Algorithmus 3.34 (Ein DNF-Algorithmus)

Eingabe: Eine aussagenlogische Formel φ

Ausgabe: Eine zu φ äquivalente Formel φ' in DNF

Verfahren:

1) Konstruiere eine zu φ äquivalente Formel φ' in NNF.
2) Wiederhole folgende Schritte:
3) Falls φ' in DNF ist, so halte mit Ausgabe φ'.
4) Ersetze eine Teilformel von φ' der Gestalt
$(\psi_1 \wedge (\psi_2 \vee \psi_3))$ durch $((\psi_1 \wedge \psi_2) \vee (\psi_1 \wedge \psi_3))$ oder
ersetze eine Teilformel von φ' der Gestalt
$((\psi_2 \vee \psi_3) \wedge \psi_1)$ durch $((\psi_2 \wedge \psi_1) \vee (\psi_3 \wedge \psi_1))$.
Sei φ'' die resultierende Formel.
5) Setze $\varphi' := \varphi''$

Satz 3.35 (Korrektheit der Algorithmen 3.33 und 3.34)

Für jede aussagenlogische Formel φ gilt:

(a) Algorithmus 3.33 hält bei Eingabe der Formel φ nach endlich vielen Schritten an und gibt eine zu φ äquivalente Formel in KNF aus.

(b) Algorithmus 3.34 hält bei Eingabe der Formel φ nach endlich vielen Schritten an und gibt eine zu φ äquivalente Formel in DNF aus.

(hier ohne Beweis)

Beispiel 3.36

Sei $\varphi := \left(\left(\neg V_0 \wedge (V_0 \to V_1)\right) \vee (V_2 \to V_3)\right)$

Transformation von φ in NNF:

$$\varphi = \left(\left(\neg V_0 \wedge (V_0 \to V_1)\right) \vee (V_2 \to V_3)\right)$$
$$\equiv \left(\left(\neg V_0 \wedge (\neg V_0 \vee V_1)\right) \vee (\neg V_2 \vee V_3)\right) =: \varphi'$$

Transformation von φ in DNF mittels Algorithmus 3.34:

Schritt 1: Transformation von φ in NNF
$\rightsquigarrow$ liefert $\varphi' = \left(\left(\neg V_0 \wedge (\neg V_0 \vee V_1)\right) \vee (\neg V_2 \vee V_3)\right)$

1-maliges Anwenden von Zeile 4 des Algo. auf φ' liefert:

$$\varphi'' := \left(\left(\left(\underline{\neg V_0 \wedge \neg V_0}\right) \vee \left(\underline{\neg V_0 \wedge V_1}\right)\right) \vee \left(\underline{\neg V_2} \vee \underline{V_3}\right)\right)$$

Diese Formel ist die DNF-Formel, die von dem Algo. ausgegeben wird.

Transformation von φ in KNF mittels Algorithmus 3.33:

Schritt 1: Transformation von φ in NNF
$\rightsquigarrow$ liefert $\varphi' = \left(\left(\neg V_0 \wedge (\neg V_0 \vee V_1)\right) \vee (\neg V_2 \vee V_3)\right)$

1-maliges Anwenden von Zeile 4 des Algo. liefert:

$$\varphi'' := \left(\left(\underline{\neg V_0 \vee (\neg V_2 \vee V_3)}\right) \wedge \left(\underline{(\neg V_0 \vee V_1) \vee (\neg V_2 \vee V_3)}\right)\right)$$

Dies ist die KNF-Formel, die von dem Algo. ausgegeben wird.

Unmittelbar vor Definition 3.15 wurde darauf hingewiesen, dass die Aufgabe, für eine gegebene Formel φ herauszufinden, ob sie erfüllbar ist, im Allgemeinen ein recht schwieriges Problem ist. Für den Spezialfall, dass φ eine Formel in **DNF** ist, lässt sich das Erfüllbarkeitsproblem allerdings sehr effizient lösen, wie die folgende Beobachtung zeigt.

<u>Beobachtung 3.37</u>: (effizienter Erfüllbarkeitstest für DNF-Formeln)

Sei φ eine Formel in DNF, d.h. φ ist von der Form

$$\bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} l_{i,j} \right), \quad \text{für Literale } l_{i,j}.$$

D.h. φ ist von der Form

$$\alpha_1 \vee \cdots \vee \alpha_n, \quad \text{wobei}$$

für jedes $i \in \{1, \dots, n\}$ α_i die konjunktive Klausel

$$\alpha_i := l_{i,1} \wedge \cdots \wedge l_{i,m}$$

ist.

