ADS: Algorithmen und Datenstrukturen 1

Teil XIII

Peter F. Stadler & Konstantin Klemm

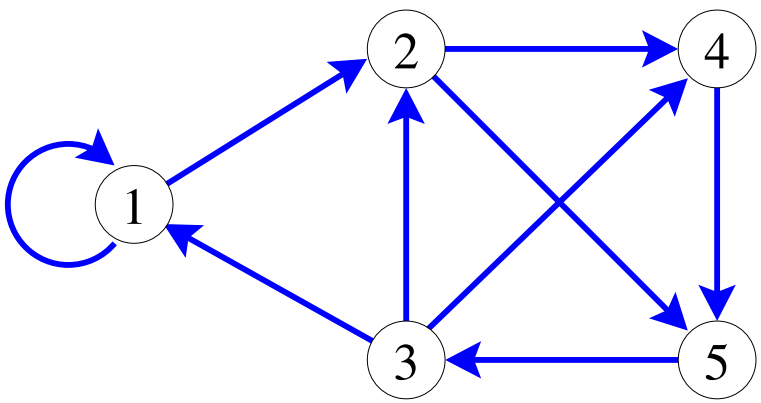
Bioinformatics Group, Dept. of Computer Science & Interdisciplinary Center for Bioinformatics, **University of Leipzig**

26. Januar 2011

Gerichteter Graph

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*.

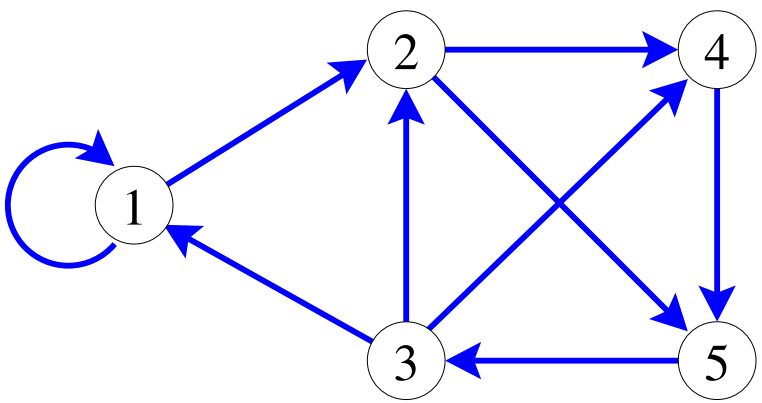
Beispiel: $V = \{1, 2, 3, 4, 5\}$, $E = \{(1, 1), (1, 2), (2, 4), (2, 5), (3, 1), (3, 2), (3, 4), (4, 5), (5, 3)\}$



Vorgänger, Nachfolger, Grad

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

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



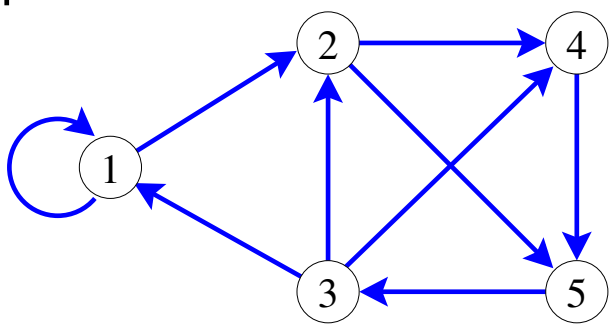
$\text{eg}(3) = 1$, $\text{pred}(3) = \{5\}$
$\text{ag}(3) = 3$, $\text{succ}(3) = \{1, 2, 4\}$

Speicherung von Graphen: Adjazenzmatrix

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

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

Beispiel:



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

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 in Listen

Knoten- und Kantenlisten

- Speicherung von Graphen als Liste von Zahlen (z.B. in Array oder verketteter Liste)
- Knoten werden von 1 bis n durchnummeriert; Kanten als Paare von Knoten

Kantenliste

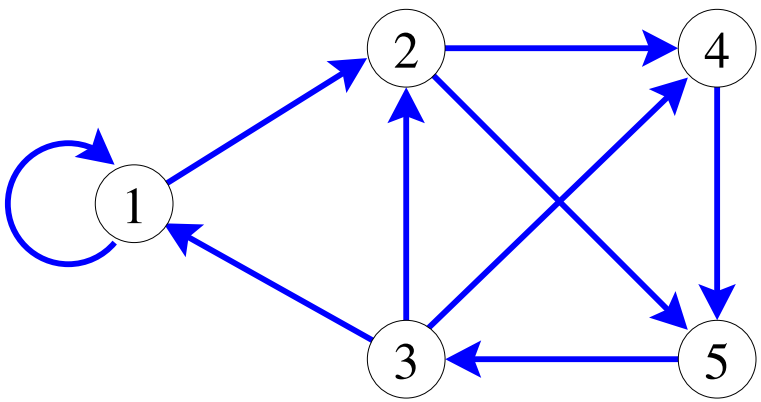
- Liste: Knotenzahl, Kantenzahl, Liste von Kanten (je als 2 Zahlen)
- Speicherbedarf: $2 + 2m$ (m = Anzahl Kanten)

