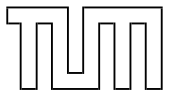
Fortgeschrittene Netzwerk- und Graph-Algorithmen

Dr. Hanjo Täubig

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

Wintersemester 2007/08



Übersicht

Vorlesungsdaten

- Modul: IN2158
- Bereich:
  Informatik III (Theoretische Informatik)
- Semesterwochenstunden:
  4 SWS Vorlesung + 2 SWS Übung
- ECTS: 8 Punkte
- Vorlesungszeiten:
  Montag 12:30 - 14:00 Uhr (MI Hörsaal 2)
  Donnerstag 15:30 - 17:00 Uhr (MI Hörsaal 2)
- Übung:
  Freitag 14:00 - 15:30 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: 289-17740
- Raum: 03.09.039

- Sprechstunde: Mittwoch 13-14 Uhr
  (oder nach Vereinbarung)

Hinweis

Am Montag, dem 3. Dezember 2007 entfällt die Vorlesung aufgrund einer Dienstreise.
Gleiches gilt für die Übung am Dienstag, dem 4. Dezember 2007.

Voraussetzungen

- Voraussetzungen:
  Stoff des Informatik Grundstudiums
  (Einführung in die Informatik, Diskrete Strukturen)
- 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/)

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).

- 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*
  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.

- Notation: $G = (V, E)$

- Anzahl der Knoten: $n = |V|$
  Anzahl der Kanten: $m = |E|$

Gerichtete und ungerichtete Graphen

- Wir unterscheiden ungerichtete und 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 ein 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: $w : E \mapsto \mathbb{R}$
Schreibweise: $w(e)$ für das Gewicht einer Kante $e \in E$

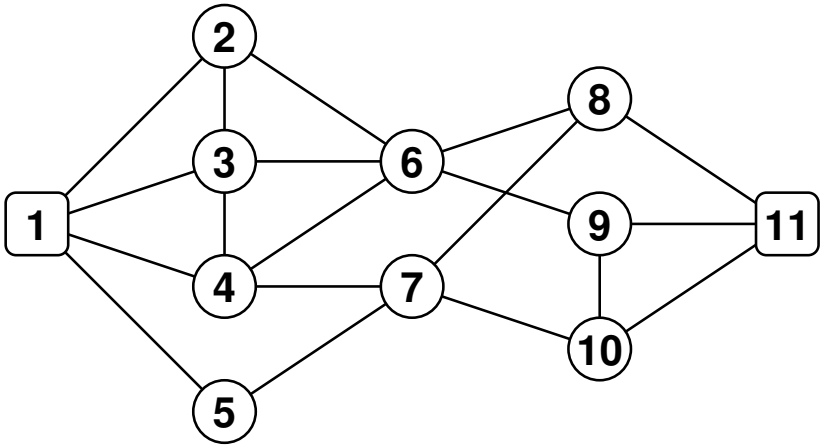
Wege, Pfade und Kreise

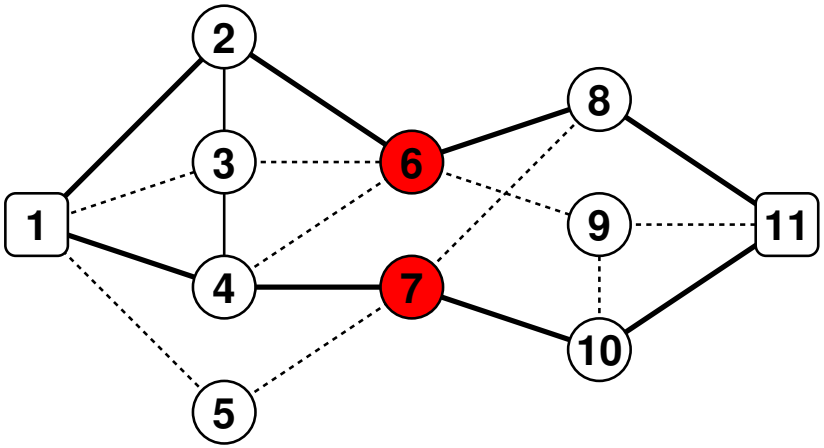
- Ein *Weg* (engl. walk) in einem Graphen $G = (V, E)$ ist eine alternierende Folge von Knoten und Kanten $x_0, e_1, \ldots, e_k, x_k$, so dass
  - $\forall i \in [0, k] : x_i \in V$ und
  - $\forall i \in [1, k] : e_i = \{x_{i-1}, x_i\}$ bzw. $e_i = (x_{i-1}, x_i) \in E$.
- Die *Länge* eines Weges ist die Anzahl der enthaltenen Kanten.

