt offensichtlich

$$\sum_{g \text{ Gebiet}} d(g) = i(G) = 2m(G).$$

Hierbei bezeichnet $i(G)$ die Anzahl aller Inzidenzen von Gebieten und Kanten, wobei jede Kante mit 2 Inzidenzen zu Buche schlägt. Ein ebener Graph wird durch das Tripel $G = (V, E, R)$ beschrieben, wobei R aus den Rändern aller Gebiete von G besteht.



$R = \{\{a, f, g\}, \{a, i, j\}, \{b, h, e, g\},$
$\{b, c, j\}, \{c, d, h\}, \{e, d, k\}, \{f, i, k\}\}$

Satz 29 (Euler, 1750) *Sei $G = (V, E, R)$ ein zusammenhängender ebener Graph. Dann gilt*

$$n(G) - m(G) + r(G) = 2. \qquad (*)$$

Beweis: Wir führen den Beweis durch Induktion über die Kantenzahl $m(G) = m$.

$m = 0$: Da G zusammenhängend ist, muss dann $n = 1$ sein. Somit ist auch $r = 1$, also $(*)$ erfüllt.

$m = 1$: In diesem Fall muss $n = 2$ und $r = 1$ sein, weshalb $(*)$ erfüllt ist.

$m - 1 \rightsquigarrow m$: Sei G ein zusammenhängender ebener Graph mit m Kanten.

1. Fall: G ist ein Baum. Nach Entfernen eines Blattes u ist der resultierende Graph $G - u$ immer noch zusammenhängend und eben, besteht jedoch aus $n - 1$ Knoten, $m - 1$ Kanten und r Gebieten. Nach Induktionsvoraussetzung gilt daher $(n - 1) - (m - 1) + r = 2$, so dass $(*)$ erfüllt ist.
2. Fall: G ist kein Baum. Sei $K = u_0, u_1, \ldots, u_l$ ein Kreis in G. Nach Entfernen der Kante $\{u_0, u_1\}$ entsteht ein zusammenhängender ebener Graph mit $m-1$ Kanten, n Knoten und $r - 1$ Gebieten. Nach Induktionsvoraussetzung gilt daher $n - (m - 1) + (r - 1) = 2$, so dass $(*)$ auch in diesem Fall erfüllt ist.

■

Korollar 30 *Sei $G = (V, E)$ ein planarer Graph mit $n \geq 3$ Knoten. Dann gilt $m \leq 3n - 6$.*

Beweis: Wir betrachten eine beliebige planare Einbettung von G, wobei wir o.B.d.A. annehmen, dass G zusammenhängend ist. Da $n \geq 3$ ist, steuert jedes Gebiet g $d(g) \geq 3$ Kanten-Inzidenzen zur Gesamtzahl $i = \sum_g d(g)$ bei. Daher ist $2m = i = \sum_g d(g) \geq 3r$ bzw. $r \leq 2m/3$ und es ergibt sich mit Eulers Formel

$$m = n + r - 2 \leq n + 2m/3 - 2,$$

was $m \leq 3n - 6$ impliziert. ■

Korollar 31 *K_5 ist nicht planar.*

Beweis: Wegen $n = 5$, also $3n - 6 = 9$ und wegen $m = \binom{5}{2} = 10$ gilt $m \not\leq 3n - 6$. ■

Korollar 32 *Sei $G = (V, E)$ ein dreiecksfreier, planarer Graph mit $n \geq 3$ Knoten. Dann gilt $m \leq 2n - 4$.*

Beweis: Wir betrachten eine beliebige planare Einbettung von G, wobei wir wieder o.B.d.A. annehmen können, dass G zusammenhängend ist. Wie im vorigen Beweis gilt $i = 2m$, wobei i die Anzahl der Inzidenzen von Kanten und Gebieten ist. Da $n \geq 3$ und G zusammenhängend und dreiecksfrei ist, steuert andererseits jedes Gebiet mindestens 4 Inzidenzen bei. Daher gilt $2m = i \leq 4r$ bzw. $r \leq m/2$ und es ergibt sich mit Eulers Formel

$$m = n + r - 2 \leq n + m/2 - 2,$$

was $m \leq 2n - 4$ impliziert. ■

Korollar 33 *$K_{3,3}$ ist nicht planar.*

Beweis: Wegen $n = 6$, also $2n - 4 = 8$ und wegen $m = 3 \cdot 3 = 9$ gilt $m \not\leq 2n - 4$. ■

2.6 Färbung von Graphen und das 4-Farben-Problem

Definition 34 (***chromatische Zahl***)

Sei $G = (V, E)$ ein Graph und sei $k \in \mathbf{N}$. Eine k-***Färbung*** von G ist eine Abbildung $c : V \to \{1, \ldots, k\}$ mit $c(u) \neq c(v)$ für alle $\{u, v\} \in E$. Falls eine solche Abbildung existiert, heißt G k-***färbbar***.

Die ***chromatische Zahl*** von G ist

$$\chi(G) = \min\{k \geq 1 \mid G \text{ ist } k\text{-färbbar}\}.$$

Beispiel 35

$$\chi(E_n) = 1,\ \chi(K_{n,m}) = 2,\ \chi(K_n) = n,$$

$$\chi(C_n) = \begin{cases} 2, & n \textit{ gerade} \\ 3, & \textit{sonst.} \end{cases}$$

Lemma 36 *(i)* $\chi(G) \geq \omega(G)$,

(ii) $\chi(G) \geq n/\alpha(G)$,

(iii) $\chi(G) \leq \Delta(G) + 1$.

Beweis:

(i) ist klar.

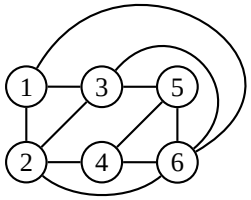
(ii) Sei G ein Graph und sei c eine $\chi(G)$-Färbung von G. Da dann die Mengen $S_i = \{u \in V \mid c(u) = i\}$, $i = 1, \ldots, \chi(G)$, stabil sind, folgt $\|S_i\| \leq \alpha(G)$ und somit gilt

$$n \ = \ \sum_{i=1}^{\chi(G)} \|S_i\| \ \leq \ \chi(G)\alpha(G).$$

(iii) Betrachte folgenden Färbungsalgorithmus:

1 **Eingabe:** $G = (V, E)$ mit $V = \{v_1, \ldots, v_n\}$
2 $c(v_1) \leftarrow 1$

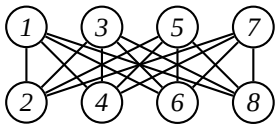
3 **for** $i \leftarrow 2$ **to** n **do**
4 $F \leftarrow \{c(v_j) \mid j < i, v_j \in \Gamma(v_i)\}$
5 $c(v_i) \leftarrow \min\{k \geq 1 \mid k \notin F\}$
6 **end**
7 **Ausgabe:** c



i	1	2	3	4	5	6
$c(v_i)$	1	2	3	1	2	4

Da zur Färbung von v_i höchstens $\|F\| \leq \Delta(G)$ Farben verboten sind, benutzt der Algorithmus nur Farben $c(v_i) \leq \Delta(G) + 1$. ■

Beispiel 37 *Obiger Algorithmus benötigt zur Färbung von $K_{n,n}$ nur 2 Farben.*



i	1	2	3	4	5	6	7	8
$c(v_i)$	1	2	1	2	1	2	1	2

Als nächstes betrachten wir folgendes Entscheidungsproblem.

k-FÄRBUNG:

Gegeben: Ein Graph G.

Gefragt: Ist G k-färbbar?

Satz 38 *3-FÄRBUNG ist $\mathcal{NP}$-vollständig.*

Beweis: Wir zeigen 3-SAT $\leq_p$ 3-FÄRBUNG. Sei

$$F = \bigwedge_{i=1}^{m} \underbrace{\bigvee_{j=1}^{3} l_{ij}}_{C_i}$$

mit $l_{ij} \in \{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$ eine 3-KNF-Formel, wobei Mehrfachvorkommen von Literalen in Klauseln zulässig sind. Gesucht ist ein Graph G_F, so dass G_F genau dann 3-färbbar ist, wenn F erfüllbar ist.