Knotenliste

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

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



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

Speicherung von Graphen: Vergleich

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

Welche Repräsentation geeigneter ist, hängt vom Problem ab:

- Frage: Gibt es Kante von a nach b? $\rightarrow$ Matrix
- Durchsuchen von Knoten in durch Nachbarschaft gegebener Reihenfolge $\rightarrow$ Listen

Kantenfolgen, Pfade, Zyklen

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

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

Graph-Isomorphismus

Seien A und B Adjazenzmatrizen gerichteter Graphen, z.B.

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

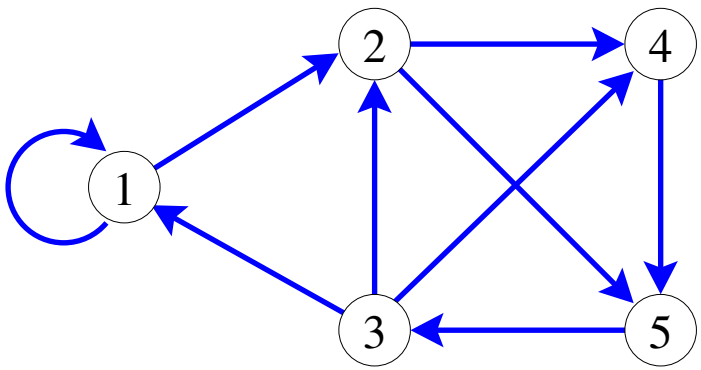
Frage: Sind A und B gleich bis auf Umbenennen der Knoten, also Permutation von Knotenindizes?

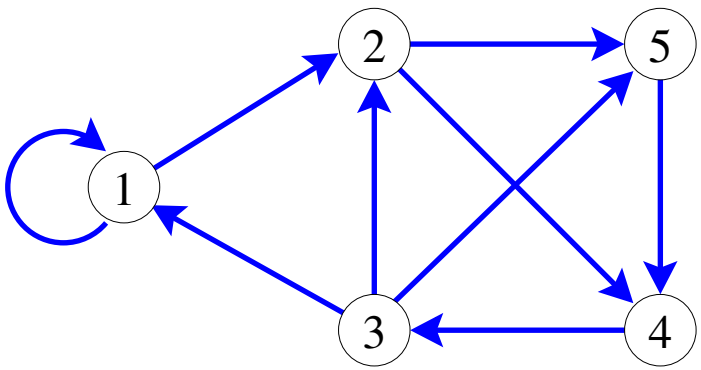
Graph-Isomorphismus

Seien A und B Adjazenzmatrizen gerichteter Graphen, z.B.

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

Frage: Sind A und B gleich bis auf Umbenennen der Knoten, also Permutation von Knotenindizes? Ja!





Graph-Isomorphismus

Seien G_A und G_B gerichtete Graphen mit Adjazenzmatrizen A und B. G_A heisst *isomorph* zu G_B, geschrieben $G_A \simeq G_B$,

$$:\Leftrightarrow$$

Es gibt eine Permutationsmatrix P so dass

$$A = PBP^{-1}$$

Eine quadratische Matrix P heisst Permutationsmatrix $:\Leftrightarrow$ in jeder Zeile und jeder Spalte von P steht genau eine 1, überall sonst null.

Graph-Isomorphismus: Effizienter Algorithmus?

- Entscheidungsproblem: Sind zwei gegebene Graphen isomorph oder nicht?
- Anwendungsbeispiel Chemie, große Molekülgraphen. Ist gefundene Verbindung identisch zu einer bereits bekannten?
- Praktische Fälle oft schnell lösbar, Vergleich der Gradsequenz, Distanzen, Spektrum (Eigenwerte) der Matrix, ...
- Offenes Problem: Gibt es ein allgemeines Verfahren mit Polynomialzeit, also existiert $\alpha > 0$ so daß