- Wir bezeichnen einen Weg als
  - *Pfad*, falls $e_i \neq e_j$ für $i \neq j$,
  - *einfachen Pfad*, falls $x_i \neq x_j$ für $i \neq j$.

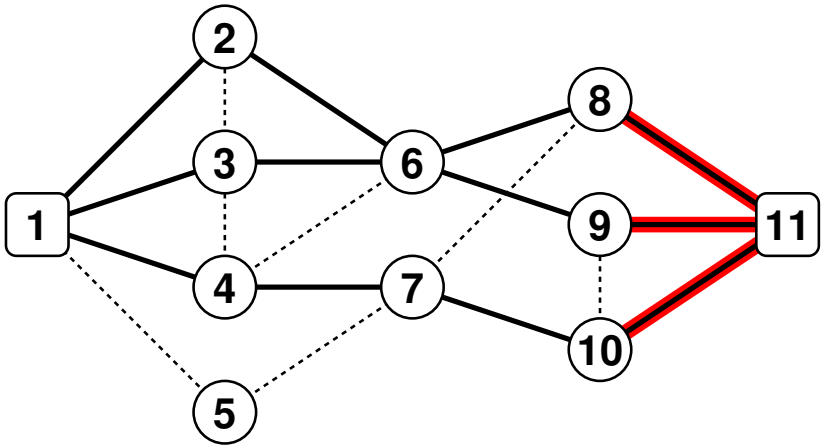
- Ein Weg heißt *Kreis*, falls $x_0 = x_k$.

Disjunkte s-t-Pfade





2 knotendisjunkte 1-11-Pfade



3 kantendisjunkte 1-11-Pfade

Graphrepräsentation

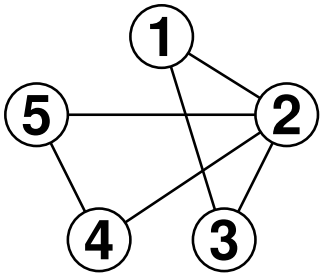
Wie kann man Graphen im Computer repräsentieren?
Vor- und Nachteile bei z.B. folgenden Fragen:

- 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 einzufügen oder zu löschen?

Graphrepräsentationen

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

Kantenliste



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

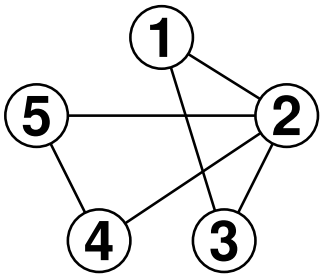
Vorteil:

- 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



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

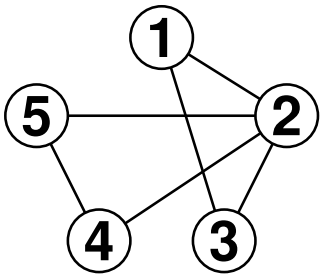
Vorteil:

- 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

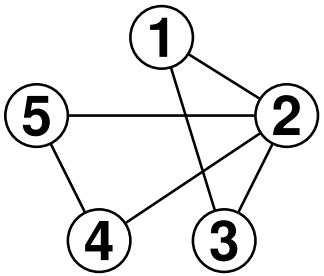


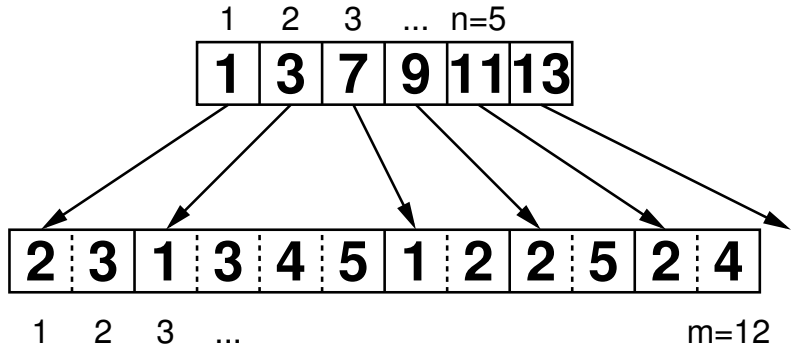
$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Nachteil:

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

Adjazenzarray





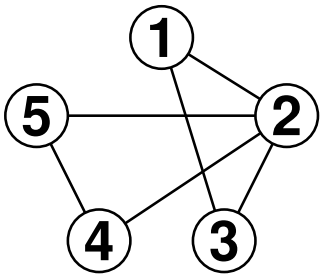
Vorteil:

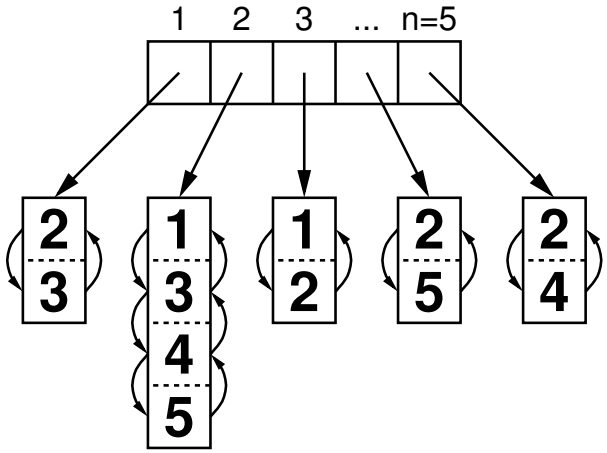
- Speicherbedarf $n + m + \Theta(1)$
  (besser Kantenliste mit $2m$)

Nachteil:

- Einfügen und Löschen von Kanten ist schwierig

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

Graphtraversierung

- Tiefensuche (depth first search, dfs)
  - Topologische Sortierung (bei DAGs)
  - Zweifachzusammenhangskomponenten
  - Starke Zusammenhangskomponenten

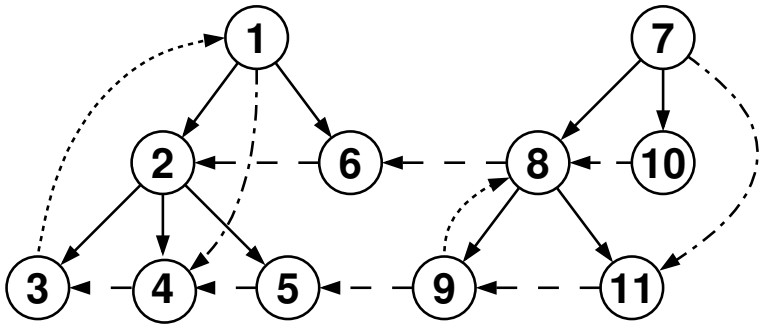
- Breitensuche (breadth first search, bfs)
  - schichtenweise Traversierung in aufsteigendem Abstand vom Ursprungsknoten

Traversierung / Kantentypen

DFS und BFS erzeugen Bäume bzw. Wälder und teilen die Kanten in folgende Klassen:

- Baumkanten (tree edges)
- Vorwärtskanten (forward edges)
- Rückwärtskanten (backward edges)
- Querkanten (cross edges)

Beispiel:



Blöcke und Artikulationsknoten

Definition

Ein Knoten v eines Graphen G heißt *Artikulationsknoten*, wenn sich die Anzahl der Zusammenhangskomponenten von G durch das Entfernen von v erhöht.

Definition

Die *Zweifachzusammenhangskomponenten* oder *Blöcke* eines Graphen sind die maximalen Teilgraphen, die 2-fach zusammenhängend sind.