Klar: φ ist erfüllbar $\iff$ Für mindestens ein $i \in \{1, \dots, n\}$ ist α_i erfüllbar.

Da α_i eine Konjunktion von Literalen (d.h. von Variablen und/oder negierten Variablen) ist, gilt:

$\mathcal{K}_i$ ist erfüllbar $\iff$ es gibt keine $j, j' \in \{1, \dots, m_i\}$, so dass $l_{i,j} = \neg l_{i,j'}$.

Daher ist der folgende Algorithmus dafür geeignet, zu testen, ob eine gegebene DNF-Formel erfüllbar ist.

<u>Eingabe:</u> Eine aussagenlogische Formel $\varphi = \bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} l_{i,j} \right)$ in DNF

<u>Ziel:</u> Entscheide, ob φ erfüllbar ist

<u>Verfahren:</u>

1.) Für $i = 1, \dots, n$
2.) Für $j = 1, \dots, m_i$
3.) Für $j' = j+1, \dots, m_i$
4.) Falls $l_{i,j} = \neg l_{i,j'}$ oder $l_{i,j'} = \neg l_{i,j}$, dann (falls $i = n$ ist, so mache in Zeile 6 weiter; ansonsten setze $i := i+1$ und mache in Zeile 2 weiter)
5.) Halte mit Ausgabe "φ ist erfüllbar"
6.) Halte mit Ausgabe "φ ist unerfüllbar"

Um aussagenlogische Formeln φ von <u>beliebiger</u> Form auf Erfüllbarkeit zu testen, kann man dann folgendermaßen vorgehen:

<u>Schritt 1:</u> Transformiere φ in eine äquivalente Formel φ' in DNF (z.B. mit Algo. 3.34)

<u>Schritt 2:</u> Entscheide, ob φ' erfüllbar ist (z.B. mit dem obigen Verfahren)

Das Durchführen von Schritt 1 kann dabei u.U. aber leider wieder sehr lange dauern, da es einige Formeln gibt, zu denen äquivalente Formeln in DNF zwangsläufig sehr groß sind. Dies wird durch den folgenden Satz präzisiert:

<u>Satz 3.38</u>:

Sei $n \in \mathbb{N}_{>0}$, seien $X_1, \dots, X_n, Y_1, \dots, Y_n$ $2n$ verschiedene aussagenlogische Variablen, und sei

$$\varphi_n := \bigwedge_{i=1}^{n} (X_i \leftrightarrow Y_i).$$

Dann hat jede zu φ_n äquivalente Formel in DNF mindestens 2^n konjunktive Klauseln.

<u>Beweis</u>: Übung.

4. Graphen und Bäume

Bei Modellierungsaufgaben geht es oft darum, Objekte sowie Beziehungen zwischen Objekten zu beschreiben. Graphen und Bäume (Bäume sind Graphen mit bestimmten Eigenschaften) eignen sich dazu oft besonders gut.

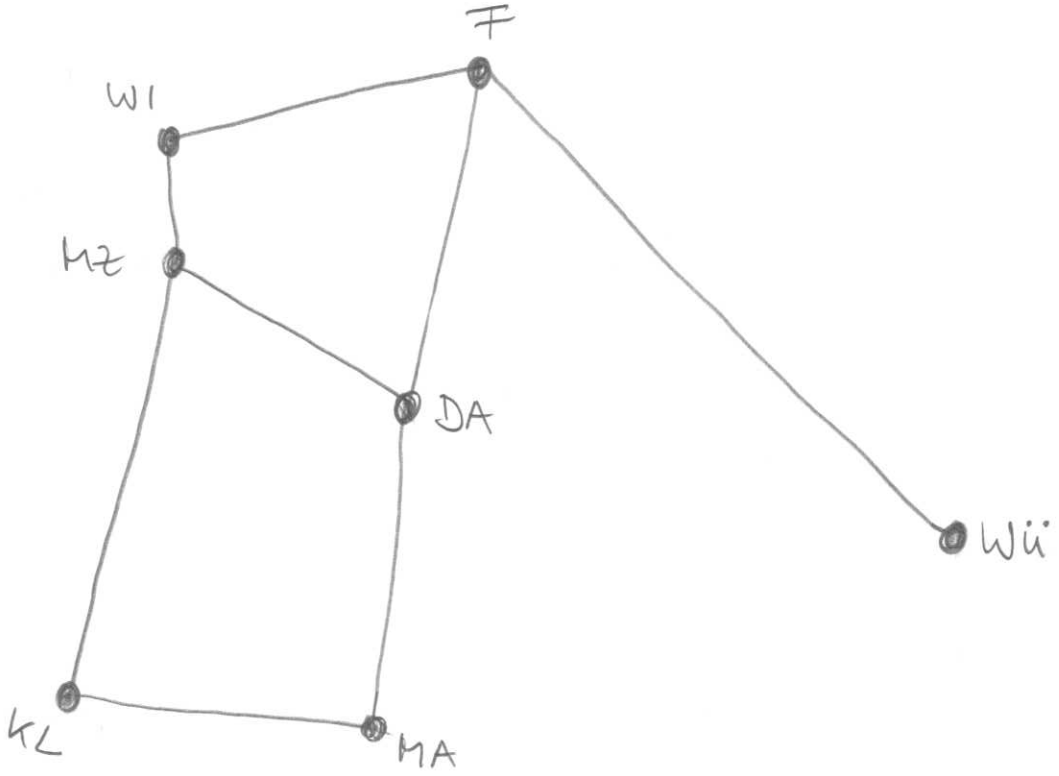
Anschaulich besteht ein Graph aus Knoten und Kanten:

- "Knoten" repräsentieren dabei "gleichartige Objekte"
- "Kanten" repräsentieren Beziehungen zwischen je zwei "Objekten".

Je nach Aufgabenstellung werden ungerichtete Graphen oder gerichtete Graphen verwendet.

Beispiel 4.1

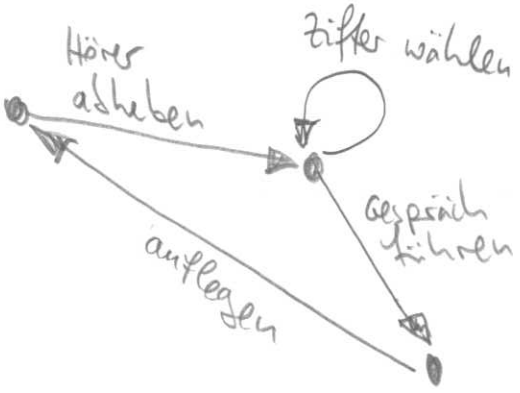
(a) Skizze eines ungerichteten Graphen, der die Autobahnverbindungen zwischen einigen Städten darstellt:



F ≙ Frankfurt
WI ≙ Wiesbaden
MZ ≙ Mainz
KL ≙ Kaiserslautern

DA ≙ Darmstadt
MA ≙ Mannheim
Wü ≙ Würzburg

(b) Skizze eines gerichteten Graphen, der den prinzipiellen Ablauf eines Telefonats darstellt:



4.1 Graphen

Grundlegende Definitionen

Definition 4.2 (ungerichteter Graph)

Ein <u>ungerichteter Graph</u> $G = (V, E)$ besteht aus einer Menge V, die <u>Knotenmenge von G</u> genannt wird, und einer Menge

$$E \subseteq \{ \{i,j\} : i \in V,\ j \in V,\ i \neq j \},$$

die <u>Kantenmenge von G</u> genannt wird. Die Elemente aus V heißen <u>Knoten</u> von G (auch: "Ecken"; englisch: <u>vertices</u>, singular: vertex); die Elemente aus E heißen <u>Kanten</u> von G (englisch: <u>edges</u>, singular: edge).

Beispiel 4.3:
$G = (V, E)$ mit

$V := \{ MZ, WI, MA, DA, KL, F, Wü \}$ und

$E := \{ \{MZ, WI\}, \{WI, F\}, \{F, DA\}, \{F, Wü\}, \{MZ, DA\}, \{MZ, KL\}, \{KL, MA\}, \{D, MA\} \}$

ist ein ungerichteter Graph, der die Autobahnverbindungen zwischen Mainz (MZ), Wiesbaden (WI), Mannheim (MA), Darmstadt (DA), Kaiserslautern (KL), Frankfurt (F) und Würzburg (Wü) repräsentiert.

Beispiel 4.1 (a) zeigt diesen Graphen G in <u>graphischer Darstellung</u>: Knoten werden als Punkte dargestellt, Kanten als Verbindungslinien zwischen Punkten.

<u>Beachte</u>: Laut Definition 4.2 gibt es zwischen zwei Knoten i und j aus V

- <u>höchstens</u> eine Kante; diese wird mit $\{i,j\}$ bezeichnet und graphisch dargestellt als 

- <u>keine</u> Kante, falls $i=j$ ist. In der graphischen Darstellung eines ungerichteten Graphs sind also "Schleifen" der Form 

<u>nicht</u> erlaubt.