Wir bauen G_F aus $m+1$ Teilgraphen $H_0, H_1, \ldots, H_m$ zusammen. Dabei ist H_0 der Graph (V_0, E_0) mit

$$\begin{aligned} V_0 &= \{u, x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\} \\ E_0 &= \{\{u, x_1\}, \ldots, \{u, x_n\}, \{u, \bar{x}_1\}, \ldots, \{u, \bar{x}_n\}, \{x_1, \bar{x}_1\}, \ldots, \{x_n, \bar{x}_n\}\} \end{aligned}$$

und für $i = 1, \ldots, m$ ist H_i der Graph (V_i, E_i) mit

$$\begin{aligned} V_i &= \{v, a_i, b_i, c_i, y_i, z_i\} \\ E_i &= \{\{v, y_i\}, \{v, z_i\}, \{a_i, y_i\}, \{a_i, z_i\}, \{b_i, y_i\}, \{c_i, z_i\}, \{b_i, c_i\}\} \end{aligned}$$

Nun bilden wir $G_F = (V, E)$, indem wir die Teilgraphen $H_0, H_1, \ldots, H_m$ vereinigen und die Kante $\{u, v\}$ sowie für jede Klausel C_i die Kanten $\{l_{i1}, a_i\}$, $\{l_{i2}, b_i\}$ und $\{l_{i3}, c_i\}$ hinzufügen:

$$\begin{aligned} V &= V_0 \cup V_1 \cup \cdots \cup V_m \\ E &= E_0 \cup E_1 \cup \cdots \cup E_m \cup \{\{u, v\}\} \cup \bigcup_{i=1}^{m} \{\{l_{i1}, a_i\}, \{l_{i2}, b_i\}, \{l_{i3}, c_i\}\} \end{aligned}$$

Die folgende Behauptung ist eine unmittelbare Konsequenz aus der Konstruktion von H_0.

Behauptung 1 *Für jede 3-Färbung c von H_0 mit $c(u) = 2$ gilt $c(x_i) + c(\bar{x}_i) = 1$.*

Auch die folgende Behauptung ist leicht zu beweisen.

Behauptung 2 *Für jede 3-Färbung c von H_i mit $c(v) = 0$ gilt $0 \in \{c(a_i), c(b_i), c(c_i)\}$.*

Ist nämlich c eine 3-Färbung von H_i mit $c(v) = 0$ und $0 \notin \{c(a_i), c(b_i), c(c_i)\}$, so können wir o.B.d.A. annehmen, dass $c(b_i) = 1$ und $c(c_i) = 2$ ist. Dies führt jedoch auf einen Widerspruch, da dann $c(y_i) = 2$ und $c(z_i) = 1$, also $c(a_i) = 0$ gelten muss.

Schließlich benötigen wir noch folgende Behauptung.

Behauptung 3 *Seien $a', b', c' \in \{0, 1\}$ mit $a'b'c' \neq 000$. Dann gibt es eine 3-Färbung c von H_i mit*

- *$c(v) = 0$ und*
- *$c(a_i) \neq a'$, $c(b_i) \neq b'$, $c(c_i) \neq c'$.*

Für den Beweis von Behauptung 3 betrachten wir die drei Fälle a) $b' = 1$, b) $b' = 0$, $c' = 1$, sowie c) $b' = c' = 0$, $a' = 1$. Im Fall a) leistet die 3-Färbung

v	a_i	b_i	c_i	y_i	z_i
0	2	0	2	1	1

das Gewünschte und im Fall c) leistet dies die 3-Färbung

v	a_i	b_i	c_i	y_i	z_i
0	0	2	1	1	2

Der Fall b) ist symmetrisch zum Fall a).

Mit Hilfe dieser drei Behauptungen fällt es nun leicht, die Äquivalenz

$$F \in \text{3-SAT} \Longleftrightarrow G_F \text{ ist 3-färbbar}$$

nachzuweisen.

„⇒“: Sei $a = a_1 \cdots a_n$ eine erfüllende Belegung für F. Dann kann die Funktion

$$c : \{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n, u, v\} \to \{0, 1, 2\}$$

mit

$$c(u) = 2, c(v) = 0, c(x_i) = a_i, c(\bar{x}_i) = 1 - a_i$$

nach Behauptung 3 zu einer 3-Färbung c' von G_F erweitert werden, da jede Klausel C_i ein Literal l_{ij} mit der Farbe $c(l_{ij}) = 1$ enthält.

„⇐“: Ist c eine 3-Färbung von G_F, dann muss $c(u) \neq c(v)$ gelten. Daher existiert eine Permutation π auf $\{0, 1, 2\}$ mit $\pi(c(u)) = 2$ und $\pi(c(v)) = 0$. Folglich ist $c'(w) := \pi(c(w))$, $w \in V$, eine 3-Färbung von G_F mit $c'(u) = 2$ und $c'(v) = 0$. Wegen Behauptung 1 ist dann $a = a_1 \cdots a_n$ mit $a_i := c(x_i)$ eine Belegung, die auf Grund von Behauptung 2 die Formel F erfüllt, da andernfalls $c'(l_{i1}) = c'(l_{i2}) = c'(l_{i3}) = 0$ für ein i und somit $= \notin \{c'(a_i), c'(b_i), c'(c_i)\}$ gelten müsste.

■

4-Farben-Vermutung: Jede Landkarte kann mit 4 Farben gefärbt werden, so dass aneinander grenzende Länder unterschiedliche Farben erhalten. Offensichtlich lässt sich jede Landkarte in einen planaren Graphen transformieren, indem man für jedes Land einen Knoten zeichnet und benachbarte Länder durch eine Kante verbindet. (Länder, die sich nur in einem Punkt berühren, werden nicht als benachbart betrachtet.)

Diese Vermutung wurde 1878 von Kempe „bewiesen“. Erst 1890 entdeckte Heawood den Fehler, welcher im „Beweis“ von Kempe enthalten war. Dabei entstand dann der 5-Farben-Satz. Dass die 4-Farben-Vermutung tatsächlich stimmt, wurde erst 1976 von Appel und Haken bewiesen. Hierbei handelt es sich jedoch nicht um einen Beweis im klassischen Sinne, da zur Überprüfung der auftretenden Spezialfälle der Einsatz von Computern benötigt wird.

Lemma 39 *Sei G ein planarer Graph. Dann ist $\delta(G) \leq 5$.*

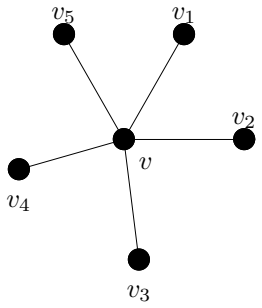
Beweis: Für $\|V\| \leq 6$ ist die Behauptung klar. Für $\|V\| > 6$ würde aus der Annahme $\delta(G) \geq 6$ sofort $\|E\| = \frac{1}{2}\sum_{u \in V} d(u) \geq \frac{1}{2}\sum_{u \in V} 6 = 3\|V\|$ folgen, was aber der Ungleichung $\|E\| \leq 3 \cdot \|V\| - 6$, die für planare Graphen mit mindestens drei Knoten gilt, widerspräche. ■

Satz 40 (Kempe, Heawood) *Jeder planare Graph $G = (V, E)$ ist 5-färbbar.*

Beweis: Durch Induktion über $n = \|V\|$.

$n \leq 5$: Klar.

$n > 5$: Nach vorigem Lemma existiert ein Knoten v mit $d(v) \leq 5$. Nach Induktionsvoraussetzung existiert eine 5-Färbung c von $G - v$. Ist $\Gamma(v)$ durch c mit höchstens vier Farben gefärbt, so können wir c auf v fortsetzen. Andernfalls betrachten wir eine feste kreuzungsfreie Einbettung von G in die Ebene. Darin seien die Nachbarn $v_1, v_2, \ldots, v_5$ von v in dieser Reihenfolge im Uhrzeigersinn um v angeordnet und es gelte o. B. d. A. $c(v_i) = i$.



1. Beweismöglichkeit:

Für Farben i und j bezeichne $G_{i,j}$ den Teilgraphen $G[\{u \in V \mid c(u) \in \{i, j\}\}]$ von G, der durch alle mit i oder j gefärbten Knoten induziert wird.

1. Fall: In $G_{1,3}$ existiert kein Weg von v_1 nach v_3.

   Dann lässt sich c zu einer Färbung c' mit $c'(v_1) = 3$ modifizieren, indem in der Zusammenhangskomponente von $G_{1,3}$, in welcher sich v_1 befindet, die Farben 1 und 3 vertauscht werden.

2. Fall: In $G_{1,3}$ existiert ein Weg von v_1 nach v_3.

   Dann kann es in $G_{2,4}$ keinen Weg von v_2 nach v_4 geben, d. h. c lässt sich zu einer Färbung c' mit $c'(v_2) = 4$ modifizieren.

In beiden Fällen benutzt c' für $\Gamma(v)$ nur vier Farben, so dass für u eine Farbe übrig bleibt.

2. Beweismöglichkeit:

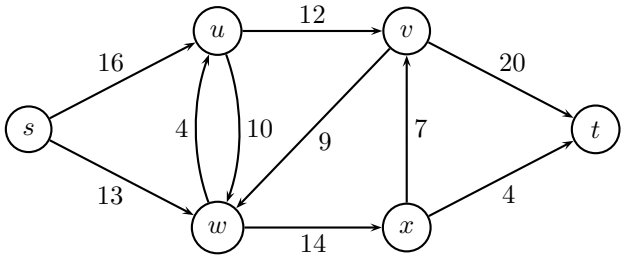
Da G planar ist, ist in G kein K_5 enthalten, d. h. es existieren $i \neq j \in \{1, \ldots, 5\}$ mit $\{v_i, v_j\} \notin E$. Nach Induktionsvoraussetzung existiert für den Graph G', der aus $G - v$ durch Identifikation von v_i mit v_j entsteht, eine 5-Färbung c. Trennen wir nun v_i wieder von v_j, so ist c immer noch eine 5-Färbung mit $c(v_i) = c(v_j)$, d. h. v kann mit der verbliebenen Farbe gefärbt werden. ■

2.7 Flüsse in Netzwerken

Definition 41 (***Netzwerk***)

Ein ***Netzwerk*** $N = (V, E, s, t, c)$ besteht aus einem gerichteten Graphen $G = (V, E)$ mit zwei ausgezeichneten Knoten $s, t \in V$, der ***Quelle*** s und der ***Senke*** t, sowie einer ***Kapazitätsfunktion*** $c : V \times V \to \mathbb{N}$ mit $c(u, v) = 0$ für alle $(u, v) \notin E$.

