Fortgeschrittene Netzwerk- und Graph-Algorithmen

Dr. Hanjo Täubig

Lehrstuhl für Effiziente Algorithmen
(Prof. Dr. Ernst W. Mayr)
Institut für Informatik
Technische Universität München

Wintersemester 2009/10

Übersicht

1. Grundlagen

Vorlesungsdaten

- Modul: IN2158
- Bereich:
  Informatik III (Theoretische Informatik)
- Semesterwochenstunden:
  4 SWS Vorlesung + 2 SWS Übung
- ECTS: 8 Punkte
- Vorlesungszeiten:
  Mittwoch 10:15 - 11:45 Uhr (MI Hörsaal 2)
  Freitag 14:15 - 15:45 Uhr (MI Hörsaal 2)
- Übung:
  Freitag 12:15 - 13:45 Uhr (MI Praktikumsraum 03.09.034)

Dozent

- Dr. Hanjo Täubig
  (Lehrstuhl für Effiziente Algorithmen / Prof. Mayr)
- eMail:
  taeubig@in.tum.de
- Web:
  http://www14.in.tum.de/personen/taeubig/
- Telefon: 089 / 289-17740
- Raum: 03.09.039
- Sprechstunde: Mittwoch 13-14 Uhr
  (oder nach Vereinbarung)

Hinweis

Am Mittwoch, dem 4. November 2009 entfällt die Vorlesung aufgrund der Fachschaftsvollversammlung.

Voraussetzungen

- Voraussetzungen:
  Stoff des Informatik Grundstudiums
  - Diskrete Strukturen
  - Grundlagen: Algorithmen und Datenstrukturen
  - Einführung in die Theoretische Informatik

- vorteilhaft:
  Effiziente Algorithmen und Datenstrukturen I/II

Inhalt

- Inhalt:
  - Zentralitätsindizes
  - Dichte in (Teil-)Graphen
  - Alternative Algorithmen für Zusammenhangsprobleme
  - Clustering
  - Netzwerk-Statistik
  - Netzwerk-Vergleich
  - Spektrale Analyse
  - Robustheit

Literatur

- Die Vorlesung orientiert sich an folgendem Buch:

  U. Brandes, Th. Erlebach (Eds.):
  Network Analysis – Methodological Foundations

  Webzugriff:
  http://www.springerlink.com/openurl.asp?genre=issue&issn=0302-9743&volume=3418

  (Automatische Proxy-Konfiguration: http://pac.lrz-muenchen.de/)

- weitere Literatur:

  K. Mehlhorn, P. Sanders:
  Algorithms and Data Structures: The Basic Toolbox

  J. Bang-Jensen, G. Gutin:
  Digraphs: Theory, Algorithms and Applications

  V. Turau:
  Algorithmische Graphentheorie

Übersicht

1. Grundlagen
   - Netzwerke und Graphen
   - Graphrepräsentation

Netzwerke

- Netzwerk: Objekt bestehend aus
  - *Elementen* und
  - *Interaktionen* bzw. *Verbindungen* zwischen den Elementen

- Ein Netzwerk ist ein *informales Konzept*.
  - Wir haben dafür keine exakte Definition.
  - Die Elemente und insbesondere ihre Verbindungen können einen ganz unterschiedlichen Charakter haben.
  - Manchmal manifestieren sie sich in real existierenden Dingen, machmal sind sie nur gedacht (virtuell).

Beispiele für Netzwerke

Beispiele:

- Kommunikationsnetze (Internet, Telefonnetz),
- Verkehrsnetze (Straßennetz, Schienennetz, Flugnetz, Nahverkehrsnetz),
- Versorgungsnetzwerke (Strom, Wasser, Gas, Erdöl),
- wirtschaftliche Netzwerke (Geld- und Warenströme, Handel)
- biologische Netzwerke (Metabolische und Interaktionsnetzwerke),
- soziale Netzwerke (Communities),
- Publikationsnetzwerke

Erkenntnis

- Erkenntnis:
  Netze sind allgegenwärtig im täglichen Leben jedes Menschen. Wenn sie nicht richtig funktionieren, kann das weitreichende Folgen haben (z.B. Stromausfall bei Überlastung).