Jede Kante $\{i,j\}$ eines ungerichteten Graphen ist also eine <u>2</u>-elementige Menge von Knoten des Graphen.

(<u>Bemerkung</u>: Diese Definition ungerichteter Graphen wird in den meisten Büchern verwendet. Im Buch von Kastens und Kleine-Büning werden davon abweichend in ungerichteten Graphen auch "Schleifen" der Form

erlaubt.)

Notation 4.4:

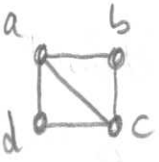
Sei $G = (V, E)$ ein ungerichteter Graph.

- Ein Knoten $v \in V$ heißt <u>inzident</u> mit einer Kante $e \in E$, falls $v \in e$.
- Die beiden mit einer Kante $e \in E$ inzidenten Knoten nennen wir die <u>Endknoten von e</u>, und wir sagen, dass e diese beiden Knoten <u>verbindet</u>.
- Zwei Knoten $v, v' \in V$ heißen <u>benachbart</u> (bzw. <u>adjazent</u>), falls es eine Kante $e \in E$ gibt, deren Endknoten v und v' sind (dh $e = \{v, v'\}$).

Definition 4.5: (Grad)

Sei $G = (V, E)$ ein ungerichteter Graph und sei $v \in V$ ein Knoten von G.

Der <u>Grad von v in G</u> (engl.: degree), kurz: $\mathrm{Grad}_G(v)$, ist die Anzahl der Kanten, die v als Endknoten haben.
D.h. $\mathrm{Grad}_G(v) := |\{e \in E : v \in e\}|$.
Der <u>Grad von G</u> ist $\mathrm{Grad}(G) := \max\{\mathrm{Grad}_G(v) : v \in V\}$, d.h. $\mathrm{Grad}(G)$ gibt den maximalen Grad eines Knotens von G an.

<u>Bsp</u>: $G =$ 

$\mathrm{Grad}_G(a) = 3$ $\quad \mathrm{Grad}_G(b) = 2$ $\quad \mathrm{Grad}(G) = 3$

$\mathrm{Grad}_G(d) = 2$ $\quad \mathrm{Grad}_G(c) = 3$

Definition 4.6 (gerichteter Graph)

Ein <u>gerichteter Graph</u> $G = (V, E)$ besteht aus einer Menge V, die <u>Knotenmenge von G</u> genannt wird, und einer Menge

$$E \subseteq \{ (i,j) : i \in V, \ j \in V \},$$

die <u>Kantenmenge von G</u> genannt wird. Die Elemente aus V heißen <u>Knoten</u> (bzw "Ecken"), die Elemente aus E heißen (gerichtete) <u>Kanten</u> von G.

Beispiel 4.7

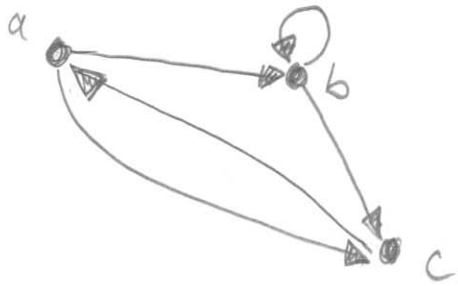
$G = (V, E)$ mit

$V := \{ a, b, c \}$ und

$E := \{ (a,b), (b,b), (b,c), (c,a), (a,c) \}$

ist ein gerichteter Graph.

Graphische Darstellung:



Knoten werden dabei als Punkte dargestellt; eine Kante der Form (i,j) wird als Pfeil von Knoten i nach Knoten j dargestellt, also $i \longrightarrow j$.

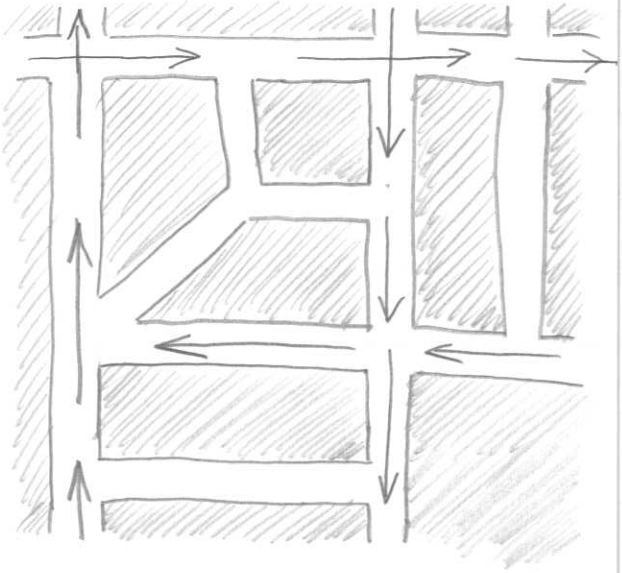
Notation 4.8

Sei $G = (V, E)$ ein gerichteter Graph.

