

# Diskrete Mathematik – Graphentheorie (Übersicht)

Dr. C. Löh

2. Februar 2010

---

## 0 Graphentheorie – Grundlagen

**Definition** (Graph, gerichteter Graph).

- Ein *Graph* ist ein Paar  $G = (V, E)$ , wobei  $V$  eine Menge ist (die Menge der *Knoten*) und  $E \subset \{\{u, v\} \mid u, v \in V, u \neq v\}$  eine Teilmenge ist (die Menge der *Kanten*).
- Ein *gerichteter Graph* ist ein Paar  $G = (V, E)$ , wobei  $V$  eine Menge ist und  $E \subset V \times V$  eine Teilmenge ist.

**Konvention.** Wir werden im folgenden immer annehmen, dass [gerichtete] Graphen nur *endlich viele* Knoten bzw. Kanten haben.

Außerdem wurden die folgenden Begriffe/Beispiele behandelt:

- vollständige Graphen, vollständige bipartite Graphen
- Modellierung von Situationen/Problemen durch Graphen
- Isomorphismen von [gerichteten] Graphen
- adjazente/benachbarte Knoten
- Grad von Knoten in [gerichteten] Graphen, Ein-Grad und Aus-Grad in gerichteten Graphen
- Gradfolge
- Untergraphen und induzierte Untergraphen
- [gerichtete] Wege, [gerichtete] Kreise
- [stark] zusammenhängende [gerichtete] Graphen, [starke] Zusammenhangskomponenten
- Abstand von Knoten
- Adjazenzmatrizen, Adjazenzlisten

**Satz** (Summe der Grade in einem Graph).

- Sei  $G = (V, E)$  ein [gerichteter] Graph. Dann ist

$$\sum_{v \in V} \deg(v) = 2 \cdot \#E.$$

- Insbesondere ist in jedem [gerichteten] Graph die Anzahl der Knoten mit ungeradem Grad gerade.

# 1 Bäume

**Definition** (Baum, Wald).

- Ein Graph ist ein *Baum*, wenn er zusammenhängend ist und keine Kreise enthält.
- Ein Graph ist ein *Wald*, wenn alle Zusammenhangskomponenten Bäume sind.

Weitere Begriffe zu Bäumen:

- Blätter, innere Knoten  
(jeder Baum mit mindestens zwei Knoten besitzt ein Blatt)
- aufspannende Bäume  
(jeder zusammenhängende Graph besitzt einen aufspannenden Baum)

**Satz** (Charakterisierung von Bäumen). *Sei  $G = (V, E)$  ein Graph mit  $V \neq \emptyset$ . Dann sind die folgenden Aussagen äquivalent:*

1. *Der Graph  $G$  ist ein Baum.*
2. *Je zwei Knoten von  $G$  sind durch genau einen Weg verbunden.*
3. *Der Graph  $G$  ist zusammenhängend und es gilt  $\#E = \#V - 1$ .*

Begriffe zu Wurzel- und Binärbäumen:

- Wurzelbäume
- Vorgänger/Nachfolger von Knoten in Wurzelbäumen
- Beispiele von Wurzelbäumen (Spielbäume, Parsebäume)
- Höhe von Wurzelbäumen und Induktion über Teilbäume
- Binärbäume

**Satz** (Höhe von Binärbäumen). *Für alle Binärbäume  $T = (V, E)$  gilt*

$$2^{\text{ht}(T)+1} \geq \#V + 1.$$

Grundlegende Algorithmen zur Bestimmung von aufspannenden Bäumen sind Breitensuche (breadth first search) und Tiefensuche (depth first search), siehe Abbildung 1). Eigenschaften von Breitensuche/Tiefensuche:

- Die Algorithmen terminieren.
- Der ausgegebene Graph ist tatsächlich ein Baum, der alle Knoten aus der Zusammenhangskomponente des Startknotens enthält.
- Für Breitensuche gilt: Mit Hilfe der FIFO-Eigenschaft der Queue kann man induktiv beweisen, dass der ausgegebene Baum kürzeste Wege zwischen dem Startknoten und den anderen Knoten seiner Zusammenhangskomponente liefert.
- Tiefensuche hingegen berechnet im allgemeinen keine kürzesten Wege; dafür ist Tiefensuche „lokaler.“