Die folgende Abbildung zeigt ein Netzwerk N.



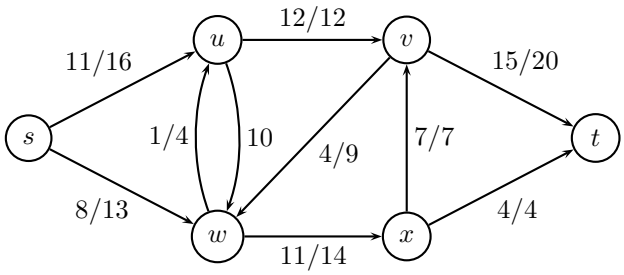
Definition 42 (***Fluss***)

Ein ***Fluss*** für N ist eine Funktion $f : V \times V \to \mathbb{Z}$ mit

- $f(u, v) \leq c(u, v)$, „Kapazitätsbedingung“
- $f(u, v) = -f(v, u)$, „Symmetriebedingung“
- Für alle $u \in V - \{s, t\} : \sum_{v \in V} f(u, v) = 0$. „Kirchhoffsches Gesetz“

Der ***Größe*** von f ist $|f| = \sum_{v \in V} f(s, v)$.

Die folgende Abbildung zeigt einen Fluss f für das Netzwerk N.



Problem: Wie lässt sich ein Fluss maximaler Größe bestimmen?

Definition 43 (***Restnetzwerk***)

Sei $N = (V, E, s, t, c)$ ein Netzwerk und sei f ein Fluss für N. Das zugeordnete ***Restnetzwerk*** ist $N_f = (V, E_f, s, t, c_f)$, wobei

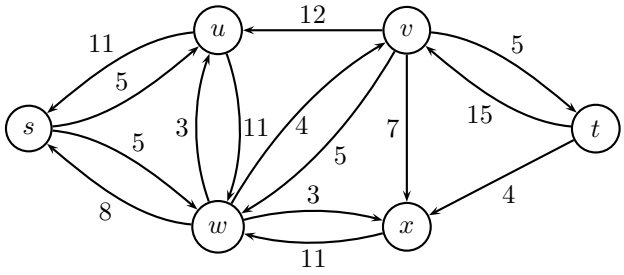
$$c_f(u, v) = c(u, v) - f(u, v)$$

und

$$E_f = \{(u, v) \in V \times V \mid c_f(u, v) > 0\}$$

ist.

Die folgende Abbildung zeigt das Restnetzwerk N_f, das zum Netzwerk N und zum Fluss f gehört.



Definition 44 (***Erweiterungspfad***)

Sei $N_f = (V, E_f, s, t, c_f)$ ein Restnetzwerk. Dann ist $P = u_0, \ldots, u_k$ ein ***Erweiterungspfad*** für N_f, falls P ein Pfad von s nach t im Graphen (V, E_f) ist, d.h. es gilt

- $u_0 = s$,
- $u_k = t$,
- $c_f(u_i, u_{i+1}) > 0$ für $i = 0, \ldots, k - 1$.

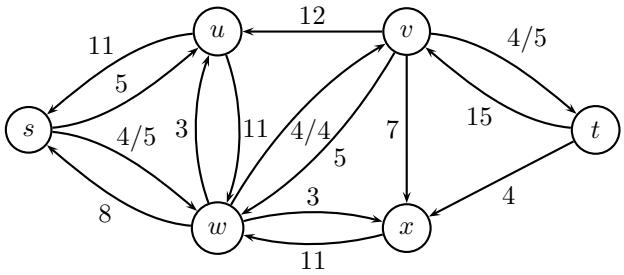
Die ***Restkapazität*** von P in N_f ist

$$c_f(P) = \min\{c_f(u, v) \mid (u, v) \text{ liegt auf } P\}$$

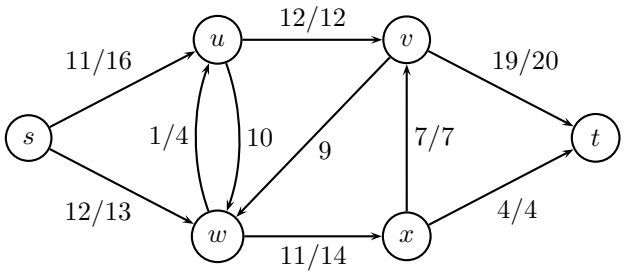
und der zu P gehörige Fluss in N_f ist

$$g_P(u, v) = \begin{cases} c_f(P), & (u, v) \text{ liegt auf } P, \\ -c_f(P), & (v, u) \text{ liegt auf } P, \\ 0, & sonst. \end{cases}$$

Die folgende Abbildung zeigt den zum Erweiterungspfad $P = s, w, v, t$ gehörigen Fluss g_P in N_f. Die Restkapazität von P ist $c_f(P) = 4$.



Es ist leicht zu sehen, dass g_P tatsächlich ein Fluss für N_f ist. Durch Addition der beiden Flüsse f und g_P erhalten wir einen größeren Fluss f'.



Lemma 45 *Sei $N = (V, E, s, t, c)$ ein Netzwerk, f ein Fluss für N, sei P ein Erweiterungspfad für das Restnetzwerk N_f und sei g_P der zu P gehörige Fluss in N_f. Dann ist $f' = f + g_P$ ein Fluss für N mit $|f'| = |f| + |g_P|$.*

Nun können wir den **Ford-Fulkerson-Algorithmus** angeben:

```
1  Eingabe: Netzwerk N = (V, E, s, t, c)
2    setze f(u, v) = 0 für alle (u, v) ∈ V × V
3    while es gibt einen Erweiterungspfad P für N_f do
4      f ← f + g_P
5    end
6  Ausgabe: f
```

Frage: Berechnet der Algorithmus von Ford-Fulkerson tatsächlich einen optimalen Fluss?

Um diese Frage mit ja beantworten zu können, müssen wir zeigen, dass f ein optimaler Fluss ist, falls es im Restnetzwerk N_f keinen Erweiterunspfad mehr gibt.

Definition 46 (***Schnitt***)

Sei $N = (V, E, s, t, c)$ ein Netzwerk. Ein ***Schnitt*** durch N ist eine Partition (S, T) von V mit $s \in S$ und $t \in T$. Die ***Kapazität*** eines Schnittes (S, T) ist

$$c(S, T) = \sum_{u \in S, v \in T} c(u, v)$$

Ist f ein Fluss für N, so heißt

$$f(S,T) = \sum_{u \in S, v \in T} f(u,v)$$

der ***Fluss über den Schnitt*** (S,T).

Lemma 47 *Für jeden Schnitt (S,T) und jeden Fluss f gilt:*

1. $f(S,T) \leq c(S,T)$
2. $f(S,T) = |f|$

Beweis: zu 1:

$$f(S,T) = \sum_{u \in S, v \in T} \underbrace{f(u,v)}_{\leq c(u,v)} \leq \sum_{u \in S, v \in T} c(u,v) = c(S,T)$$

zu 2: Wir führen einen Induktionsbeweis über $\|S\| = k$ durch.

Induktionsanfang: Für $\|S\| = 1$ ist die Behauptung klar.

Induktionsschritt: Sei (S,T) ein Schnitt mit $\|S\| = k+1$. Wähle ein $v \in S - \{s\}$ und betrachte den Schnitt $(S',T') := (S - \{v\}, T \cup \{v\})$. Nach Induktionsvoraussetzung ist $f(S',T') = |f|$. Somit gilt also:

$$f(S,T) = f(S',T') - \sum_{u \in S} f(u,v) + \sum_{u \in T} f(v,u) = |f| + \sum_{u \in V} f(v,u) = |f|$$

■

Satz 48 (Min-Cut-Max-Flow-Theorem) *Sei f ein Fluss für $N = (V,E,s,t,c)$. Dann sind folgende Aussagen äquivalent:*

1. *f ist maximal.*
2. *In N_f existiert kein Erweiterungspfad.*
3. *Es gibt einen Schnitt (S,T) mit $c(S,T) = |f|$.*

Beweis:

„1 $\Rightarrow$ 2“: Würde ein Erweiterungspfad existieren, so könnte f verbessert werden, was ein Widerspruch wäre.