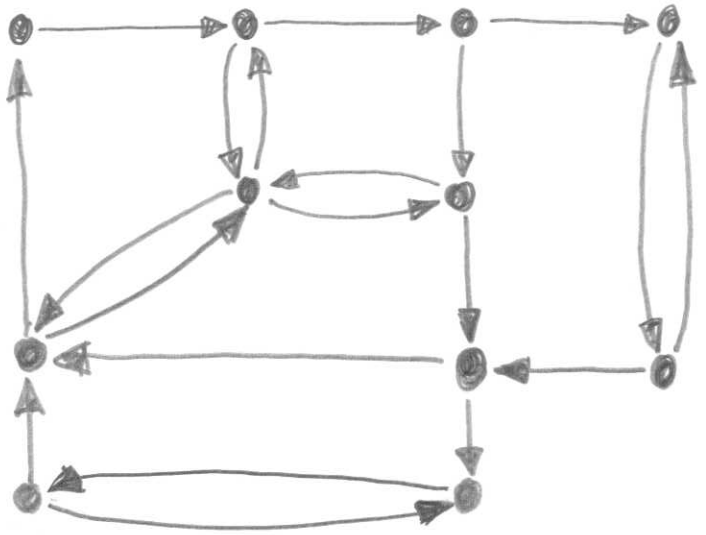
- Ist $e = (i, j) \in E$, so heißt i der Ausgangsknoten von e und j der Endknoten von e, und wir sagen, dass e von i nach j verläuft.
- Ein Knoten $v \in V$ heißt <u>inzident</u> mit einer Kante $e \in E$, falls v der Ausgangs- oder Endknoten von e ist.
- Zwei Knoten $v, v' \in V$ heißen <u>benachbart</u> (bzw. <u>adjazent</u>), falls $(v, v') \in E$ oder $(v', v) \in E$.
- Eine Kante der Form (v, v) (d.h. deren Ausgangs- und Endpunkt identisch ist), wird <u>Schleife</u> bzw. <u>Schlinge</u> genannt.

Beispiel 4.9: (Modellierung durch gerichtete Graphen)

In der folgenden Straßenkarte sind Einbahnstraßen durch Pfeile markiert.



Diese Straßenkarte können wir durch einen gerichteten Graphen repräsentieren, der für jede Straßenkreuzung einen Knoten enthält, und in dem es eine Kante von "Kreuzung" i zu "Kreuzung" j gibt, falls man von i nach j fahren kann, ohne zwischendurch eine weitere Kreuzung zu passieren. Graphisch lässt sich dieser gerichtete Graph folgendermaßen darstellen:



Weitere Beispiele zur Modellierung durch Graphen:

- Computer-Netzwerk:
  Knoten repräsentieren Computer; Kanten repräsentieren Netzwerkverbindungen

- das World Wide Web:
  Knoten repräsentieren Webseiten; Kanten repräsentieren Hyperlinks

Definition 4.10

Sei $G = (V, E)$ ein gerichteter Graph und sei $v \in V$ ein Knoten von G.

- Der <u>Ausgangsgrad von v in G</u> (engl.: out-degree), kurz: $\text{Aus-Grad}_G(v)$, ist die Anzahl der Kanten, die v als Ausgangsknoten haben.

  D.h.: $\text{Aus-Grad}_G(v) := |\{e \in E : \text{es ex. } v' \in V \text{ s.d. } e = (v, v')\}|$

- Der <u>Eingangsgrad von v in G</u> (engl: in-degree), kurz: $\text{Ein-Grad}_G(v)$, ist die Anzahl der Kanten, die v als Eingangsknoten haben.

  D.h.: $\text{Ein-Grad}_G(v) := |\{e \in E : \text{es ex. } v' \in V \text{ s.d. } e = (v', v)\}|$.