**Algorithmus** (Breitensuche/Tiefensuche).

<i>Eingabe</i>	Ein Graph $G$ und ein Knoten $r$ von $G$
<i>Ausgabe</i>	Ein aufspannender Baum der Zusammenhangskomponente von $r$ in $G$ .
Datenstrukturen	$T$ ein Graph [die Ausgabe] $M$ eine Menge von Knoten [bereits besichtigte Knoten] $Q$ Queue von Paaren von Knoten [welcher Knoten ist als nächstes zu bearbeiten?] für <i>Breitensuche</i> : FIFO-Queue für <i>Tiefensuche</i> : LIFO-Queue
Initialisierung	$T$ Baum/Graph, der nur den Knoten $r$ enthält $M$ Die Menge $\{r\}$ $Q$ Die FIFO-/LIFO-Queue aller Paare $(v, r)$ enthält, wobei $v$ ein Knoten in $G$ ist, der zu $r$ benachbart ist
<i>Algorithmus</i>	Solange $Q$ nicht-leer ist: <ul style="list-style-type: none"><li>- entferne das erste Element <math>(u, v)</math> aus <math>Q</math></li><li>- falls <math>u \notin M</math> ist:<ul style="list-style-type: none"><li>- füge <math>u</math> und die Kante <math>\{u, v\}</math> zu <math>T</math> hinzu</li><li>- für alle Nachbarn <math>w</math> von <math>u</math> in <math>G</math>:<ul style="list-style-type: none"><li>- falls <math>w \notin M</math> ist: füge <math>(w, u)</math> zu <math>Q</math> hinzu</li></ul></li><li>- füge <math>u</math> zu <math>M</math> hinzu</li></ul></li></ul> Gib $T$ aus.

Abbildung 1: Breitensuche/Tiefensuche

**Definition** (Gewichteter Graph).

- Ein *gewichteter Graph* ist ein Graph  $G = (V, E)$  zusammen mit einer Funktion  $\delta: E \rightarrow \mathbb{R}$ , der *Gewichtsfunktion*.
- Sei  $G = (V, E)$  ein gewichteter Graph mit Gewichtsfunktion  $\delta: E \rightarrow \mathbb{R}$ . Die *Länge* (bezüglich  $\delta$ ) eines Weges  $v_0v_1 \dots v_n$  in  $G$  ist

$$\ell_\delta(v_0v_1 \dots v_n) := \sum_{j=0}^{n-1} \delta(\{v_j, v_{j+1}\}).$$

Dijkstras Algorithmus (siehe Abbildung 2) zur Bestimmung kürzester Wege in gewichteten Graphen (mit nichtnegativer Gewichtsfunktion) ist eine Variante von Breitensuche. Eigenschaften von Dijkstras Algorithmus:

- Der Algorithmus terminiert.
- Der ausgegebene (gewichtete) Graph ist tatsächlich ein Baum, der alle Knoten aus der Zusammenhangskomponente des Startknotens enthält.
- Da die verwendete Queue  $Q$  eine Priority-Queue ist, kann man induktiv zeigen, dass die dritte Komponente der Elemente  $(u, v, d)$  von  $Q$  jeweils die Länge (bezüglich der Gewichtsfunktion) eines kürzesten *bisher bekannten* Weges in  $G$  von  $r$  nach  $u$  ist.

Da die Gewichtsfunktion keine negativen Werte annimmt, kann man zeigen: Ist  $((u, v), d)$  das vorderste Element von  $Q$ , so gibt es in  $G$  keinen Weg von  $r$  nach  $u$ , dessen Länge (bezüglich  $\delta$ ) kleiner als  $d$  ist.

Also liefert der ausgegebene Baum kürzeste Wege von  $r$  in  $G$ .

Minimale aufspannende Bäume in zusammenhängenden Graphen können mit Hilfe von Kruskals Algorithmus bestimmt werden:

**Algorithmus** (Kruskals Algorithmus).

<i>Eingabe</i>	Ein zusammenhängender gewichteter Graph $G = (V, E)$ mit Gewichtsfunktion $\delta$
<i>Ausgabe</i>	Ein minimaler aufspannender Baum von $G$
Datenstrukturen	$T$ ein gewichteter Graph [die Ausgabe]
Initialisierung	$T$ der gewichtete Graph, der alle Knoten aus $G$ enthält aber keine Kanten
<i>Algorithmus</i>	Solange $T$ kein aufspannender Baum von $G$ ist: <ul style="list-style-type: none"> <li>– finde eine Kante <math>e \in E</math>, die nicht in <math>T</math> liegt, minimalen Gewichts <math>\delta(e)</math> mit folgender Eigenschaft: fügt man <math>e</math> zu <math>T</math> hinzu, so entsteht in <math>T</math> kein Kreis.</li> <li>– füge die Kante <math>e</math> mit Gewicht <math>\delta(e)</math> zu <math>T</math> hinzu</li> </ul> Gib $T$ aus.