„2 $\Rightarrow$ 3“: Betrachte den Schnitt (A,B) mit

$$\begin{aligned} S &= \{u \in V \mid u \text{ ist in } N_f \text{ von } s \text{ aus erreichbar}\} \\ T &= V \setminus S. \end{aligned}$$

Dann gilt:

- $s \in S$
- $t \in T$, da kein Erweiterungspfad existiert
- $c_f(u,v) = 0$ für alle $u \in S$ und $v \in T$. Wegen $c_f(u,v) = c(u,v) - f(u,v)$ folgt somit

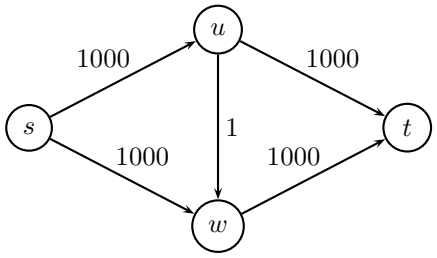
$$|f| = f(S,T) = \sum_{u \in S, v \in T} f(u,v) = \sum_{u \in S, v \in T} c(u,v) = c(S,T).$$

„3 $\Rightarrow$ 1“: Klar, da für jeden Fluss f' gilt:

$$|f'| \leq c(S,T) = |f|.$$

■

Bemerkung 49 *Der Algorithmus von Ford-Fulkerson hat exponentielle Zeitkomplexität:*



Bei diesem Netzwerk benötigt Ford-Fulkerson abhängig von der Wahl des Erweiterungspfads zwischen 2 und 2000 Schleifendurchläufe zur Bestimmung des maximalen Flusses. Dies lässt sich (nicht nur in diesem Beispiel) dadurch vermeiden, dass man immer einen kürzesten Erweiterungspfad bestimmt (solange einer existiert). Diese Vorgehensweise wird als ***Edmonds-Karp-Strategie*** *bezeichnet und führt auf eine polynomiale Laufzeit.*

2.8 Heiratssatz

Ein Matching $M \subseteq E$ in einem Graphen $G = (V, E)$ ordnet jedem in M vorkommenden Knoten einen seiner Nachbarknoten als Partner zu.

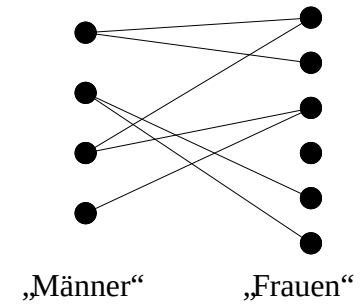
Definition 50 (***Matching***)

Sei $G = (V, E)$ ein Graph. $M \subseteq E$ heißt ***Matching***, falls für alle $e, e' \in M$ gilt:

$$e \neq e' \Rightarrow e \cap e' = \emptyset$$

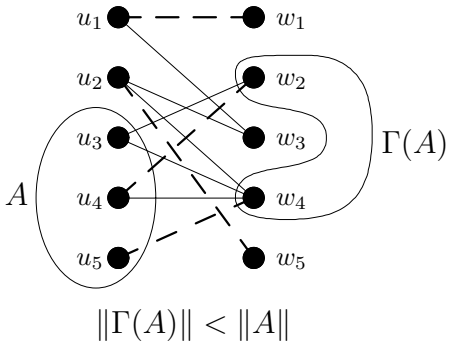
Die Kanten von G beschreiben also die möglichen Partnerbeziehungen und durch ein Matching M wird eine mit diesen Vorgaben kompatible Menge von Partnerschaften realisiert. Das Ziel besteht nun darin, ein optimales (d.h. möglichst großes) Matching M in G zu finden. Häufig ist hierbei G ***bipartit***, d.h. V lässt sich in zwei disjunkte Knotenmengen U und V zerlegen mit $\|e \cap U\| = \|e \cap V\| = 1$ für alle Kanten $e \in E$.

Beispiel 51

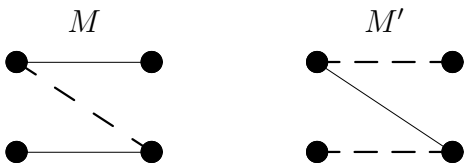


$\{u, w\} \in E$ heißt: u und w sind sich sympathisch.

Beispiel 52 *Der folgende bipartite Graph G enthält ein Matching der Größe 4 (gestrichelt gezeichnet). Wegen $\Gamma(\{u_3, u_4, u_5\}) = \{w_2, w_4\}$ kann es offensichtlich kein größeres Matching in G geben.*



Bemerkung 53 *Ein bezüglich Mengeninklusion maximales Matching muss nicht optimal (also nicht maximal bezüglich der Kardinalität) sein:*



$M = \{\{u_1, w_2\}\}$ ist maximal, da es sich nicht erweitern lässt. M ist jedoch kein optimales Matching, da $M' = \{\{u_1, w_1\}, \{u_2, w_2\}\}$ größer ist. Die Greedy-Methode funktioniert also nicht.

Satz 54 *Für einen bipartiten Graphen G kann ein optimales Matching in Polynomialzeit berechnet werden.*

Beweis: Wir reduzieren das Auffinden eines optimalen Matchings in G auf folgendes Flussproblem:

Sei $G = (V, E)$ mit $V = U \dot\cup W$. Betrachte das Netzwerk $N(G) = (V', E', s, t, c)$ mit

- $V' = V \dot\cup \{s, t\}$,
- $E' = \{(s, u) \mid u \in U\} \cup \{(u, w) \in U \times W \mid \{u, w\} \in E\} \cup \{(w, t) \mid w \in W\}$ und
- $c(e) = 1$ für alle $e \in E'$.

Dann können wir jedem Fluss f in $N(G)$ ein Matching $M_f = \{\{u, w\} \in E \mid f(u, w) = 1\}$ mit $\|M_f\| = |f|$ zuordnen. Umgekehrt entspricht einem Matching M der Fluss

$$f_M(v, v') = \begin{cases} 1, & (v, v') \in E_M \\ -1, & (v', v) \in E_M \\ 0, & \text{sonst,} \end{cases}$$

wobei

$$E_M = \{(s, u) \mid u \in U_M\} \cup \{(u, w) \in U \times W \mid \{u, w\} \in M\} \cup \{(w, t) \mid w \in W_M\},$$

$U_M = \{u \in U \mid \exists w : \{u, w\} \in M\}$ und $W_M = \{w \in W \mid \exists u : \{u, w\} \in M\}$ ist. Da der Ford-Fulkerson Algorithmus einen maximalen Fluss f in $N(G)$ liefert, ist das zughörige Matching M_f optimal. Da die Größe $|f|$ von f durch die Anzahl der Kanten in E beschränkt ist, wird die while-Schleife des Ford-Fulkerson Algorithmus höchstens $\|E\|$-mal durchlaufen, was auf eine polynomielle Laufzeit führt. ■

Satz 55 (Heiratssatz von Hall) *Sei $G = (V, E)$ ein bipartiter Graph mit der Knotenpartition $V = U \dot\cup W$. Dann existiert in G genau dann ein Matching M mit $\|M\| = \|U\|$, wenn für jede Teilmenge $A \subseteq U$ gilt: $\|\Gamma(A)\| \geq \|A\|$, wobei $\Gamma(A) = \bigcup_{u \in A} \Gamma(u)$.*

Beweis:

„⇒“: Klar, da in $\Gamma(A)$ für jedes $u \in A$ der Matchingpartner von u enthalten ist.

„⇐“: Gelte nun $\|\Gamma(A)\| \geq \|A\|$ für alle $A \subseteq U$. Sei (S, T) ein beliebiger Schnitt durch das zu G gehörige Netzwerk $N(G)$. Da (S, T) alle Kanten (s, u) mit $u \in U - S$ und für jeden Knoten $w \in \Gamma(S \cap U)$ entweder die Kante (w, t) (im Fall $w \in S$) oder eine Kante (u, w) mit $u \in S \cap U$ (im Fall $w \in T$) enthält, folgt

$$c(S, T) \geq \|U - S\| + \underbrace{\|\Gamma(S \cap U)\|}_{\geq \|S \cap U\|} \geq \|U - S\| + \|S \cap U\| = \|U\|.$$

Da somit jeder Schnitt mindestens die Kapazität $\|U\|$ hat, muss nach dem Min-Cut-Max-Flow-Theorem ein Fluss f mit $|f| \geq \|U\|$ existieren, welcher ein Matching M_f der gewünschten Größe liefert.

■

Bemerkung 56 *Sei $\beta(G)$ die maximale Größe eines Matchings in einem bipartiten Graphen $G = (V, E)$ mit $V = U \dot{\cup} W$. Dann gilt*

$$\beta(G) = \|U\| - \max_{A \subseteq U} (\|A\| - \|\Gamma(A)\|) .$$

2.9 Kürzeste Wege

Gegeben ist ein gerichteter Graph $G = (V, E)$ mit einem ***Startknoten*** $s \in V$ und einem ***Zielknoten*** $t \in V$, sowie eine Längenfunktion $l : E \to \mathbb{N}$. Die Aufgabe, einen möglichst kurzen Weg von s nach t zu finden, wird durch folgenden nach Dijkstra benannten Algorithmus in Polynomialzeit gelöst.