<u>Bsp:</u> $G =$ 

$\text{Ein-Grad}_G(a) = 0$ $\quad$ $\text{Aus-Grad}_G(a) = 1$

$\text{Ein-Grad}_G(b) = 2$ $\quad$ $\text{Aus-Grad}_G(b) = 1$

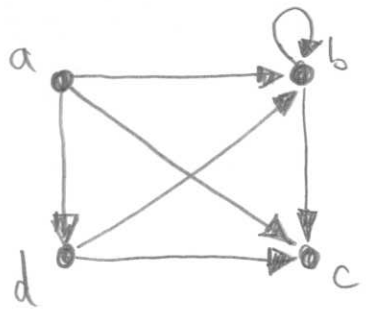
Bemerkung 4.11 (Darstellung von Graphen)

Es gibt mehrere Arten, Graphen darzustellen, zum Beispiel

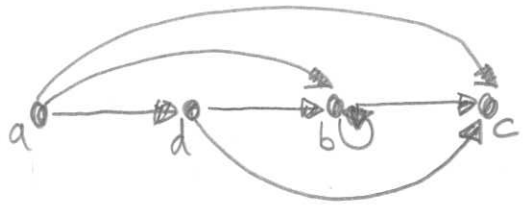
- <u>abstrakt</u>, durch Angabe der Knotenmenge V und der Kantenmenge E

  <u>Beispiel:</u> $G_1 = (V_1, E_1)$ mit
  $V_1 = \{a, b, c, d\}$ und
  $E_1 = \{(a,b), (a,c), (a,d), (b,b), (b,c), (d,b), (d,c)\}$

- graphisch (bzw. anschaulich). Der Beispiel-Graph G_1 wird graphisch dargestellt durch:



oder, äquivalent dazu, durch

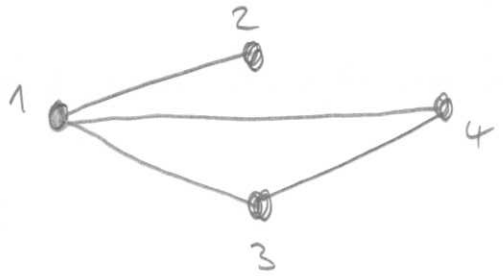


- durch Angabe einer Adjazenzliste, die zu jedem Knoten i eine Liste aller Knoten angibt, zu denen eine von i ausgehende Kante führt.
Der Beispiel-Graph G_1 wird durch folgende Adjazenzliste repräsentiert:

Knoten	Nachbarn
a	(b, c, d)
b	(b, c)
c	()
d	(b, c)

Auf die gleiche Art können auch ungerichtete Graphen durch eine Adjazenzliste repräsentiert werden.

Beispielsweise der Graph $G_2 :=$



durch die Adjazenzliste

Knoten	Nachbarn
1	(2, 3, 4)
2	(1)
3	(1, 4)
4	(1, 3)

- durch Angabe einer <u>Adjazenzmatrix</u>, d.h. eine Tabelle, deren Zeilen und Spalten mit Knoten beschriftet ist, und die in der mit Knoten i beschrifteten Zeile und der mit Knoten j beschrifteten Spalte den Eintrag 1 hat, falls es eine Kante von Knoten i nach Knoten j gibt — und den Eintrag 0, falls es keine Kante von i nach j gibt.

Adjazenzmatrix von G_1:

	a	b	c	d
a	0	1	1	1
b	0	1	1	0
c	0	0	0	0
d	0	1	1	0

Adjazenzmatrix von G_2:

	1	2	3	4
1	0	1	1	1
2	1	0	0	0
3	1	0	0	1
4	1	0	1	0

Wege in Graphen

Definition 4.12

Sei $G = (V, E)$ ein (gerichteter oder ungerichteter) Graph.

(a) Ein <u>Weg</u> in G ist ein Tupel

$$(v_0, \dots, v_\ell) \in V^{\ell+1},$$

für ein $\ell \in \mathbb{N}$, so dass f.a. i mit $0 \leq i < \ell$ gilt:

- falls G ein gerichteter Graph ist, so ist $(v_i, v_{i+1}) \in E$
- falls G ein ungerichteter Graph ist, so ist $\{v_i, v_{i+1}\} \in E$

Das Tupel $(v_0, \dots, v_\ell)$ wird dann "ein Weg von v_0 nach v_ℓ" genannt; ℓ ist die <u>Länge des Weges</u> (dh: die <u>Länge</u> des Weges gibt gerade an, wie viele <u>Kanten</u> auf dem Weg durchlaufen werden).