**Algorithmus** (Dijkstras Algorithmus).

<i>Eingabe</i>	Ein gewichteter Graph $G$ mit nichtnegativer(!) Gewichtsfunktion $\delta$ und ein Knoten $r$ von $G$
<i>Ausgabe</i>	Ein (gewichteter) aufspannender Baum der Zusammenhangskomponente von $r$ in $G$ , der (bezüglich $\delta$ ) kürzeste Wege in $G$ zwischen dem Startknoten und den anderen Knoten dieser Zusammenhangskomponente liefert.
Datenstrukturen	$T$ ein gewichteter Graph [die Ausgabe] $M$ eine Menge von Knoten [bereits besichtigte Knoten] $Q$ Priority-Queue von Paaren $((u, v), d)$ , wobei $u$ und $v$ Knoten von $G$ sind und $d \in \mathbb{R}$ ist (je niedriger die dritte Komponente ist, desto eher wird das Element wieder aus $Q$ entfernt) [welcher Knoten ist als nächstes zu bearbeiten?]
Initialisierung	$T$ gewichteter Graph, der nur den Knoten $r$ enthält $M$ Die Menge $\{r\}$ $Q$ Priority-Queue aller Paare $((v, r), \delta(\{r, v\}))$ , wobei $v$ ein Knoten in $G$ ist, der zu $r$ benachbart ist
<i>Algorithmus</i>	Solange $Q$ nicht-leer ist: <ul style="list-style-type: none"><li>- entferne das erste Element <math>((u, v), d)</math> aus <math>Q</math></li><li>- falls <math>u \notin M</math> ist:<ul style="list-style-type: none"><li>- füge <math>u</math> und die Kante <math>\{u, v\}</math> (mit dem Gewicht <math>\delta(\{u, v\})</math>) zu <math>T</math> hinzu</li><li>- für alle Nachbarn <math>w</math> von <math>u</math> in <math>G</math>:<ul style="list-style-type: none"><li>- falls <math>w \notin M</math> ist: füge <math>((w, u), d + \delta(\{u, w\}))</math> zu <math>Q</math> hinzu</li></ul></li></ul></li><li>- füge <math>u</math> zu <math>M</math> hinzu</li></ul> Gib $T$ aus.

Abbildung 2: Dijkstras Algorithmus

## 2 Eulersche Graphen

**Definition** (Euler-Tour, Euler-Rundtour, Eulerscher Graph).

- Eine *Euler-Tour* in einem Graph  $G = (V, E)$  ist eine Folge  $v_0, \dots, v_n$  von Knoten von  $G$  mit
  - für alle  $j \in \{0, \dots, n-1\}$  ist  $\{v_j, v_{j+1}\} \in E$ ,
  - die Kanten  $\{v_0, v_1\}, \dots, \{v_{n-1}, v_n\}$  sind verschieden, und
  - es ist  $\{\{v_j, v_{j+1}\} \mid j \in \{0, \dots, n-1\}\} = E$ .
- Eine *Euler-Rundtour* in einem Graph  $G = (V, E)$  ist eine Folge  $v_0, \dots, v_n$  von Knoten von  $G$  mit
  - für alle  $j \in \{0, \dots, n-1\}$  ist  $\{v_j, v_{j+1}\} \in E$  und  $\{v_n, v_0\} \in E$ ,
  - die Kanten  $\{v_0, v_1\}, \dots, \{v_{n-1}, v_n\}, \{v_n, v_0\}$  sind verschieden, und
  - es ist  $\{\{v_0, v_1\}, \dots, \{v_{n-1}, v_n\}, \{v_n, v_0\}\} = E$ .
 (Sind nur die ersten beiden Bedingungen erfüllt, so handelt es sich um eine *Rundtour*.)
- Ein Graph heißt *Eulersch*, wenn er eine Euler-Rundtour besitzt.



*Königsberger Brückenproblem.* Ist der Graph im Bild rechts Eulersch?

**Satz** (Charakterisierung Eulerscher Graphen). *Sei  $G$  ein zusammenhängender Graph. Dann sind die folgenden Aussagen äquivalent:*

1. Der Graph  $G$  ist Eulersch.
2. Alle Knoten von  $G$  haben geraden Grad.
3. Die Menge der Kanten von  $G$  kann in (kanten)disjunkte Kreise in  $G$  zerlegt werden.