Eingabe: gerichteter Graph $G = (V, E)$ mit s, t, l wie oben
  $g(s) \leftarrow 0$
  for all $u \in V - \{s\}$ **do**
    $g(u) \leftarrow \infty$
  end
  $T \leftarrow V$
  while $t \in T$ **do**
    Finde $u \in T$ **mit** $g(u)$ **minimal**
    $T \leftarrow T - \{u\}$
    for all $v \in \Gamma(u) \cap T$ **do**
      if $g(u) + l(u, v) < g(v)$ **then**
        $g(v) \leftarrow g(u) + l(u, v)$
        $\alpha(v) \leftarrow u$
      end
    end
  end
Ausgabe: Weg $u_0, \ldots, u_k$ mit $u_k = t$, $\alpha(u_i) = u_{i-1}$ für $i = 1, \ldots, k$
  und $u_0 = s$ (natürlich nur, falls $g(t) < \infty$ ist)

Natürlich können wir mit dem **Dijkstra-Algorithmus** auch kürzeste Wege zu allen Knoten $u \in V$ bestimmen, falls wir die while-Bedingung $t \in T$ durch $T \neq \emptyset$ ersetzen.

Für einen Knoten $u \in V$ sei $\sigma(u)$ die Länge eines kürzesten Weges von s nach u. Falls in G kein Weg von s nach t existiert, sei $\sigma(u) = \infty$. Das folgende Lemma beweist die Korrektheit des Dijkstra-Algorithmus'.

Lemma 57 *Für jeden Knoten u, der in Schritt 9 ausgewählt wird, gilt $g(u) = \sigma(u)$. Im Fall $g(u) < \infty$ ist zudem $u_0, \ldots, u_k$ mit $u_k = u$, $u_{i-1} = \alpha(u_i)$ für $i = 1, \ldots, k$ und $u_0 = s$ ein Weg von s nach u der Länge $g(u)$.*

Beweis: Wir beweisen die Aussage des Lemmas durch Induktion über die Reihenfolge, in der die Knoten aus T entfernt werden.

Induktionsanfang: Zuerst wird der Startknoten s aus T entfernt. Wegen $g(u) = 0 = \sigma(u)$ trifft die Aussage auf s zu.

Induktionsschritt: Sei u ein Knoten, der aus T entfernt wird und gelte die Aussage für alle Knoten u', die vor u aus T entfernt wurden.

1. Fall: $g(u) = \infty$. Sei u' der erste Knoten, der mit dem Wert $g(u') = \infty$ aus T gewählt wird. Dann gilt zum Zeitpunkt der Wahl von u' $g(v) = \infty$ für alle $v \in T$ und $g(v) < \infty$ für alle $v \in V - T$. Da zu diesem Zeitpunkt alle Knoten in $\Gamma(V - T)$ bereits einen endlichen Wert $g(v)$ erhalten haben, kann keine Kante von einem Knoten in $V - T$ zu einem Knoten in T existieren. Daher kann $u \in T$ nicht von $s \in V - T$ aus erreichbar sein.

2. Fall: $g(u) < \infty$. Wir zeigen zuerst $g(u) \geq \sigma(u)$. Sei u' der Knoten, bei dessen Wahl aus T $g(u)$ auf $g(u') + l(u', u)$ gesetzt wurde. Nach Induktionsvoraussetzung liefert α einen Weg von s nach u' der Länge $g(u')$, welcher sich durch Hinzufügen der Kante (u', u) zu einem Weg von s nach u der Länge $g(u)$ erweitern lässt.
   Es ist also nur noch $g(u) \leq \sigma(u)$ zu zeigen. Sei $v_0, \ldots, v_k$ ein kürzester Weg von $s = v_0$ nach $u = v_k$. Dann ist

$$\sigma(v_{i+1}) = \sum_{j=0}^{i} l(v_j, v_{j+1})$$

   und daher $\sigma(v_i) \leq \sigma(v_{i+1})$ für $i = 0, \ldots, k-1$. Weiterhin sei v_l der letzte Knoten auf diesem Weg, der vor u aus T entfernt wird. Da der Knoten v_{l+1} nicht vor u aus T entfernt wird, folgt $g(u) \leq g(v_{l+1})$. Da $v_{l+1} \in \Gamma(v_l) \cap T$, wenn v_l aus T entfernt wird, ist $g(v_{l+1}) \leq g(v_l) + l(v_l, v_{l+1})$. Schließlich gilt nach Induktionsvoraussetzung $g(v_l) = \sigma(v_l)$ und daher folgt

$$g(u) \leq g(v_{l+1}) \leq \underbrace{g(v_l)}_{\sigma(v_l)} + l(v_l, v_{l+1}) = \sigma(v_{l+1}) \leq \sigma(u).$$

■

2.10 Die Greedy-Methode und Matroide

Definition 58 (***Matroid***)

Sei E eine endliche Menge und sei $U \subseteq \mathcal{P}(E)$ ein System von Teilmengen von E. Dann heißt (E, U) ***Matroid***, falls gilt:

- $\emptyset \in U$,
- $A \subseteq B, B \in U \Rightarrow A \in U$ (U ist unter Teilmengenbildung abgeschlossen) und
- $A, B \in U, \|A\| < \|B\| \Rightarrow \exists x \in B - A : A \cup \{x\} \in U$. (Austauscheigenschaft)

Bemerkung 59 *Bezeichne U_{max} das System aller bzgl. $\subseteq$ maximalen Mengen in U. Aufgrund der Austauscheigenschaft müssen dann alle Mengen $A \in U_{max}$ dieselbe Mächtigkeit haben. Man beachte, dass dies zwar eine notwendige, aber keine hinreichende Bedingung für das Vorliegen der Austauscheigenschaft ist, wie das nächste Beispiel zeigt.*

Beispiel 60

- *Sei $E = \{1, \ldots, 4\}$. Dann bildet zwar E zusammen mit*

  $$U = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}\}$$

  ein Matroid, nicht jedoch

  $$U' = \{\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{3, 4\}\},$$

  obwohl alle Mengen in $U'_{max} = \{\{1, 2\}, \{3, 4\}\}$ dieselbe Mächtigkeit haben.

- *Sei $E = \{1, \ldots, m\}$. Dann bildet für $k \in \{0, \ldots, m\}$ das Paar (E, U_k) mit*

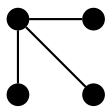
  $$U_k = \{A \subseteq E \mid \|A\| \leq k\}$$

  ein Matroid.

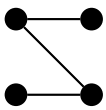
- *Sei $G = (V, E)$ ein Graph und sei*

  $$M(G) = \{E' \subseteq E \mid E' \text{ ist ein Matching in } G\}.$$

  Betrachte die beiden Graphen $G = (V, E)$ und $G' = (V, E')$.



G



G'

Dann ist $(E, M(G))$ zwar ein Matroid, nicht jedoch $(E', M(G'))$.

- *Sei $G = (V, E)$ ein zusammenhängender Graph. Dann ist (E, U) mit*

$$U = \{E' \subseteq E \mid (V, E') \text{ ist zyklenfrei } \}$$

ein Matroid. Die ersten beiden Eigenschaften sind leicht zu sehen. Zum Nachweis der Austauscheigenschaft seien A und B zklenfreie Kantenmengen mit $\|A\| < \|B\|$. Dann zerlegt die Kantenmenge A die Knotenmenge V in $k \geq 1$ Zusammenhangskomponenten $V_1, \ldots, V_k$. Jede Kante in B verbindet nun entweder zwei Knoten aus derselben Komponente V_i oder zwei Knoten aus verschiedenen Komponenten V_i und V_j. Da B höchstens $\sum_{i=1}^{k}(\|V_i\| - 1) = \|A\|$ Kanten des ersten Typs enthalten kann, muss B midestens eine Kante e des 2. Typs enthalten. Dann ist aber $A \cup \{e\}$ zklenfrei und damit in U.

*Für $E' \subseteq E$ heißt $G' = (V, E')$ **aufspannender Baum** von G, falls G' ein Baum ist. Es ist klar, dass $G' = (V, E')$ genau dann ein aufspannender Baum von G ist, wenn $E' \in U_{max}$ ist.*

Sei nun $w : E \to \mathbb{Z}$ eine beliebige Gewichtsfunktion auf E. Gesucht ist eine Menge A in U_{max} mit maximalem (bzw. minimalem) Gewicht.

$$w(A) = \sum_{x \in A} w(x).$$

Die Greedy-Strategie versucht diese Aufgabe dadurch zu lösen, dass sie ausgehend von der leeren Menge jeweils das Element mit dem größtmöglichen Gewicht zu A hinzufügt.

```
1  Eingabe: (E, U, w) wie oben beschrieben
2    T ← ∅
3    for i ← 1 to ‖E‖ do
4      Bestimme ein e_i in E − {e_1, ..., e_{i−1}} mit maximalem Gewicht
5      if T ∪ {e_i} ∈ U then
6        T ← T ∪ {e_i}
7      end
8    end
9  Ausgabe: T
```

Wie der folgende Satz zeigt, ist der Greedy-Algorithmus tatsächlich erfolgreich, falls die zugrunde liegende Struktur ein Matroid ist.