<u>Beachte</u>: Gemäß dieser Definition ist für jedes $v \in V$ das Tupel (v) ein Weg der Länge 0 von v nach v.

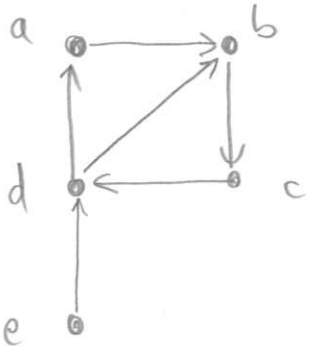
(b) Ein Weg heißt <u>einfach</u>, wenn kein Knoten mehr als einmal in dem Weg vorkommt

(c) Ein Weg $(v_0, \dots, v_\ell)$ heißt <u>Kreis</u>, wenn $\ell \geq 1$ und $v_\ell = v_0$ ist.

(d) Ein Kreis $(v_0, \dots, v_\ell)$ heißt <u>einfach</u>, wenn $(v_0, \dots, v_{\ell-1})$ ein einfacher Weg ist.

Beispiel 4.13

$G :=$



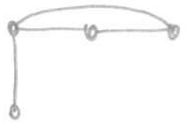
- (e, d, b, c, d) ist ein Weg der Länge 4, aber kein einfacher Weg
- (d, b, c, d) ist ein einfacher Kreis
- (e, d, a, b) ist ein einfacher Weg
- (b, d, a) ist kein Weg.
- (a, b, c, d, b, c, d, a) ist ein Kreis, aber kein einfacher Kreis.

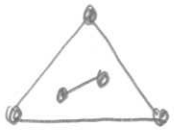
Definition 4.14

Ein gerichteter Graph heißt <u>azyklisch</u>, falls er keinen Kreis enthält. Solche Graphen werden im Englischen "directed acyclic graph", kurz: DAG, genannt.

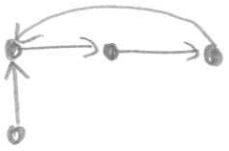
Definition 4.15 (zusammenhängend, stark zusammenhängend, orientierbar)

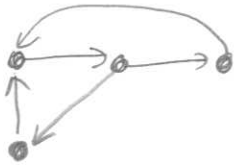
(a) Ein ungerichteter Graph $G = (V, E)$ heißt <u>zusammenhängend</u>, wenn für alle Knoten $v, w \in V$ gilt: Es gibt in G einen Weg von v nach w.

Beispiel:  ist zusammenhängend,

 ist nicht zusammenhängend

(b) Ein gerichteter Graph $G = (V, E)$ heißt <u>stark zusammenhängend</u>, wenn für alle Knoten $v, w \in V$ gilt: Es gibt in G einen Weg von v nach w.

Beispiel:  ist <u>nicht</u> stark zusammenhängend (da es z.B. keinen Weg vom Knoten links oben zum Knoten links unten gibt)

 ist stark zusammenhängend

(c) Ein ungerichteter Graph $G = (V, E)$ heißt <u>orientierbar</u>, wenn man für jede Kante $e \in E$ eine Richtung so festlegen kann, dass der entstehende gerichtete Graph stark zusammenhängend ist.

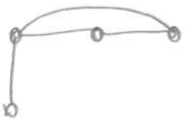
Beispiel:  ist orientierbar;  nicht.

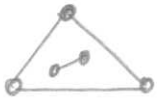
Definition 4.16 (Zusammenhangskomponente, starke Zusammenhangskomponente)

(a) Sei $G = (V, E)$ ein ungerichteter Graph.
Ein Graph $G' = (V', E')$ heißt <u>Zusammenhangskomponente</u> von G, falls die folgenden Bedingungen erfüllt sind:

1) G' ist ein Teilgraph von G, dh $V' \subseteq V$ und $E' \subseteq E$
2) G' ist zusammenhängend
3) Für jeden zusammenhängenden Teilgraphen $G'' = (V'', E'')$ von G mit $V' \subseteq V''$ und $E' \subseteq E''$ gilt:
$V' = V''$ und $E' = E''$.