**Algorithmus** (Hierholzers Algorithmus).

<i>Eingabe</i>	Ein (nicht-leerer) Eulerscher Graph $G$
<i>Ausgabe</i>	Eine Euler-Rundtour in $G$
Datenstrukturen	$c$ eine Folge von Knoten von $G$
Initialisierung	$c$ Folge, die aus einem einzelnen Knoten von $G$ besteht
<i>Algorithmus</i>	Solange $c$ keine Euler-Rundtour von $G$ ist: <ul style="list-style-type: none"> <li>- finde eine Kante <math>e \in E</math>, die nicht in <math>c</math> liegt aber einen Knoten mit <math>c</math> gemeinsam hat</li> <li>- finde eine Rundtour <math>c'</math>, die <math>e</math> aber keine Kante von <math>c</math> enthält</li> <li>- füge <math>c'</math> so in <math>c</math> ein, dass eine Rundtour entsteht</li> </ul> Gib $c$ aus.

**Definition** (Hamiltonscher Graph). Ein Graph heißt *Hamiltonsch*, wenn er einen Kreis besitzt, der jeden Knoten (genau) einmal durchläuft.

*Warnung.* Das Problem zu entscheiden, ob ein gegebener Graph Hamiltonsch ist, ist ein sogenanntes NP-vollständiges Problem.

### 3 Planarität

**Definition** (Planare Einbettung, planarer Graph).

- Sei  $G = (V, E)$  ein gerichteter Graph. Eine *planare Einbettung* von  $G$  besteht aus
  - einer Abbildung  $f: V \rightarrow \mathbb{R}^2$ , und
  - einer Menge  $\{f_e: [0, 1] \rightarrow E \mid e \in E\}$  von stetigen Abbildungen mit folgenden Eigenschaften:
    - Für alle  $e = (u, v) \in E$  ist  $f_e(0) = f(u)$  und  $f_e(1) = f(v)$ .
    - Die Abbildung  $f$  ist injektiv.
    - Die zusammengesetzte Abbildung  $\coprod_{e \in E} f_e|_{(0,1)}: \coprod_{e \in E} (0, 1) \rightarrow \mathbb{R}^2$  ist injektiv.
- Sei  $G = (V, E)$  ein (ungerichteter) Graph. Eine *planare Einbettung* von  $G$  besteht aus der Wahl einer Orientierung auf  $E$  zusammen mit einer planaren Einbettung des zugehörigen gerichteten Graphen.
- Ein Graph heißt *planar*, wenn er eine planare Einbettung besitzt.

**Definition** (Facetten einer planaren Einbettung). Sei  $G = (V, E)$  ein Graph und sei  $(f: V \rightarrow \mathbb{R}^2, \{f_e: [0, 1] \rightarrow \mathbb{R}^2 \mid e \in E\})$  eine planare Einbettung (einer orientierten Version) von  $G$ . Die *Facetten* dieser planaren Einbettung sind die Wegzusammenhangskomponenten von  $\mathbb{R}^2 \setminus \bigcup_{e \in E} f_e([0, 1])$ .

**Satz** (Eulersche Polyederformel). Sei  $G = (V, E)$  ein zusammenhängender planarer Graph mit  $V \neq \emptyset$  und sei  $F$  die Menge der Facetten einer planaren Einbettung von  $G$ . Dann gilt

$$\#V - \#E + \#F = 2.$$

[Der (induktive) Beweis dieses Satzes beruht auf dem *Jordanschen Kurvensatz*.]

**Korollar.** Sei  $G = (V, E)$  ein zusammenhängender planarer Graph mit  $V \neq \emptyset$ .

1. Ist  $\#V \geq 3$ , so ist  $\#E \leq 3 \cdot \#V - 6$ .
2. Der Graph  $G$  besitzt einen Knoten  $v \in V$  mit  $\deg v \leq 5$ .
3. Die Graphen  $K_5$  und  $K_{3,3}$  sind nicht planar.

**Satz** (Kuratowskis Charakterisierung planarer Graphen). Ein zusammenhängender nicht-leerer Graph ist genau dann planar, wenn er keine Verfeinerung von  $K_5$  oder  $K_{3,3}$  als Untergraph enthält. [ohne Beweis]

**Definition** (Verfeinerung). Sei  $G$  ein Graph. Eine *Verfeinerung* von  $G$  ist ein Graph, in dem die Kanten von  $G$  durch unabhängige Wege zwischen ihren Endpunkten ersetzt werden.

