UNIVERSITÄT
LEIPZIG

Algorithmen und Datenstrukturen II

Vorlesung yks1

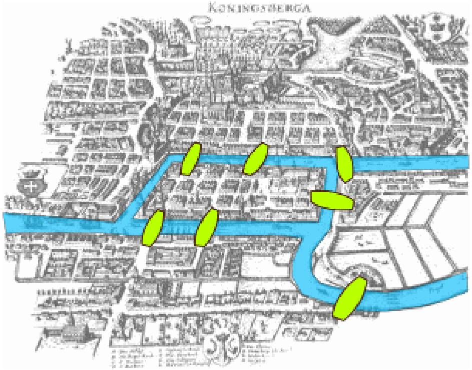
Leipzig, 02.04.2024
Peter F. Stadler & Thomas Gatter & Ronny Lorenz

GRAPHEN

Graphen - Themenübersicht

1. **Ungerichtete gewichtete Graphen**
   Grundlegende Definitionen, minimale Spannbäume
2. **Gerichtete Graphen**
   Definitionen, Speicherung, topologische Sortierung, transitive Hülle, starke Zusammenhangskomponenten
3. **Gerichtete gewichtete Graphen**
   Kürzeste Pfade, Flussnetzwerke

Königsberger Brückenproblem (Euler, 1736)



Originalartikel: *http://www.math.dartmouth.edu*

Hinweise

Achtung: Die Nomenklatur in diesem Gebiet ist in der Literatur nicht perfekt standardisiert. Vergleichen Sie die Definitionen, wenn Sie alternative Quellen nutzen!

Siehe auch Wikipedia

- Graph (Graphentheorie) *(Link)*
- Königsberger Brückenproblem *(Link)*

Ungerichteter Graph

Definition

Ein Tupel (V, E) heißt ***(ungerichteter) Graph***, genau dann wenn

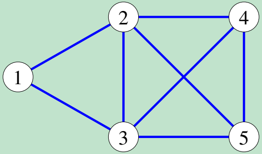
- V eine endliche Menge und
- E eine Menge ungeordneter Paare von Elementen in V ist.

V heißt **Knotenmenge**, die Elemente von V heißen **Knoten**.
E heißt **Kantenmenge**, die Elemente von E heißen **Kanten**.

Beispiel ungerichteter Graph

Beispiel

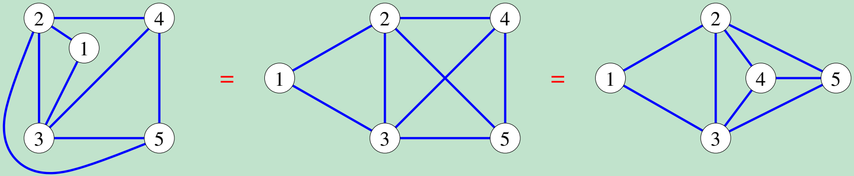
$V = \{1, 2, 3, 4, 5\}$
$E = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{3, 5\}, \{4, 5\}\}$



Beispiel ungerichteter Graph

Beispiel

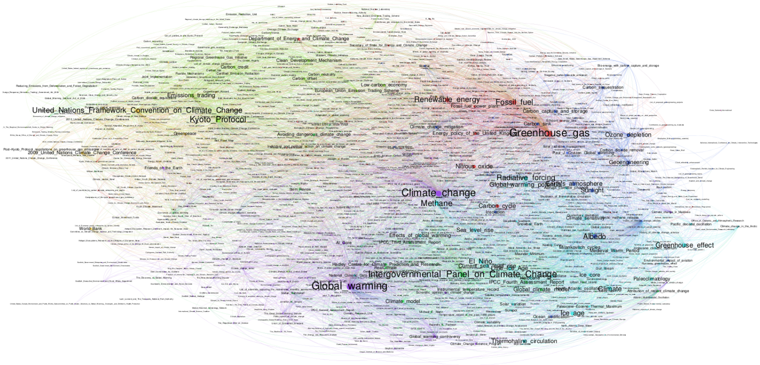
$V = \{1, 2, 3, 4, 5\}$
$E = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{3, 5\}, \{4, 5\}\}$



