

Teil 5

# Algorithmen auf Graphen

# Graphen

Graphen gehören in der Informatik zu den am meisten verwendeten mathematischen Strukturen.

Sie treten

- als technische Netzwerke (Internet, LAN, Telefonnetz, Stromnetz),
- als logische Netzwerke (WWW),
- als soziale Netzwerke (Bekannschaftsgraphen etc.),
- als natürliche Netzwerke (Metabolische Netzwerke etc.),
- oder als Abstraktionen (Wort-/Textbeziehungen, Mathematik etc.)

auf.

Deshalb sind viele praktische und theoretische Probleme stark mit Problemen auf Graphen der unterschiedlichsten Form verbunden.

## Definition (Graphen)

Ein (einfacher) Graph  $G = (V, E)$  besteht aus einer (endlichen) Menge  $V$  von **Knoten** und einer Menge  $E$  von **Kanten**. Sei  $[V]^2$  die Menge aller zweielementigen Teilmengen von  $V$ .

- 1 Falls  $E \subseteq [V]^2$  ist  $G$  **ungerichtet**.
- 2 Falls  $E \subseteq V^2 \setminus \{(v, v) \mid v \in V\}$  ist  $G$  **gerichtet**.

Existiert für jede Kante  $e \in E$  eine Zahl  $w(e)$  so sprechen wir von einem **gewichteten** Graphen und bezeichnen ihn häufig mit  $G = (V, E; w)$ .

In einem **ungerichteten** Graphen sind die Kanten somit **Mengen**  $\{u, v\}$  von zwei **verschiedenen** Knoten, d.h.  $\{u, v\} = \{v, u\}$ .

In einem **gerichteten** Graphen sind die Kanten **geordnete Paare**  $(u, v)$  von zwei **verschiedenen** Knoten, d.h.  $(u, v) \neq (v, u)$ .

Zur Vereinfachung, schreiben wir, wenn keine Verwechslung möglich ist, in beiden Fällen  $(u, v)$  für eine Kante von  $u$  nach  $v$ .

# Sprechweisen

Wir verwenden die folgenden Begriffe im Zusammenhang mit Graphen:

- 1 Zwei Knoten sind **adjazent**, wenn eine Kante zwischen ihnen existiert.
- 2 Eine Kante  $(u, v)$  ist **inzident** zu  $u$  und  $v$ .
- 3 Für eine Kante  $e = (u, v)$  ist  $u$  der **Start-** und  $v$  der **Zielknoten** von  $e$ .
- 4 Ist  $(u, v)$  eine Kante, dann ist  $u$  ein **Vorgänger** von  $v$ , und  $v$  ein **Nachfolger** von  $u$ .

Im gewichteten Fall, bezeichnet  $w(u, v)$  das Gewicht der Kante  $(u, v)$ .

Je nach Situation sprechen wir auch von **der Länge**  $l(u, v)$  oder **der Kapazität**  $c(u, v)$  einer Kante.

Im gerichteten Falls ist  $d_{in}(v) := \sum_{u \in V} w(u, v)$  der **Eingangsgrad** und  $d_{out}(v) := \sum_{u \in V} w(v, u)$  der **Ausgangsgrad** von  $v$ .

Im ungerichteten Falls ist  $d(v) := \sum_{u \in V} w(u, v)$  der **Grad** von  $v$ .

## Variationen

In der Regel verbieten wir **Eigenschleifen**, d.h. Kanten der Form  $(v, v)$ . Aber in manchen Situationen sind sie recht praktisch.

Bislang sind wir davon ausgegangen, dass nur **eine** Kante  $(u, v)$  von  $u$  nach  $v$  existiert. In einem **gerichteten Multigraphen** ist die Kantenmenge  $E$  eine beliebige Menge zusammen mit zwei Abbildungen  $\text{source}: E \rightarrow V$  und  $\text{target}: E \rightarrow V$ , so dass  $\text{source}(e)$  der Start- und  $\text{target}(e)$  der Zielknoten einer Kante  $e \in E$  ist.

In einem **ungerichteten Multigraphen** ist die Kantenmenge  $E$  eine beliebige Menge zusammen mit einer Abbildung  $E \rightarrow [V]^2$  in die zweielementigen Teilmengen von  $V$ .