**Satz** (Klassifikation der regulären Polyeder). Es gibt genau fünf reguläre (dreidimensionale beschränkte) Polyeder: den Tetraeder, den Würfel, den Oktaeder, den Dodekaeder und den Ikosaeder.

## 4 Färbungen

**Definition** (Färbung, chromatische Zahl). Sei  $G = (V, E)$  ein Graph.

- Sei  $k \in \mathbb{N}$ . Eine  $k$ -Färbung ist eine Abbildung  $c: V \rightarrow \{1, \dots, k\}$  mit

$$\forall_{\{u,v\} \in E} \quad c(u) \neq c(v).$$

- Die *chromatische Zahl* von  $G$  ist definiert als

$$\chi(G) := \min\{k \in \mathbb{N} \mid G \text{ besitzt eine } k\text{-Färbung}\}.$$

- Der Graph  $G$  heißt *bipartit*, falls  $\chi(G) \leq 2$ .

Probleme, die als Färbungsprobleme formuliert werden können:

- Das Färben von Landkarten
- Wieviele Register werden zur Ausführung eines Programms benötigt?

*Warnung.* Das Problem, zu einem gegebenen Graph die chromatische Zahl zu bestimmen, ist ein sogenanntes NP-vollständiges Problem.

**Satz** (Elementare Abschätzungen der chromatischen Zahl).

1. Für alle Graphen  $G$  ist  $\chi(G) \leq \Delta(G) + 1$ , wobei  $\Delta(G)$  der maximale Knotengrad von  $G$  ist.
2. Bäume sind bipartit.
3. Für alle Graphen  $G$  ist  $\chi(G) \geq \omega(G)$ , wobei  $\omega(G)$  die Cliquenzahl von  $G$  ist.

**Satz** (Vier- bzw. Fünffarbensatz).

1. Jeder planare Graph ist 4-färbbar; insbesondere sind alle ebenen Landkarten von zusammenhängenden Ländern 4-färbbar. [ohne Beweis]
2. Jeder planare Graph ist 5-färbbar. [mit Beweis]

**Satz** (Graphen mit großer chromatischer Zahl und großem Umfang).

1. Für alle  $g \in \mathbb{N}$  und alle  $\chi \in \mathbb{N}$  gibt es einen Graph mit Umfang  $g$  und chromatischer Zahl  $\chi$ . [ohne Beweis]
2. Für alle  $\chi \in \mathbb{N}$  gibt es einen Graph mit Umfang mindestens  $4$  und chromatischer Zahl  $\chi$  (Konstruktion von Mycielski).

## 5 Matchings

**Definition** (Matching). Sei  $G = (V, E)$  ein Graph.

- Eine Menge  $M \subset E$  von Kanten von  $G$  heißt *unabhängig*, falls

$$\forall e, f \in M \quad e \cap f = \emptyset.$$

- Ein *Matching* für  $H \subset V$  in  $G$  ist eine Menge  $M \subset E$  unabhängiger Kanten von  $G$  mit

$$H \subset \bigcup M.$$

- Ein *perfektes Matching* von  $G$  ist ein Matching für  $V$ .

**Satz** (Heiratssatz von Hall). Sei  $G = (V, E)$  ein bipartiter Graph mit Bipartition  $V = H \sqcup D$ . Dann gibt es in  $G$  genau dann ein Matching für  $H$ , wenn (sogenannte Heiratsbedingung)

$$\forall S \subset H \quad \#N(S) \geq \#S.$$

**Definition** (Überdeckung). Sei  $G = (V, E)$  ein Graph. Eine *Überdeckung* von  $G$  ist eine Menge  $U \subset V$  von Knoten, so dass jede Kante von  $G$  mindestens einen Knoten aus  $U$  enthält.

**Satz** (Satz von König). Sei  $G$  ein bipartiter Graph. Die maximale Größe eines Matchings in  $G$  stimmt mit der minimalen Größe einer Überdeckung von  $G$  überein.

Der Satz von König ist eines von vielen Beispielen von Sätzen in der Graphentheorie, die ein gewisses Maximum als ein Minimum von etwas anderem erkennen. Ein weiteres prominentes Beispiel ist:

**Satz** (MaxFlow-MinCut-Theorem). Der maximale Gesamtfluss in einem Netzwerk ist gleich der minimalen Kapazität eines Schnittes in diesem Netzwerk. [ohne Beweis]