Graphen: Beispiele realer Systeme

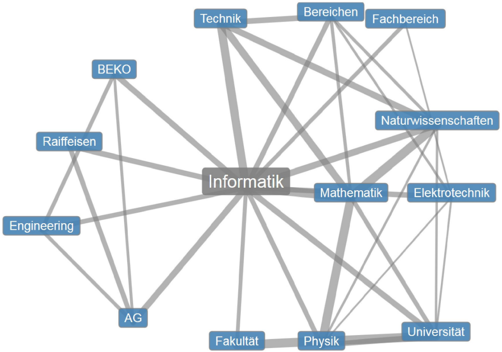
System	Knoten	Kanten
Internet	Router	Datenleitungen
WWW	Webseiten/-dokumente	Hyperlinks
Gesellschaft	Personen	soziale Kontakte
Sprache	Wörter	gemeinsames Auftreten
Molekül	Atome	chem. Bindungen

Netzwerk von Artikeln über Klimawandel (Okt 2012)



Quelle: *http://www.emapsproject.com*

Wortschatzgraph

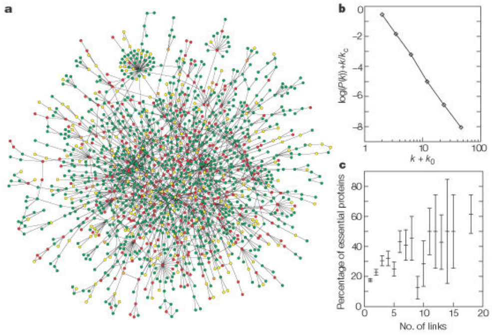


Quelle: *http://corpora.uni-leipzig.de*

Social Network (Facebook, 2010)

Quelle: *http://blog.revolutionanalytics.com*

Graph von Protein-Wechselwirkungen (Hefe)



Jeong, Mason, Barabási & Oltvai, *Nature* 2001.

Teilgraphen

Definition

Seien $G = (V, E)$ und $G' = (V', E')$ Graphen.

- G' heißt **Teilgraph** von G, wenn $V' \subseteq V$ und $E' \subseteq E$ ist.
- G' heißt **aufspannender Teilgraph** von G, wenn G' Teilgraph von G mit $V' = V$ ist (G' enthält dieselben Knoten wie G).
- G' heißt **induzierter Teilgraph** von G, wenn G' Teilgraph von G ist und für alle $e = \{a, b\} \in E$ gilt: $\{a, b\} \subseteq V' \Rightarrow e \in E'$. (Alle Kanten aus G, deren zwei Knoten in G' liegen, sind auch in G' enthalten.)

Pfade, Zyklen, Zusammenhang

Definition

Sei $G = (V, E)$ ein Graph, $\ell \in \mathbb{N}$ und $k = (v_0, v_1, \dots, v_\ell) \in V^{\ell+1}$.

- k heißt **Weg** der Länge ℓ, wenn für alle $i \in \{1, \dots, \ell\}$ gilt: $\{v_{i-1}, v_i\} \in E$
- k heißt **Pfad** der Länge ℓ, wenn k ein Weg ist und für alle $i, j \in \{0, \dots, \ell\}$ mit $i \neq j$ gilt: $v_i \neq v_j$.
- k heißt **Zyklus** (oder **Kreis**) der Länge ℓ, wenn $(v_1, \dots, v_\ell)$ ein Pfad der Länge $\ell - 1$ ist, $v_0 = v_\ell$ und $\{v_0, v_1\} \in E$.
- G heißt **zusammenhängend**, wenn für alle $x, y \in V$ ein Pfad zwischen x und y existiert.

Pfade, Zyklen, Zusammenhang

Achtung!
Im Wikipedia-Beitrag zu ***Weg (Graphentheorie)*** wird *Weg* wie unser Begriff *Pfad* verwendet!

Wälder und Bäume

Definition

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

- G heißt **Wald** (oder **zyklenfrei**), wenn kein Weg in G ein Zyklus ist.
- G heißt **Baum**, wenn G ein Wald ist und G zusammenhängend ist.