Blöcke und DFS

Modifizierte DFS nach R. E. Tarjan:

- `num`$[v]$: DFS-Nummer von v
- `low`$[v]$: minimale Nummer `num`$[w]$ eines Knotens w, der von v aus über beliebig viele (≥ 0) Baumkanten abwärts, evt. gefolgt von einer einzigen Rückwärtskante erreicht werden kann

- `low`$[v]$: Minimum von
  - `num`$[v]$
  - `low`$[w]$, wobei w ein Kind von v im DFS-Baum ist
  - `num`$[w]$, wobei $\{v, w\}$ eine Rückwärtskante ist

Blöcke und DFS

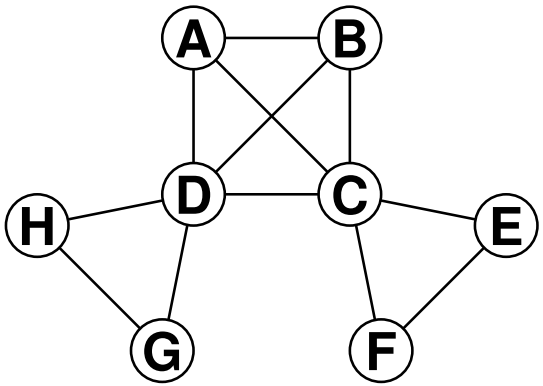
Lemma

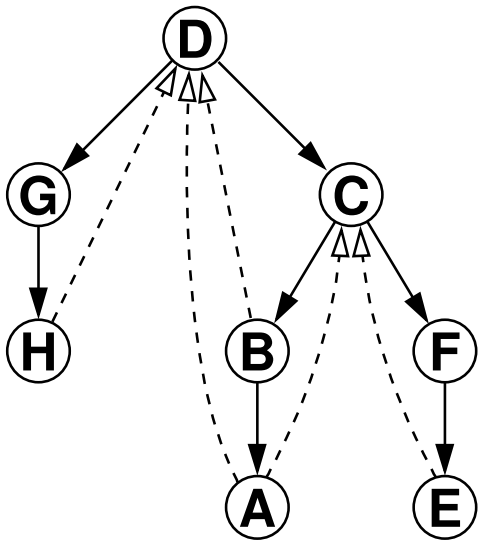
Sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph und T ein DFS-Baum in G.
Ein Knoten $a \in V$ ist genau dann ein Artikulationsknoten, wenn

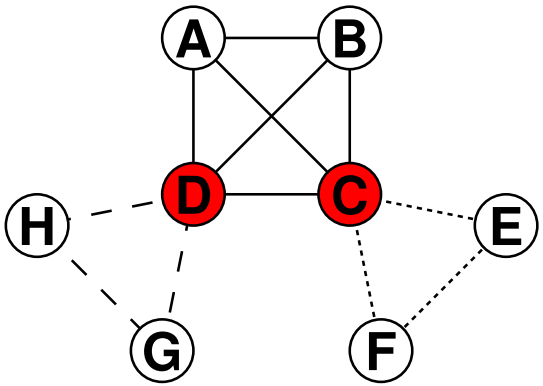
- *a die Wurzel von T ist und mindestens 2 Kinder hat, oder*
- *a nicht die Wurzel von T ist und es ein Kind b von a mit $low[b] \geq num[a]$ gibt.*

Blöcke und DFS

Die Kanten werden auf einem Stack gesammelt und nach der Erkennung eines Artikulationsknotens wird der gesamte Block abgepflückt.







Starke Zhk. und DFS

Modifizierte DFS nach R. E. Tarjan:

- `num[`v`]`: DFS-Nummer von v
- `low[`v`]`: minimale Nummer `num[`w`]` eines Knotens w, der von v aus über beliebig viele (≥ 0) Baumkanten abwärts, evt. gefolgt von einer einzigen Rückwärtskante oder einer Querkante zu einer ZHK, deren Wurzel echter Vorfahre von v ist, erreicht werden kann
- `low[`v`]`: Minimum von
  - `num[`v`]`
  - `low[`w`]`, wobei w ein Kind von v im DFS-Baum ist
  - `num[`w`]`, wobei $\{v, w\}$ eine Rückwärtskante ist
  - `num[`w`]`, wobei $\{v, w\}$ eine Querkante ist und die Wurzel der starken Zusammenhangskomponente von w ist Vorfahre von v
- Ein Knoten v ist genau dann Wurzel einer starken Zusammenhangskomponente, wenn `num[`v`]`=`low[`v`]`.

Starke Zusammenhangskomponenten