$$T(n) \in O(n^{\alpha}) \qquad ?$$

  Problemgröße $n =$ Anzahl Knoten des Graphen
- Beste bekannte obere Schranke

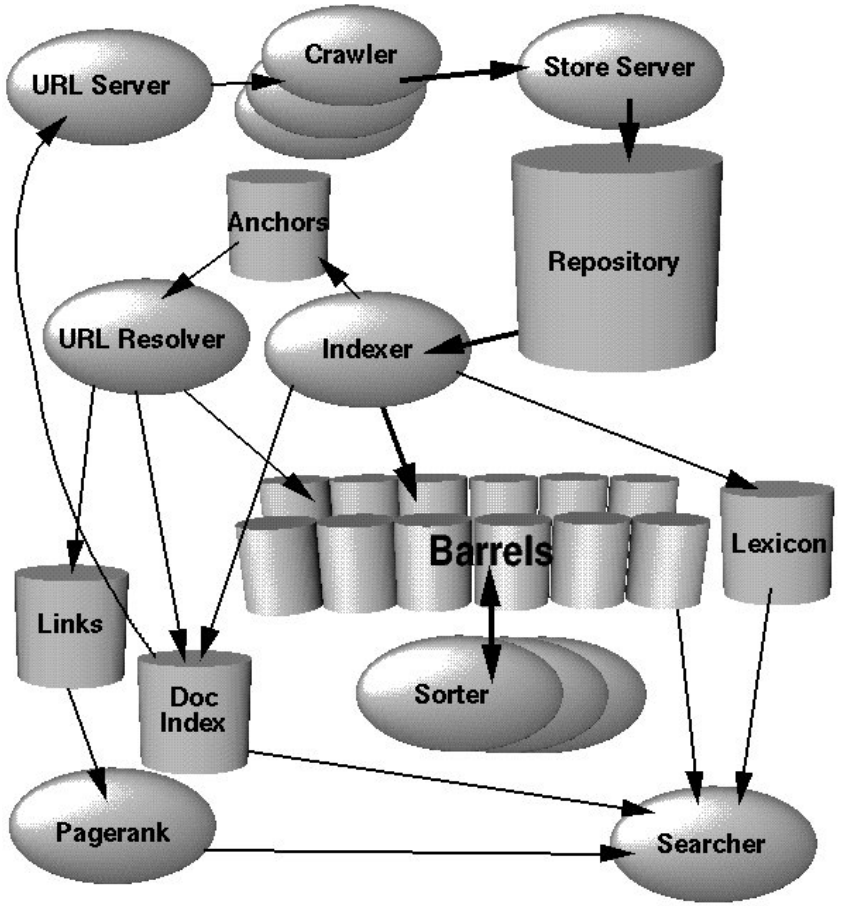
$$T(n) \in O(2^{\sqrt{n}})$$

Suche im WWW

Suchmaschine (web search engine)

- liefert Menge von Dokomenten (webpages), die mit einer Anfrage übereinstimmende Strings enthalten
- Problem: Relevanz der Dokumente?
- Menge mit passenden Dokumenten ist oft sehr gross. In welcher Reihenfolge sollen sie dem Nutzer empfohlen werden?
- Idee: Graphstruktur des WWW ausnutzen, um Rangfolge festzulegen

Google



Brin & Page, *The Anatomy of a Large-Scale Hypertextual Web Search Engine*

`http://infolab.stanford.edu/~backrub/google.html`

WWW als gerichteter Graph

- Gerichteter Graph $G = (V, E)$ = World Wide Web
- Knoten = Seite (webpage)
- gerichtete Kanten = Verweise ("links")

A = Adjazenzmatrix des WWW mit

$$a_{ij} = \begin{cases} 1 & \text{falls Seite } i \text{ auf Seite } j \text{ zeigt.} \\ 0 & \text{sonst} \end{cases}$$

PageRank

Wichtigkeit $W(j)$ einer Seite j ist proportional zu Anzahl und Wichtigkeit von Seiten, die auf j verweisen.

$$W(j) = \frac{1-d}{n} + d \sum_{i=1}^{n} \frac{a_{ij}}{ag(i)} W(i)$$

mit $d \in [0, 1]$, Brin & Page wählen $d = 0.85$

Modell "Random Surfer": Folgt meist (mit Wahrscheinlichkeit d) einem zufällig gleichverteilt gewählten Link auf der aktuellen Seite. Mit Wahrscheinlichkeit $1 - d$ wird jedoch eine Seite zufällig gleichverteilt aus allen n Seiten des WWW als nächste gewählt.

W ist die stationäre Verteilung dieses Prozesses, also W(j) = Wahrscheinlichkeit, dass sich Random Surfer bei Seite j befindet.

Mehr zu Graphen im Sommersemester ...

- in *Algorithmen und Datenstrukturen II*
- in *Graphentheorie* (Vorlesung) als Teil des Moduls *Biologische Netzwerke und Graphen*.
  freitags 15:15 Uhr, Raum 109, Härtelstr. 16-18