Satz 61 *Falls (E, U) ein Matroid ist, gibt der Greedy-Algorithmus eine Menge T in U_{max} mit maximalem Gewicht aus.*

Beweis: Sei $E = \{e_1, \ldots, e_n\}$ mit $w(e_1) \geq \cdots \geq w(e_n)$ und sei $T = \{e_{i_1}, \ldots, e_{i_k}\}$ die vom Greedy-Algorithmus berechnete Lösung mit $i_1 < \cdots < i_k$.

Wir zeigen zunächst $T \in U_{max}$. Würde ein $T' \in U$ mit $T \subseteq T'$ und $\|T'\| > \|T\|$ existieren, so müsste T' ein Element e_i mit $i \notin \{i_1, \ldots, i_k\}$ enthalten. Sei i der kleinste solche Index und sei j der größte Index mit $i_j < i$. Dann hätte aber e_i wegen $\{e_{i_1}, \ldots, e_{i_j}, e_i\} \subseteq T' \in U$ zu T hinzugefügt werden müssen.

Wir müssen noch zeigen, dass $w(T)$ maximal ist. Würde ein $T' = \{e_{j_1}, \ldots, e_{j_k}\} \in U_{max}$ mit $j_1 > \cdots > j_k$ und $w(T') > w(T)$ existieren, so müsste T' ein Element e_{j_l} mit $w(e_{j_l}) > w(e_{i_l})$ (und somit $j_l < i_l$) enthalten. Sei l der kleinste solche Index. Wenden wir nun die Austauscheigenschaft auf die beiden Mengen

$$A = \{e_{i_1}, \ldots, e_{i_{l-1}}\} \text{ und } B = \{e_{j_1}, \ldots, e_{j_l}\}$$

an, so muss $B - A$ ein Element e_{j_k} mit $A \cup \{e_{j_k}\} \in U$ enthalten. Wegen

$$w(e_{j_k}) \geq w(e_{j_l}) > w(e_{i_l})$$

hätte dann aber das Element e_{j_k} vor e_{i_l} zu T hinzugefügt werden müssen. ■

Wie der vorige Satz zeigt, stellt die Austauscheigenschaft eine hinreichende Bedingung für das Auffinden einer optimalen Lösung durch den Greedy-Algorithmus dar. Tatsächlich ist sie auch notwendig: Ist die Austauscheigenschaft für zwei Mengen A und B in U nicht erfüllt (es gilt also $\|A\| < \|B\|$ und $A \cup \{x\} \notin U$ für alle $x \in B - A$), so berechnet der Greedy-Algorithmus für die Gewichtsfunktion

$$w(x) = \begin{cases} \|B\| + 1, & x \in A, \\ \|B\|, & x \in B - A, \\ 0, & \text{sonst,} \end{cases}$$

eine Menge T mit dem Gewicht $w(T) = (\|B\| + 1)\|A\| \leq \|B\|^2 - 1$, obwohl jede Obermenge T' von B in U_{max} ein Gewicht $w(T') \geq w(B) = \|B\|^2$ besitzt.

Beispiel 62 *Wie wir gesehen haben, enthält U_{max} alle aufspannenden Bäume von $G = (V, E)$, falls wir das Matroid (E, U) mit*

$$U = \{E' \subseteq E \mid (V, E') \textit{ ist zyklenfrei } \}$$

zugrundelegen. Ist daher $w : E \to \mathbb{Z}$ eine beliebige Gewichtsfunktion auf E, so berechnet der Greedy-Algorithmus einen aufspannenden Baum mit maximalem (bzw. minimalem) Gesamtgewicht. Die Variante, bei der das Gesamtgewicht minimiert werden soll, ist auch als Kruskal-Algorithmus bekannt:

```
1  Eingabe: (V, E, w) wie oben beschrieben
2    T ← ∅
3    for i ← 1 to ‖E‖ do
4      Bestimme eine Kante e_i in E − {e_1, ..., e_{i−1}} mit minimalem Gewicht
5      if T ∪ {e_i} ist zyklenfrei then
6        T ← T ∪ {e_i}
```

```
7       end
8     end
9   Ausgabe: T
```

Auch der Dijkstra-Algorithmus zur Bestimmung von kürzesten Wegen von einem Knoten $s \in V$ zu allen Knoten in einem Graphen $G = (V, E)$ mit einer Längenfunktion $l : E \to \mathbb{N}$ verfährt nach der Greedy-Strategie. Das zugrunde liegende Matroid (S, U) wird hierbei allerdings nicht direkt auf der Kantenmenge E von G, sondern auf der Menge S aller von s ausgehenden zklenfreien Pfade in G gebildet: U enthält alle Teilmengen von S, deren Pfade auf verschiedene Endknoten in G führen. Nun ist leicht zu sehen, dass der Dijkstra-Algorithmus in jedem Schleifendurchlauf einen Pfad in S zu einem neuen Knoten bestimmt, dessen Länge $l(S)$ minimal ist. Anders als der kanonische Greedy-Algorithmus durchläuft der Dijkstra-Algorithmus hierzu jedoch nicht alle Pfade in S, wodurch er wesentlich effizienter ist.

2.11 Approximationsalgorithmen

Ein **Optimierungsproblem** O ist durch folgende Komponenten festgelegt.

Eingabemenge D: Menge aller **Problemeingaben**, die vorkommen können;

Lösungsmenge $S(x)$: Jede Eingabe x besitzt eine nichtleere Menge $S(x)$ von **zulässigen Lösungen**;

Auswertungsfunktion $f(y)$: Jeder zulässigen Lösung $y \in S(x)$ ist ein **Wert** $f(y) \in \mathbb{R}^+$ zugeordnet, der optimiert werden soll.

Bei einem **Maximierungsproblem** geht es darum, zu einer gegebenen Eingabe x eine Lösung $y_{\max} \in S(x)$ zu finden, für die f maximal wird, d.h.

$$\forall y \in S(x) : f(y) \leq f(y_{\max})$$

Entsprechend soll bei einem **Minimierungsproblem** eine Lösung $y_{\min}$ mit möglichst kleinem Wert unter f bestimmt werden:

$$\forall y \in S(x) : f(y) \geq f(y_{\min})$$

In beiden Fällen bezeichnen wir das Optimum $f(y_{\max})$ bzw. $f(y_{\min})$ mit $OPT(x)$. Die Klasse PO enthält alle Optimierungsprobleme, für die bei Eingabe x eine optimale Lösung y in Polynomialzeit bestimmt werden kann.

Beispiel 63

- *Wie wir gesehen haben, kann ein maximaler Fluss in einem Netzwerk in Polynomialzeit bestimmt werden, d.h.* MAXFLOW *ist in* PO *lösbar.*
- *Bei den beiden Optimierungsproblemen* MAXCUT *und* MINCUT *ist jeweils*

  $D =$ *Menge aller ungerichteten Graphen* $G = (V, E)$*;*
  $S(G) =$ *Menge aller Schnitte* $(S, V - S)$ *mit* $S \notin \{\emptyset, V\}$*;*
  $f(S, V - S) = \|\{e = \{u, v\} \in E \mid u \in S, v \in V - S\}\|$.

  Wie der Name sagt, soll bei MAXCUT *ein Schnitt maximaler und bei* MINCUT *ein Schnitt minimaler Größe bestimmt werden.*

  Es ist leicht zu sehen, dass MINCUT *in* PO *enthalten ist. Hierzu betrachten wir zu zwei Knoten* $s, t \in V$ *das Netzwerk* $N(s, t) = (V, E', s, t, c)$ *mit* $E' = \{(u, v) \mid \{u, v\} \in E\}$ *und* $c(u, v) = 1$ *für alle Kanten* $(u, v) \in E'$*. Berechnen wir nun für einen festen Knoten* $s \in V$ *den maximalen Fluss* f_t *zu allen Knoten* $t \in V - \{s\}$ *in* $N(s, t)$*, dann liefert jeder Fluss* f_t *einen Schnitt* $(S_t, V - S_t)$ *mit* $f(S_t, V - S_t) = |f_t|$*, unter denen sich der gesuchte Schnitt minimaler Größe befindet. Dagegen ist* MAXCUT *nicht in* PO *enthalten, außer wenn* $\mathcal{P} = \mathcal{NP}$ *ist (ohne Beweis).*
- *Bei* MINNODECOVER (MINNC) *ist*

  $D =$ *Menge aller ungerichteten Graphen* $G = (V, E)$*;*
  $S(G) =$ *Menge aller Knotenüberdeckungen* K *in* G*, d.h.* $K \subseteq V$ *und für jede Kante* $e \in E$ *gilt:* $e \cap K \neq \emptyset$*;*
  $f(K) = \|K\|$.
- *Bei* MAXCLIQUE *ist*

  $D =$ *Menge aller ungerichteten Graphen* $G = (V, E)$*;*
  $S(G) =$ *Menge aller Cliquen* C *in* G*, d.h.* $C \subseteq V$ *und für je zwei Knoten* $u, v \in C$ *gilt* $(u, v) \in E$*;*
  $f(C) = \|C\|$.