<u>Beispiel:</u>

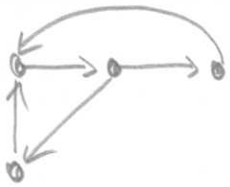
$G =$  hat nur eine Zusammenhangskomponente, nämlich G selbst

$G =$  hat zwei Zusammenhangskomponenten, nämlich  und 

(b) Sei $G = (V, E)$ ein gerichteter Graph
Ein Graph $G' = (V', E')$ heißt <u>starke Zusammenhangskomponente</u> von G, falls die folgenden Bedingungen erfüllt sind:

1) G' ist ein Teilgraph von G
2) G' ist stark zusammenhängend
3) Für jeden stark zusammenhängenden Teilgraphen $G'' = (V'', E'')$ von G mit $V' \subseteq V''$ und $E' \subseteq E''$ gilt:
$V' = V''$ und $E' = E''$.

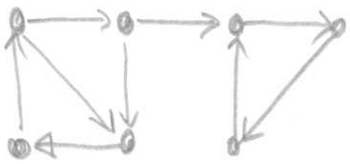
Beispiel:

G = 

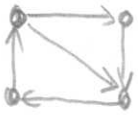
hat nur eine starke Zusammenhangs-Komponente, nämlich G selbst.

G = 

hat zwei starke Zusammenhangskomponenten, nämlich  und 

G = 

hat zwei starke Zusammenhangs-Komponenten, nämlich

 und 

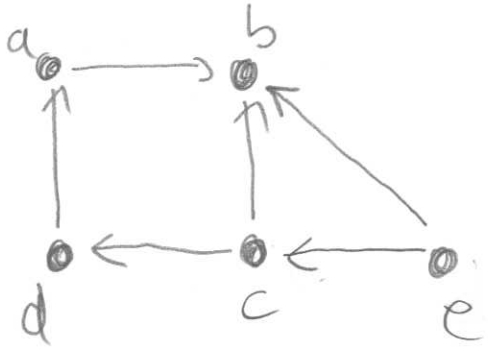
Definition 4.17 (Hamilton-Kreis und Hamilton-Weg)

Sei $G = (V, E)$ ein (gerichteter oder ungerichteter) Graph

(a) Ein Weg $W = (v_0, \dots, v_\ell)$ heißt Hamilton-Weg, wenn jeder Knoten aus V genau einmal in W vorkommt.

(b) Ein Weg $W = (v_0, \dots, v_\ell)$ heißt Hamilton-Kreis, wenn $\ell \geq 1$ und $v_\ell = v_0$ und $(v_0, \dots, v_{\ell-1})$ ein Hamilton-Weg ist.

Beispiel: Der Graph G



hat einen Hamilton-Weg, nämlich

(e, c, d, a, b),

aber keinen Hamilton-Kreis

(da $\text{Aus-Grad}_G(b) = 0$)

Ein Anwendungsbeispiel: Beim Problem des Handlungsreisenden (engl.: Travelling Salesman's Problem) geht es darum, eine Rundreise durch n Städte so durchzuführen, dass jede Stadt dabei genau 1 mal besucht wird. Es geht also darum, einen Hamilton-Kreis zu finden.

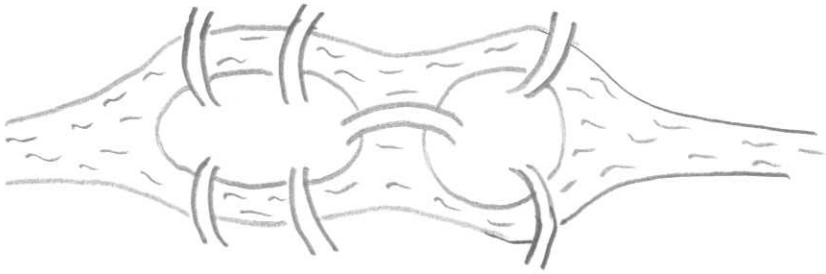
Das Problem, zu einem gegebenen Graphen zu entscheiden, ob er einen Hamilton-Kreis besitzt, ist algorithmisch ein schwieriges Problem:

man kann zeigen, dass es (genau wie das aussagenlogische Erfüllbarkeitsproblem) NP-vollständig ist.

Im Gegensatz zu Hamilton-Wegen (bei denen es darum geht, einen Weg zu finden, der jeden Knoten des Graphen genau einmal besucht), geht es bei den im Folgenden betrachteten Euler-Wegen darum, einen Weg zu finden, der jede Kante des Graphen genau einmal besucht.

Beispiel 4.18 (Königsberger Brückenproblem)

In der Stadt Königsberg gab es im 18. Jahrhundert 7 Brücken über den Fluss Pregel, die die Ufer und 2 Inseln miteinander verbanden. Skizze:



Frage: Gibt es einen Spaziergang, der jede der 7 Brücken genau einmal überquert und zum Ausgangspunkt zurückkehrt?