- Offensichtliche Folgerung:
  Es kann von großem Vorteil sein, wenn man Verfahren kennt, um Netzwerke zu analysieren, d.h. die Stärken und die Schwächen von Netzwerken festzustellen und Netzwerkvorgänge zu optimieren.

Graphen

- Graph: formales / abstraktes Objekt bestehend aus
  - einer Menge von *Knoten* (engl. nodes, vertices) und
  - einer Menge von *Kanten* (engl. edges, lines, links), die jeweils ein Paar von Knoten verbinden.
  - evt. einer Menge von Eigenschaften der Knoten und/oder Kanten
- Notation: $G = (V, E)$,
  manchmal auch $G = (V, E, w)$ im Fall gewichteter Graphen
- Anzahl der Knoten: $n = |V|$
  Anzahl der Kanten: $m = |E|$

Gerichtete und ungerichtete Graphen

- Unterscheidung: *ungerichtete / gerichtete* Kanten (bzw. Graphen):
  - ungerichtet: $E \subseteq \{\{v, w\} : v \in V, w \in V\}$
    (ungeordnetes Paar von Knoten bzw. 2-elementige Teilmenge)
  - gerichtet: $E \subseteq \{(v, w) : v \in V, w \in V\}$, also $E \subseteq V \times V$
    (geordnetes Paar von Knoten)
- Sind zwei Knoten v und w durch eine Kante e verbunden, dann nennt man
  - v und w *adjazent* bzw. *benachbart*
  - v und e *inzident* (ebenso w und e)
- Anzahl der Nachbarn eines Knotens v: *Grad* $\deg(v)$
  Bei gerichteten Graphen unterscheidet man
  - *Eingangsgrad*: $\deg^{-}(v) = |\{(w, v) \in E\}|$ und
  - *Ausgangsgrad*: $\deg^{+}(v) = |\{(v, w) \in E\}|$

Annahmen

Wir gehen meist von folgenden Annahmen aus:

- Der Graph (also die Anzahl der Knoten und Kanten) ist *endlich*.
- Der Graph ist *einfach*, d.h. E ist eine Menge und keine Multimenge (anderenfalls nennen wir G einen Multigraph).
- In den meisten Fällen gehen wir davon aus, dass der Graph keine *Schleifen* enthält (Kanten von v nach v).

Gewichtete Graphen

In Abhängigkeit von dem betrachteten Problem wird den Kanten und/oder Knoten oft eine Eigenschaft (z.B. eine Farbe oder ein numerischer Wert, das *Gewicht*) zugeordnet (evt. auch mehrere), z.B.

- Längen / Signallaufzeiten,
- Kosten,
- Kapazitäten / Bandbreite,
- Ähnlichkeiten,
- Verkehrsdichte, etc.

Wir nennen den Graphen dann

- *knotengewichtet* bzw.
- *kantengewichtet*

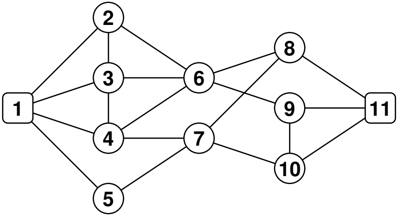
Beispiel: $w: \quad E \mapsto \mathbb{R}$

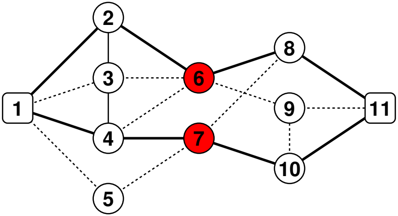
Schreibweise: $w(e)$ für das Gewicht einer Kante $e \in E$