Wir werden nur **NP-Optimierungsprobleme** O (kurz $O \in \mathrm{NPO}$) betrachten, d.h.

a) Alle zulässigen Lösungen sind polynomiell in der Größe beschränkt:

$$\exists \text{ Polynom } p \; \forall x \; \forall y \in S(x) : |y| \leq p(|x|)$$

b) Es ist einfach zu entscheiden ob eine Lösung zulässig ist, d.h. die Sprache $\{(x, y) \mid y \in S(x)\}$ ist in Polynomialzeit entscheidbar.

c) Die Funktion f ist in Polynomialzeit berechenbar.

Es ist leicht zu sehen, dass im Fall $\mathcal{P} = \mathcal{NP}$ jedes NP-Optimierungsproblem in Polynomialzeit lösbar ist. Umgekehrt gilt für viele hier betrachteten Optimierungsprobleme dass sie nur dann effizient lösbar sind, wenn $\mathcal{P} = \mathcal{NP}$ gilt. (Solche Probleme werden **NP-hart** genannt.) In der Praxis treten häufig NP-harte Probleme auf, für die es nicht unbedingt erforderlich ist, eine optimale Lösung zu berechnen. Eine Lösung die z.B. um maximal 1% vom Optimum abweicht, wird in vielen Fällen als vollkommen zufriedenstellend betrachtet. Wie wir sehen werden, ist ein Teil der Probleme sehr gut approximativ lösbar, andere Optimierungsprobleme bleiben jedoch auch im approximativen Sinne NP-hart, d.h. für sie kann nicht einmal eine annähernd optimale Lösung in Polynomialzeit gefunden werden, außer wenn $\mathcal{P} = \mathcal{NP}$ ist.

Ein **Approximationsalgorithmus** A für ein Optimierungsproblem ist ein Polynomialzeit-Algorithmus, der für jede Eingabe $x \in D$ eine Ausgabe $y \in S(x)$ erzeugt. Den Wert $f(y)$ der von A bei Eingabe x produzierten Lösung y bezeichnen wir mit $A(x)$.

Beispiel 64 *Betrachte folgenden Greedy-Algorithmus A für das Problem* MINNC*:*

```
1  Eingabe: G = (V, E)
2    K ← ∅
3    while E ≠ ∅ do
4      Finde u ∈ V mit maximalem Grad d(u)
5      K ← K ∪ {u}
6      G ← G − u
7    end
```

Da A für jeden Eingabegraphen G eine Knotenüberdeckung ausgibt, ist A ein Approximationsalgorithmus für MINNC.

Sei A ein Approximationsalgorithmus für ein Optimierungsproblem. Die **Performanz** von A bei Eingabe x ist

$$R_A(x) \quad = \quad \frac{OPT(x)}{A(x)}$$

falls es sich um ein Maximierungsproblem, und

$$R_A(x) \quad = \quad \frac{A(x)}{OPT(x)}$$

falls es sich um ein Minimierungsproblem handelt. In beiden Fällen heißt

$$R_A \quad = \quad \inf\{r \geq 1 \mid \forall x \in D : R_A(x) \leq r\}$$

die **Performanz** von A. Hierbei ist $\inf \emptyset = \infty$.

Ein Optimierungsproblem O heißt **approximierbar** (kurz $O \in$ APX), falls es einen Approximationsalgorithmus A mit $R_A < \infty$ hat.

Beispiel 65 *Der oben beschriebene Greedy-Algorithmus A für das Problem* MINNC *hat die Performanz $R_A = \infty$ (Beispielgraph). Dagegen hat der folgende Algorithmus B eine Performanz $R_B \leq 2$, weshalb* MINNC *zu* APX *gehört.*

```
1  Eingabe: G = (V, E)
2    K ← ∅
3    while E ≠ ∅ do
4      Wähle eine beliebige Kante e = {u, v} in E
5      K ← K ∪ {u, v}
6      G ← G − {u, v}
7    end
```

Ist nun $K = \{u_1, v_1, \ldots, u_k, v_k\}$ die von B unter Auswahl der Kanten $e_i = \{u_i, v_i\}$ berechnete Knotenüberdeckung und C eine optimale Knotenüberdeckung, so muss C für $i = 1, \ldots, k$ mindestens einen der beiden Endknoten von e_i enthalten. Folglich ist $\|C\| \geq k$ und somit

$$R_B(x) = \frac{B(x)}{OPT(x)} = \frac{\|K\|}{\|C\|} = \frac{2k}{\|C\|} \leq 2.$$

Beispiel 66

- *(siehe Extrablatt)*
- *Für das Minimierungsproblem* BIN PACKING,

  D = *Menge aller endlichen Folgen* $s = (s_1, \ldots, s_n)$ *mit* $s_i \in (0, 1)$ *für* $i = 1, \ldots, n$*;*
  $S(s)$ = *Menge aller Partitionen* $\sigma = \{B_1, \ldots, B_k\}$ *von* $\{1, \ldots, n\}$*, so dass für* $i = 1, \ldots, k$ *gilt:*

  $$\sum_{i \in B_j} s_i \quad \leq \quad 1$$

  (Interpretation: $s_1, \ldots, s_n$ *sind Größenangaben für* n *Gegenstände, die in möglichst wenig Behälter der Kapazität 1 gepackt werden sollen.)*
  $f(\sigma) = \|\sigma\| = k$.

  ist leicht eine Performanz von 2 zu erreichen. Hierzu betrachten wir den First-Fit Algorithmus FF*, der die Gegenstände der Reihe nach in den jeweils ersten Behälter packt, in den er passt, d.h. der i-te Gegenstand kommt in Behälter* B_j*, wobei*

  $$j \quad = \quad \min\{m \geq 1 \mid s_i + \sum_{l \in B_m} s_l \leq 1\}$$

  Wegen $FF(s) \leq \lceil 2 \cdot \sum_{i=1}^{n} s_i \rceil$ *und* $OPT(s) \leq \lceil \sum_{i=1}^{n} s_i \rceil$ *folgt für alle Eingabefolgen* s*,*

  $$FF(s) \quad \leq \quad 2 \cdot OPT(s)$$

  und daher $R_{FF} \leq 2$.

Es konnte sogar gezeigt werden, dass $FF(s) \leq 1.7\,OPT(s) + 2$ für alle Eingabefolgen s gilt. Obwohl dies eine Verbesserung zu sein scheint, lässt sich ausgehend von dieser Schranke nur $R_{FF} \leq 2.7$ ableiten (warum?), d.h. die durch die Schranke $FF(s) \leq 1.7\,OPT(s) + 2$ ausgedrückte Approximationsgüte von FF kommt nur bei Eingabefolgen s mit relativ großem $OPT(s)$ Wert (genauer: $OPT(s) \geq 7$) zum Tragen.

Die **asymptotische Performanz** eines Approximationsalgorithmus' A ist

$$R_A^\infty \quad = \quad \inf\{r \geq 1 \mid \exists t \in \mathbb{R}^+ \; \forall x \in D : OPT(x) \geq t \rightarrow R_A(x) \leq r\}$$

Beispiel 67 *Wie schon erwähnt, gilt* $FF(s) \leq 1.7\,OPT(s) + 2$*. Außerdem kann zu jedem* $t \in \mathbb{R}^+$ *eine Folge* s *konstruiert werden mit* $FF(s) \geq 1.7\,(OPT(s) - 1)$ *und* $OPT(s) \geq t$*. Daraus folgt, dass die asymptotische Performanz* R_{FF}^∞ *von FF gleich* 1.7 *ist.*

Ein Optimierungsproblem heißt **absolut approximierbar**, falls es einen Approximationsalgorithmus A und eine Konstante k gibt, so dass für alle $x \in D$ gilt :

$$|A(x) - OPT(x)| \leq k.$$

Beispiel 68 *Das Minimierungsproblem* PLANAR GRAPH COLORING,

$D =$ *Menge aller ungerichteten planaren Graphen* $G = (V, E)$*;*
$S(G) =$ *Menge aller Färbungen* c *von* V*, d.h.* $c : V \to \{1, \ldots, \|V\|\}$ *und für jedes Paar* $(u, v) \in E$ *gilt* $c(u) \neq c(v)$*;*
$f(c) = \|\{c(u) \mid u \in V\}\|$.

besitzt einen absoluten Approximationsalgorithmus, da für jeden planaren Graphen eine 5-Färbung in Polynomialzeit gefunden werden kann:

$$\exists A \, \forall G : G \text{ planar } \to |A(G) - OPT(G)| \leq k$$

mit $k = 4$ *(sogar* $k = 2$ *ist leicht zu erreichen).*

Für das Minimierungsproblem BINPACKING lässt sich, wie gesehen, leicht eine Performanz von 2 erzielen. Es stellt sich die Frage, ob es Approximationsalgorithmen für BINPACKING gibt, die eine kleinere Performanz haben? Da das NP-vollständige Entscheidungsproblem PARTITION,

gegeben: Eine endliche Folge $s = (s_1, \ldots, s_n)$ mit $s_i \in \mathbb{N}$ für $i = 1, \ldots, n$;
gefragt: Existiert eine Aufteilung von $\{1, \ldots, n\}$ in zwei disjunkte Teilmengen $B, \overline{B}$, so dass

$$\sum_{i \in B} s_i \quad = \quad \sum_{i \notin B} s_i$$

effizient lösbar ist, falls ein Approximationsalgorithmus A für BIN PACKING mit $R_A < 1.5$ existiert, kann im besten Fall eine Performanz von 1.5 für BIN PACKING erzielt werden.