Wälder und Bäume

Theorem

Ist G ein Baum, so hat G genau $|V| - 1$ Kanten.

Beweis - mittels Induktion

Basisfall: Ein Baum mit einem Knoten hat keine Kanten.
Induktionsschritt. Sei $|V| \geq 1$. Da G zusammenhängend ist und keinen Kreis enthält muß es einen Knoten x mit Grad 1 geben. Wenn man den und seine inzidente Kante entfernt, bleibt ein zusammenhängender, kreisfreier Graph, also ein Baum, mit $|V| - 1$ Knoten übrig. Der hat nach Induktionsannahme $|V| - 2$ Kanten. Also hat G genau $|V| - 1$ Kanten $\square$

Wälder und Bäume

Theorem

Ist G zusammenhängend, so hat G einen aufspannenden Teilgraphen T, so dass T ein Baum ist. T heißt dann **Spannbaum** von G.

Wie kann man das beweisen?
Hinweise: ebenfalls mittels Induktion nach der Zahl der Knoten in G.

Wie konstruieren wir (irgend)einen Spannbaum für einen zusammenhängenden Graphen G?

Gewichteter Graph

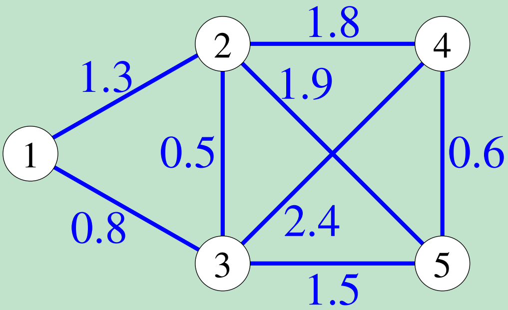
Definition

Sei (V, E) ein Graph und $w : E \to \mathbb{R}$.

- Das Tripel $G = (V, E, w)$ heißt **gewichteter** (oder **kantenbewerteter**) Graph.
- $w(e)$ heißt **Gewicht** (oder Länge) der Kante $e \in E$.

Gewichteter Graph

Beispiel



Minimaler Spannbaum

Problem

Gegeben: Gewichteter zusammenhängender Graph $G = (V, E, w)$.
Gesucht: Spannbaum $T = (V, F)$ mit *minimaler Kantensumme*. Wähle also die Kantenmenge $F \subseteq E$ so, dass

$$\sum_{e \in F} w(e)$$

möglichst klein wird.

Kruskal-Algorithmus (1956)

Algorithmus (Minimaler Spannbaum)

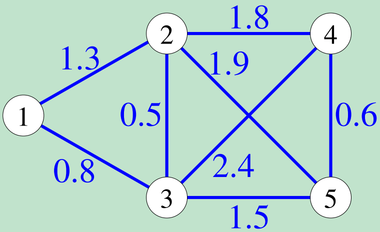
input : Graph $G = (V, E, w)$
output : Minimaler Spannbaum $T = (V, F)$ von G

$F \leftarrow \emptyset$;   `// initialiere F als leere Menge`
$L \leftarrow sort(E)$;   `// erzeuge Liste` L `der nach Gewicht sortierten Kanten in` E

while *(EMPTY(L) == False)* **do**
  entferne Kante $e = \{u, v\}$ mit kleinstem Gewicht aus L
  if *($T = (V, F)$ enthält keinen Pfad zwischen u und v)* **then**
    $F = F \cup \{e\}$;
  else
    tue nichts