Wege, Pfade und Kreise

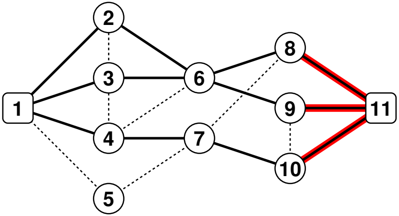
- Weg (engl. *walk*) in einem Graphen $G = (V, E)$: alternierende Folge von Knoten und Kanten $x_0, e_1, \ldots, e_k, x_k$, so dass
  - $\forall i \in [0, k] : x_i \in V$ und
  - $\forall i \in [1, k] : e_i = \{x_{i-1}, x_i\}$ bzw. $e_i = (x_{i-1}, x_i) \in E$.
- *Länge* eines Weges: Anzahl der enthaltenen Kanten
- Ein Weg ist ein Pfad, falls er (in sich) kantendisjunkt ist, falls also gilt: $e_i \neq e_j$ für $i \neq j$.
- Ein Pfad ist ein einfacher Pfad, falls er (in sich) knotendisjunkt ist, falls also gilt: $x_i \neq x_j$ für $i \neq j$.
- Ein Weg heißt *Kreis* (engl. *cycle*), falls $x_0 = x_k$.
- Eine andere Bedeutung der Begriffe *knoten-/kantendisjunkt* ergibt sich, wenn man mehrere Pfade zwischen gleichen Anfangs- und Endknoten betrachtet (siehe Abbildung).

Disjunkte *s*-*t*-Pfade





2 knotendisjunkte 1-11-Pfade



3 kantendisjunkte 1-11-Pfade

Graphrepräsentation

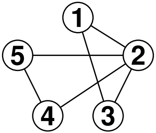
Wie kann man Graphen im Computer repräsentieren?
Vor- und Nachteile bei z.B. folgenden Fragen:

- Sind zwei gegebene Knoten v und w adjazent?
- Was sind die Nachbarn eines Knotens?
- Welche Knoten sind (direkte oder indirekte) Vorgänger bzw. Nachfolger eines Knotens v in einem gerichteten Graphen?
- Wie aufwendig ist es, einen Knoten bzw. eine Kante einzufügen oder zu löschen?

Graphrepräsentationen

- Kantenliste
- Adjazenzmatrix
- Inzidenzmatrix
- Adjazenzarray
- Adjazenzliste
- implizit

Kantenliste



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

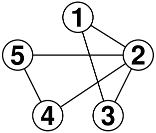
Vorteil:

- Speicherbedarf $2m + \mathcal{O}(n)$
- Einfügen und Löschen von Knoten und Kanten in $\mathcal{O}(1)$

Nachteil:

- Nachbarn nur in $\mathcal{O}(m)$ feststellbar

Adjazenzmatrix



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

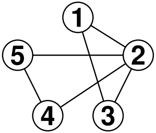
Vorteil:

- in $\mathcal{O}(1)$ feststellbar, ob zwei Knoten Nachbarn sind
- ebenso Einfügen und Löschen von Kanten

Nachteil:

- kostet $\mathcal{O}(n^2)$ Speicher, auch bei Graphen mit $o(n^2)$ Kanten
- Finden aller Nachbarn eines Knotens kostet $\mathcal{O}(n)$
- Hinzufügen neuer Knoten ist schwierig

Inzidenzmatrix

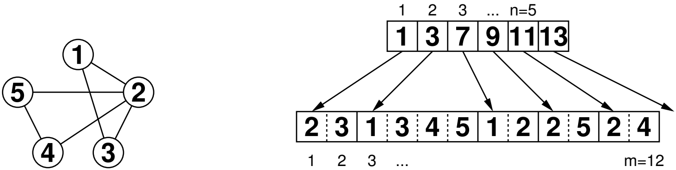


$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Nachteil:

- kostet $\mathcal{O}(mn)$ Speicher

Adjazenzarray



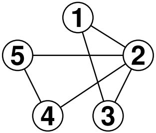
Vorteil:

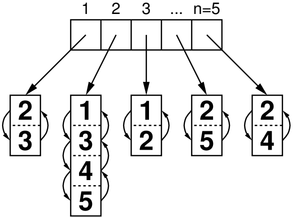
- Speicherbedarf:
  gerichtete Graphen: $n + m + \Theta(1)$
  (hier noch kompakter als Kantenliste mit $2m$)
  ungerichtete Graphen: $n + 2m + \Theta(1)$

Nachteil:

- Einfügen und Löschen von Kanten ist schwierig, deshalb nur für *statische* Graphen geeignet

Adjazenzliste





Unterschiedliche Varianten:
einfach/doppelt verkettet, linear/zirkulär

Vorteil:

- Einfügen und Löschen von Kanten in $\mathcal{O}(1)$
- mit unbounded arrays etwas cache-effizienter

Nachteil:

- Zeigerstrukturen verbrauchen relativ viel Platz und Zugriffszeit