D.h. in einem Multigraphen können **parallele Kanten** vorkommen, d.h. Kanten die dieselben Start- und Zielknoten haben.

# Teil- und Untergraphen

## Definition (Teil- und Untergraph)

Sei  $G = (V, E)$  ein Graph. Ein Graph  $H = (V', E')$  heißt **Teilgraph von  $G$** , wenn  $V' \subseteq V$  und  $E' \subseteq E$ .

Eine Knotenmenge  $V' \subseteq V$  **induziert  $H$** , wenn  $H = (V', E'')$  mit

$$E'' := \{e \in E \mid e = (u, v), u, v \in V'\}.$$

$H$  heißt dann auch **Untergraph** oder **induzierter Teilgraph** von  $G$ .

Der durch  $V \setminus V'$  induzierte Teilgraph von  $G$  wird auch mit  **$G - V'$**  bezeichnet.

Ist  $V' = \{v\}$  schreiben wir auch  **$G - v$** .

# Wege und Zusammenhang

## Definition (Weg)

Sei  $G = (V, E)$  ein Graph. Ein **Weg  $\omega$  von  $u$  nach  $v$  der Länge  $l$**  ist eine Folge  $\omega = (u = v_0, v_1, v_2, \dots, v_l = v)$  von Knoten, so dass  $(v_i, v_{i+1}) \in E$  für  $0 \leq i < l$ .

Ein Weg  $\omega$  heißt **einfach**, wenn die Knoten  $v_0, \dots, v_l$  paarweise verschieden sind.

## Definition (Zusammenhängend)

Ein ungerichteter Graph  $G = (V, E)$  ist **zusammenhängend**, wenn zwischen je zwei verschiedenen Knoten  $u$  und  $v$  ein Weg von  $u$  nach  $v$  existiert.

Ein gerichteter Graph  $G$  ist **stark zusammenhängend**, wenn zwischen je zwei verschiedenen Knoten  $u$  und  $v$  ein Weg von  $u$  nach  $v$  existiert.  $u$

## Definition (Kreis)

Ein **Zyklus** oder **Kreis** der Länge  $l$  in einem Graphen  $G = (V, E)$  ist ein Weg von  $u$  nach  $u$  der Länge  $\geq 1$ . Die **trivialen Kreise**, d.h. Kreise der Länge 0 sollen ausgeschlossen werden.

Ein Kreis  $(v_0, v_1, \dots, v_{l-1}, v_0)$  heißt **einfach**, wenn die Knoten  $v_0, \dots, v_{l-1}$  paarweise verschieden sind.

$G$  heißt **zyklenfrei** oder **azyklisch**, wenn er keinen einfachen Kreis enthält.

# Bäume und Wälder

## Definition (Gerichtete Bäume und Wälder)

Ein gerichteter Graph  $G = (V, E)$  heißt **gerichteter Wald**, wenn er kreisfrei ist und  $d_{in}(v) \leq 1$  für alle Knoten  $v \in V$ . Jeder Knoten mit  $d_{in} = 0$  heißt **Wurzel**.

Ein **gerichteter Baum** ist ein gerichteter Wald mit genau einer Wurzel.

## Definition (Ungerichtete Bäume und Wälder)

Ein ungerichteter Graph  $G = (V, E)$  heißt **ungerichteter Wald**, wenn er kreisfrei ist.

Ein **ungerichteter Baum** ist ein zusammenhängender gerichteter Wald.

# Gerichtete und ungerichtete Graphen

Häufig ist die Unterscheidung zwischen gerichteten und ungerichteten Graphen nicht notwendig.

Ein **ungerichteter** Graph  $G = (V, E)$  kann als ein **gerichteter** Graph  $G' = (V, E')$  mit

$$E' = \{(u, v) \mid \{u, v\} \in E\}$$

betrachtet werden. D.h. jede ungerichtete Kante wird zu zwei entgegengesetzten gerichteten Kanten.

Ein **gerichteter** Graph  $G = (V, E)$  kann einfach als **ungerichteter** Graph interpretiert werden, indem man die Richtungen der Kanten ignoriert.