Kruskal-Algorithmus

Beispiel eines Laufes

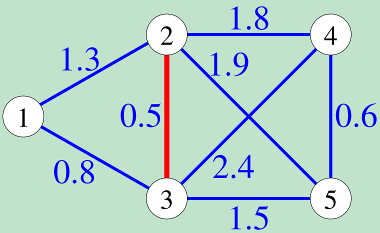


$L = [\{2,3\},\{4,5\},\{1,3\},\{1,2\},\{3,5\},\{2,4\},\{2,5\},\{3,4\}]$

$F = \{\}$

Kruskal-Algorithmus

Beispiel eines Laufes

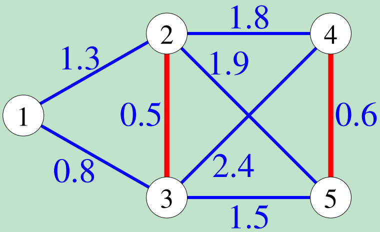


$L = [\{4,5\},\{1,3\},\{1,2\},\{3,5\},\{2,4\},\{2,5\},\{3,4\}]$

$F = \{\{2,3\}\}$

Kruskal-Algorithmus

Beispiel eines Laufes

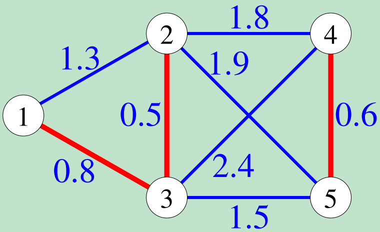


$L = [\{1,3\},\{1,2\},\{3,5\},\{2,4\},\{2,5\},\{3,4\}]$

$F = \{\{2,3\},\{4,5\}\}$

Kruskal-Algorithmus

Beispiel eines Laufes

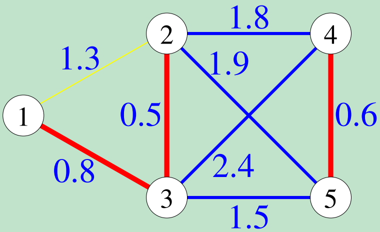


$$L = [\{1,2\}, \{3,5\}, \{2,4\}, \{2,5\}, \{3,4\}]$$

$$F = \{\{2,3\}, \{4,5\}, \{1,3\}\}$$

Kruskal-Algorithmus

Beispiel eines Laufes

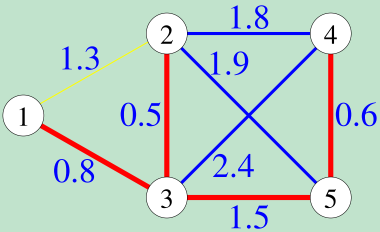


$L = [\{3,5\},\{2,4\},\{2,5\},\{3,4\}]$

$F = \{\{2,3\},\{4,5\},\{1,3\}\}$

Kruskal-Algorithmus

Beispiel eines Laufes

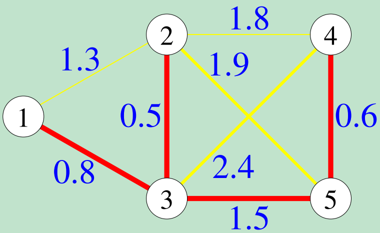


$L = [\{2,4\},\{2,5\},\{3,4\}]$

$F = \{\{2,3\},\{4,5\},\{1,3\},\{3,5\}\}$

Kruskal-Algorithmus

Beispiel eines Laufes



$L = []$

$F = \{\{2,3\},\{4,5\},\{1,3\},\{3,5\}\}$

Anmerkungen zum Kruskal-Algorithmus

Ein gutes Video zum Algorithmus ist auf studyflix *(Link)* zu finden. (Die Verweise im Video auf “Greedy Algorithmen” können Sie erst mal ignorieren)

Wie berechnen sie einen *maximalen* Spannbaum?

- **Korrektheit:** Findet der Kruskal-Algorithmus garantiert einen minimalen Spannbaum?
  $\rightarrow$ Ja, sehen wir später bei Greedy-Algorithmen.
- **Laufzeit-Komplexität:** $O(|E| \log |E|)$ $[= O(|E| \log |V|)]$. *Anmerkung:* dazu brauchen wir zusätzlich einen effizienten Test, d.h. in $O(\log |V|)$, ob zwei Knoten jeweils für Kanten F zusammenhängen: “Union-Find-Datenstruktur”.

GERICHTETE GRAPHEN

Gerichteter Graph

Definition

Ein Tupel (V, E) heißt **gerichteter Graph** (Digraph), wenn V eine endliche Menge und E eine Menge geordneter Paare von Elementen in V ist.