Sei O ein Optimierungsproblem. Dann heißt

$$R_{\text{MIN}}(P) \quad = \quad \inf\{r \geq 1 \mid \exists \text{ Approximationsalgorithmus } A \text{ für } O : R_A \leq r\}$$

die **bestmögliche Performanz** für O.

Im (optimalen) Fall, dass für ein Optimierungsproblem O der Wert von $R_{\text{MIN}}(O) = 1$ ist, existiert also für jedes $\varepsilon > 0$ ein Algorithmus A_ε mit $R_{A_\varepsilon} \leq 1 + \varepsilon$. Falls zu gegebenem ε der zugehörige Algorithmus effektiv konstruiert werden kann, spricht man von einem Approximationsschema.

Sei A ein Algorithmus, der Eingaben der Form (x, ε) entgegennimmt. Für jedes feste $\varepsilon > 0$ bezeichnen wir mit A_ε denjenigen Algorithmus, der sich aus A ergibt, indem

die zweite Eingabekomponente konstant gleich ε gesetzt wird (d.h. es bleibt nur x als Eingabeparameter übrig).

A heißt **polynomielles Approximationsschema** (PAS) für ein Optimierungsproblem, falls für jedes feste $\varepsilon > 0$ gilt:

(i) A_ε ist polynomiell zeitbeschränkt;

(ii) $R_{A_\varepsilon} \leq 1 + \varepsilon$

Ist sogar die Laufzeit von A polynomiell in $|x| + 1/\varepsilon$ beschränkt, so heißt A **volles polynomielles Approximationsschema** (FPAS).

Beispiel 69 *Nehmen wir an, wir hätten einen Algorithmus A, dessen Laufzeit bei Eingabe (x, ε) durch $|x|^{(1/\varepsilon)^{(1/\varepsilon)}} + (1/\varepsilon)^{(1/\varepsilon)}$ beschränkt ist. Dann ist zwar für jedes feste $\varepsilon > 0$ die Laufzeit von A_ε polynomiell beschränkt, A ist jedoch nicht polynomiell in $|x| + 1/\varepsilon$ beschränkt.*

Wie schon erwähnt, gilt (unter der Annahme $\mathcal{P} \neq \mathcal{NP}$) $R_{\text{MIN}}(\text{BIN PACKING}) \geq 1.5$, d.h. es gibt kein PAS für BIN PACKING. Die untere Schranke von 1.5 ergab sich durch die Betrachtung von Eingabefolgen s mit relativ kleinem $OPT(s)$ Wert. Tatsächlich existiert für jedes $\varepsilon > 0$ ein Algorithmus A_ε für BIN PACKING mit $A_\varepsilon(s) \leq (1 + \varepsilon)OPT(s) + 1$, d.h. für Eingabefolgen s mit hinreichend großem $OPT(s)$ Wert lässt sich eine Performanz beliebig nahe an 1 erzielen.

Für ein Optimierungsproblem O ist

$$R^\infty_{\text{MIN}}(O) \quad = \quad \inf\{r \geq 1 \mid \exists \text{ Approximationsalgorithmus } A \text{ für } O : R^\infty_A \leq r\}$$

die **bestmögliche asymptotische Performanz** für O.

Beispiel 70 *Da für* BIN PACKING *für jedes $\varepsilon > 0$ ein Algorithmus A_ε mit $A_\varepsilon(s) \leq (1 + \varepsilon)OPT(s) + 1$ existiert, gilt $R^\infty_{\text{MIN}}(\text{BIN PACKING}) = 1$.*

Für BIN PACKING existieren sogar asymptotische PAS und FPAS.

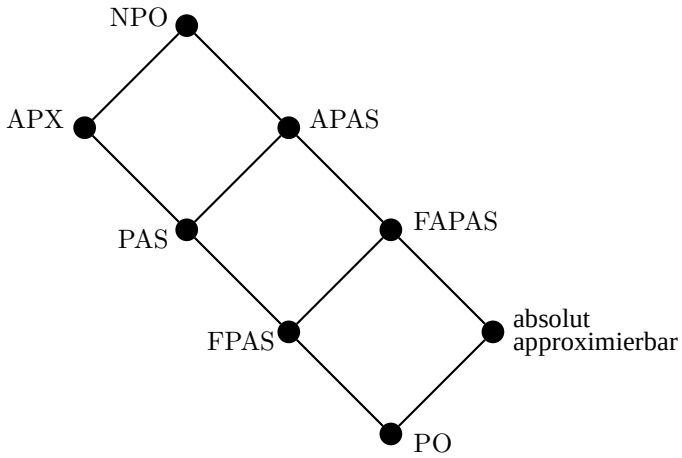
A heißt **asymptotisches polynomielles Approximationsschema** (APAS) für ein Optimierungsproblem, falls für jedes feste $\varepsilon > 0$ gilt:

(i) A_ε ist polynomiell zeitbeschränkt;

(ii) $R^\infty_{A_\varepsilon} \leq 1 + \varepsilon$

Ist sogar die Laufzeit von A polynomiell in $|x| + 1/\varepsilon$ beschränkt, so heißt A **volles asymptotisches polynomielles Approximationsschema** (FAPAS).

Eine Möglichkeit, Optimierungsprobleme zu klassifizieren, besteht darin, zu untersuchen, ob sie absolut approximierbar sind und ob sie ein FPAS, PAS, FAPAS oder ein APAS haben. Es ist klar, dass jedes absolut approximierbare Optimierungsproblem ein FAPAS und damit auch ein APAS hat. Generell bestehen die folgenden Beziehungen:



Für jedes Optimierungsproblem O, das ein PAS hat, gilt natürlich $R^{\infty}_{\text{MIN}}(O) = R_{\text{MIN}}(O) = 1$, und es gilt $R^{\infty}_{\text{MIN}}(O) = 1$, falls O ein APAS hat. Theoretisch ist es jedoch möglich, dass für ein Optimierungsproblem O $R^{\infty}_{\text{MIN}}(O) = 1$, aber $R_{\text{MIN}}(O) = \infty$ ist. Dieser Fall könnte zum Beispiel eintreten, falls für Eingaben x mit kleinem $OPT(x)$-Wert in Polynomialzeit nur Lösungen $y \in S(x)$ mit $f(y) > |x|$ gefunden werden können, während für Eingaben x mit großem $OPT(x)$-Wert leicht optimale Lösungen berechnet werden können.

Die folgende Tabelle zeigt für eine Reihe von Optimierungsproblemen, welche Typen von Approximationsalgorithmen für sie existieren (falls $\mathcal{P} \neq \mathcal{NP}$).

	FPAS	PAS	absolut	FAPAS	APAS	sonstiges
BIN-PACKING	Nein	Nein	?	Ja	Ja	$R_{\text{MIN}} = 1.5$, $R^{\infty}_{\text{MIN}} = 1$
KNAPSACK	Ja	Ja	Nein	Ja	Ja	$R_{\text{MIN}} = R^{\infty}_{\text{MIN}} = 1$
VERTEXCOVER	?	?	?	?	?	$R^{\infty}_{\text{MIN}} \leq R_{\text{MIN}} \leq 2$
SETCOVER	?	?	?	?	?	?
SCHEDULING	Nein	Ja	Nein	Nein	Ja	$R_{\text{MIN}} = R^{\infty}_{\text{MIN}} = 1$
ΔTSP	?	?	Nein	?	?	$R^{\infty}_{\text{MIN}} = R_{\text{MIN}} \leq 1.5$
TSP	Nein	Nein	Nein	Nein	Nein	$R^{\infty}_{\text{MIN}} = R_{\text{MIN}} = \infty$
COLORING	Nein	Nein	Nein	Nein	Nein	$2 \leq R^{\infty}_{\text{MIN}} = R_{\text{MIN}}$
MAXCLIQUE	Nein	?	Nein	Nein	?	$R_{\text{MIN}} = 1 \vee R^{\infty}_{\text{MIN}} = \infty$