Teil 5: Algorithmen auf Graphen

# Graphdarstellungen

# Grundlegende Operationen

$u, v$  seien Knoten und  $e$  eine Kante.

$\text{source}(e)$ : Startknoten von  $e$  (gerichtet).

$\text{target}(e)$ : Zielknoten von  $e$  (gerichtet).

$\text{edge}(u, v)$ : true falls eine Kante  $(u, v)$  existiert, sonst false.

$w(e)$ : Das Gewicht der Kante  $e$ .

$w(u, v)$ : Das Gewicht der Kante  $(u, v)$ , falls sie existiert, 0 sonst.

$d_{in}(v)$ : Der Eingangsgrad von  $v$  (gerichtet).

$d_{out}(v)$ : Der Ausgangsgrad von  $v$  (gerichtet).

$d(v)$ : Der Grad von  $v$  ( $d_{in}(v) + d_{out}(v)$ ) im gerichteten Fall).

$\text{adj}(v)$ : Eine Liste aller Nachbarn von  $v$  (ungerichtet).

$\text{incident}(v)$ : Eine Liste aller zu  $v$  inzidenten Kanten (ungerichtet).

$\text{succ}(v)$ : Eine Liste aller Vorgänger von  $v$  (gerichtet).

$\text{in}(v)$ : Eine Liste aller zu  $v$  führenden Kanten (gerichtet).

$\text{pred}(v)$ : Eine Liste aller Nachfolger von  $v$  (gerichtet).

$\text{out}(v)$ : Eine Liste aller von  $v$  ausgehenden Kanten (gerichtet).

# Die Adjazenzmatrix

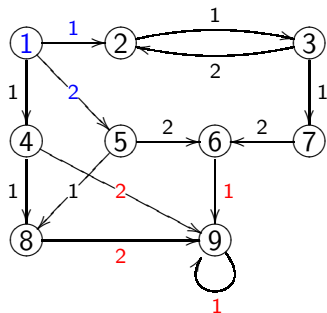
Hat der Graph  $G = (V, E; w)$  genau  $n$  Knoten, gehen wir davon aus, dass  $V = \{1, \dots, n\}$ .

Wir gehen davon aus, dass die Kantengewichte  $w(u, v)$  aus einer Menge  $X$  stammen und es einen Wert  $0 \in X$  gibt, so dass  $w(u, v) = 0$  genau dann gilt, wenn die Kante  $(u, v)$  nicht existiert.

Unter diesen Voraussetzungen können wir einen (gerichteten) Graphen  $G = (V, E)$  mittels einer **Adjazenzmatrix**  $A = (a_{u,v})_{u,v \in V}$ , einer  $n \times n$ -Matrix über  $X$  mit  $a_{u,v} = w(u, v)$ , darstellen.

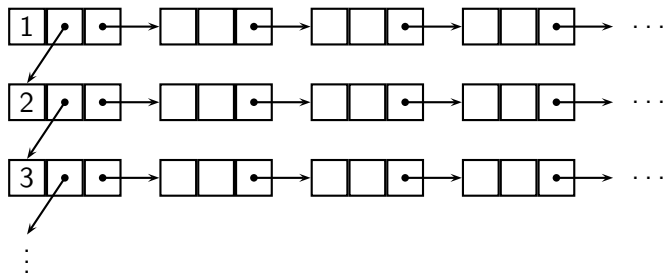
Um diese Matrix zu speichern benötigen wir  $|V|^2$  Einträge für Elemente aus  $X$ . Hat jeder Eintrag eine konstante maximale Größe, ergibt sich der Gesamt Speicherbedarf als  $O(|V|^2)$ , und ist somit unabhängig von der Anzahl der Kanten.

# Die Adjazenzmatrix


$$\begin{bmatrix} 0 & 1 & 0 & 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

# Die Adjazenzlisten

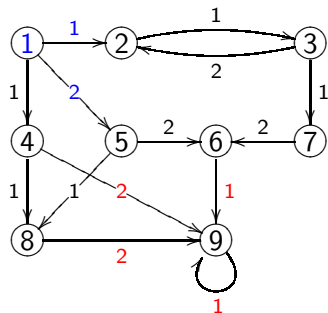
Eine Alternative besteht darin, für jeden Knoten eine Liste von adjazenten Kanten zu halten, die so genannten **Adjazenzlisten**.