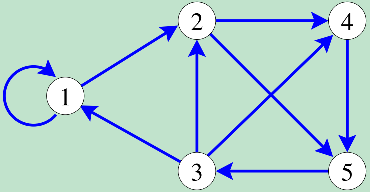
- V heißt **Knotenmenge**, die Elemente von V heißen *Knoten*.
- E heißt **Kantenmenge**, die Elemente von E heißen *Kanten*.
- Eine Kante (v, v) heißt **Schleife**.

Gerichteter Graph

Beispiel

$V = \{1, 2, 3, 4, 5\}$,

$E = \{(1, 1), (1, 2), (2, 4), (2, 5), (3, 1), (3, 2), (3, 4), (4, 5), (5, 3)\}$



Vorgänger, Nachfolger, Grad

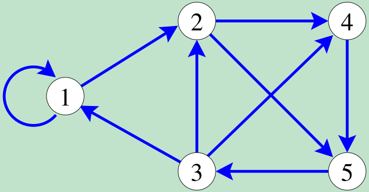
Definition

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

- $u \in V$ heißt **Vorgänger** von v, wenn $(u, v) \in E$.
- $pred(v) := \{u \in V | (u, v) \in E\}$ ist die Menge der Vorgänger von v.
- Der **Eingangsgrad** von v ist $\mathrm{eg}(v) = |\mathrm{pred}(v)|$
- $w \in V$ heißt **Nachfolger** von v, wenn $(v, w) \in E$.
- $\mathrm{succ}(v) := \{w \in V | (v, w) \in E\}$ ist die Menge der Nachfolger von v.
- Der **Ausgangsgrad** von v ist $\mathrm{ag}(v) = |\mathrm{succ}(v)|$

Vorgänger, Nachfolger, Grad

Beispiel



$\mathrm{eg}(3) = 1$, $\mathrm{pred}(3) = \{5\}$
$\mathrm{ag}(3) = 3$, $\mathrm{succ}(3) = \{1, 2, 4\}$

SPEICHERUNG

Speicherung von Graphen: Adjazenzmatrix

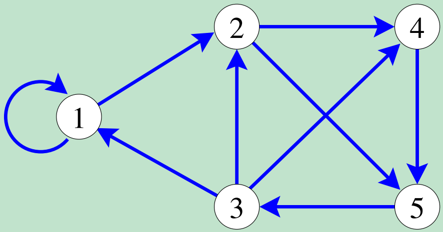
Definition

Ein Graph $G = (V, E)$ mit $|V| = n$ kann in einer Boole'schen $n \times n$-Matrix $A = (a_{ij})$ gespeichert werden, wobei gilt:

$$a_{ij} = \begin{cases} 1 & \text{falls } (i, j) \in E \\ 0 & \text{sonst} \end{cases}$$

Speicherung von Graphen: Adjazenzmatrix

Beispiel



$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Speicherung von Graphen: Adjazenzmatrix

Speicherplatzbedarf $O(n^2)$

- 1 Bit pro Position (statt Knoten/Kantennummern)
- unabhängig von Kantenmenge
- für ungerichtete Graphen ergibt sich symmetrische Belegung (Halbierung des Speicherbedarfs möglich)

Speicherung von Graphen: Listen

- Speicherung von Graphen als Liste von Zahlen (Array oder verkettete Liste)
- Knoten werden von 1 bis n durchnummeriert. Kanten als Paare von Knoten

Kantenliste

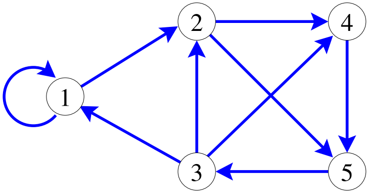
- Liste: Knotenzahl, Kantenzahl, Liste von Kanten (je als 2 Zahlen)
- Speicherbedarf: $2 + 2m$ (m = Anzahl Kanten)