Dabei enthält jeder Eintrag für jeden adjazenten Knoten drei Komponenten: Den Zielknoten, das Gewicht und einen Zeiger auf den nächsten adjazenten Knoten.

Insgesamt ergibt sich somit ein Speicherbedarf von  $O(|V| + |E|)$ .

# Die Adjazenzlisten



1 : (2, 1), (4, 1), (5, 2)

2 : (3, 1)

3 : (2, 2), (7, 1)

4 : (8, 1), (9, 2)

5 : (6, 2), (8, 1)

6 : (9, 1)

7 : (6, 2)

8 : (9, 2)

9 : (9, 1)

## Doppelt verkettete Adjazenzlisten

Statt einfach verketteter Adjazenzlisten, können selbstverständlich auch doppelt verkettete Listen verwendet werden. Außerdem kann für jede Kante noch ein zusätzlicher Zeiger auf den entsprechenden Knoten angelegt werden.

Ein Knoteneintrag besteht somit aus

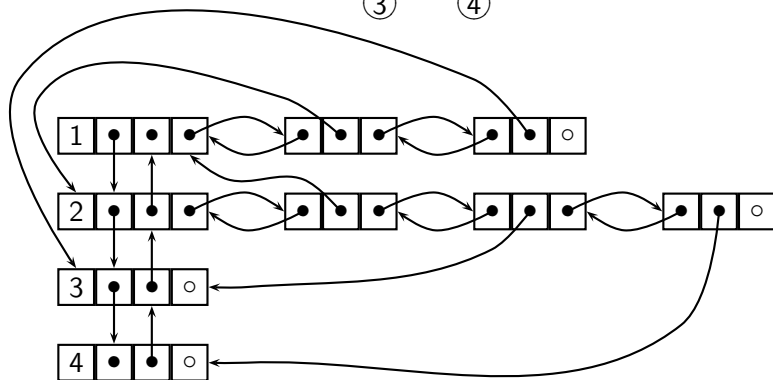
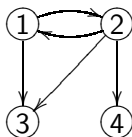
- 1 der Nummer des Knotens,
- 2 zwei Zeigern auf die beiden Nachbarn in der Knotenliste und
- 3 einem Zeiger auf den Beginn der Liste der Inzidenten Kanten.

Ein Kanteneintrag besteht aus

- 1 dem Gewicht der Kante,
- 2 zwei Zeigern auf die beiden Nachbarn in der Kantenliste,
- 3 einem Zeiger auf den Knoteneintrag des Zieles und

Wie bei den einfach verketteten Adjazenzlisten, ist der Speicherbedarf  $O(|V| + |E|)$ .

# Doppelt verkettete Adjazenzlisten



# Bemerkungen

In allen vorgestellten Darstellungen, können statt der Gewichte natürlich auch Zeiger auf Strukturen verwaltet werden, die zusätzliche Attribute der Kanten enthalten.

Die Adjazenzlistendarstellungen ermöglichen direkt die Verwendung von parallelen Kanten, indem einfach für jede Kante ein Eintrag in der Adjazenzliste gemacht wird.

In der Adjazenzmatrix können parallele Kanten so hinzugefügt werden, dass jeder Matrixeintrag ein Zeiger auf eine Liste von Kanten zwischen den beiden jeweiligen Knoten ist.

# Übersicht

	Adjazenzmatrix	Adjazenzliste
Vorteile	<ul style="list-style-type: none"><li>• Schneller Zugriff auf Kanten in <math>O(1)</math></li></ul>	<ul style="list-style-type: none"><li>• Niedriger Speicherbedarf bei „wenig“ Kanten</li><li>• Dynamischer</li></ul>
Nachteile	<ul style="list-style-type: none"><li>• Hoher Speicherbedarf <math>O( V ^2)</math></li></ul>	<ul style="list-style-type: none"><li>• aufwändigere Verwaltung</li><li>• Zugriff auf Kanten langsamer</li></ul>