Knotenliste

- Liste: Knotenzahl, Kantenzahl, Liste von Knoteninformationen
- Knoteninformation: Ausgangsgrad und die jeweiligen Nachfolger $\mathrm{ag}(v), s_1, s_2, \ldots, s_{\mathrm{ag}(v)}$
- Speicherbedarf: $2 + n + m$

Speicherung von Graphen: Adjazenzlisten

- verkettete Liste der n Knoten (oder Array-Realisierung)
- pro Knoten: verkettete Liste der Nachfolger (repräsentiert die von dem Knoten ausgehenden Kanten)
- Speicherbedarf: $n + m$ Listenelemente



$$
\begin{array}{l}
1 \rightarrow 1 \rightarrow 2 \\
\downarrow \\
2 \rightarrow 4 \rightarrow 5 \\
\downarrow \\
3 \rightarrow 1 \rightarrow 2 \rightarrow 4 \\
\downarrow \\
4 \rightarrow 5 \\
\downarrow \\
5 \rightarrow 3
\end{array}
$$

Vergleich der Speichermöglichkeiten

Komplexitätsvergleich

Operation	Adj.-matrix	Kantenliste	Knotenliste	Adjazenzliste
Einfügen Kante	$O(1)$	$O(1)$	$O(n+m)$	$O(1)/O(n)$
Löschen Kante	$O(1)$	$O(m)$	$O(n+m)$	$O(n)$
Einfügen Knoten	$O(n^2)$	$O(1)$	$O(1)$	$O(1)$
Löschen Knoten	$O(n^2)$	$O(m)$	$O(n+m)$	$O(n+m)$

– Löschen eines Knotens löscht auch zugehörige Kanten

– Änderungsaufwand abhängig von Realisierung der Adjazenzmatrix und Adjazenzliste

Vergleich der Speichermöglichkeiten

Welche Repräsentation geeigneter ist, hängt vom betrachteten Problem bzw. der Fragestellung ab.

- Gibt es eine Kante von a nach b?
  $\rightarrow$ Speicherung in Matrix
- Durchsuchen von Knoten in durch die Nachbarschaft gegebener Reihenfolge.
  $\rightarrow$ Speicherung in Listen

Mehr Informationen und interaktive Übungen zu den verschiedenen Darstellungs- und Speichervarianten von Graphen gibt es hier: *(Link)*

Kantenfolgen, Pfade, Zyklen

Definition

Sei $G = (V, E)$ gerichteter Graph, $\ell \in \mathbb{N} \cup \{0\}$ und $k = (v_0, v_1, \ldots, v_\ell) \in V^{\ell+1}$.

- k heißt **Kantenfolge** (oder **Weg**) der Länge ℓ von v_0 nach v_ℓ, wenn für alle $i \in \{1, \ldots, \ell\}$ gilt: $(v_{i-1}, v_i) \in E$
- $v_1, \ldots, v_{\ell-1}$ sind die **inneren** Knoten von k. Ist $v_0 = v_\ell$, so ist die Kantenfolge *geschlossen*.
- k heißt **Kantenzug**, wenn k Kantenfolge ist und für alle $i, j \in \{0, \ldots, \ell - 1\}$ mit $i \neq j$ gilt: $(v_i, v_{i+1}) \neq (v_j, v_{j+1})$.

Kantenfolgen, Pfade, Zyklen

Definition

Sei $G = (V, E)$ gerichteter Graph, $\ell \in \mathbb{N} \cup \{0\}$ und $k = (v_0, v_1, \ldots, v_\ell) \in V^{\ell+1}$.

- k heißt **Pfad** der Länge ℓ, wenn k eine Kantenfolge ist und für alle $i, j \in \{0, \ldots, \ell\}$ mit $i \neq j$ gilt: $v_i \neq v_j$.
- k heißt **Zyklus** (oder **Kreis**), wenn $(v_1, \ldots, v_\ell)$ ein Pfad der Länge $\ell - 1$ ist, $v_0 = v_\ell$, und $(v_0, v_1) \in E$.
- k heißt **Hamiltonscher Zyklus**, wenn k Zyklus ist und $\ell = |V|$.