Kap. 6: Graphen

Professor Dr. Petra Mutzel
Lehrstuhl für Algorithm Engineering, LS11
Fakultät für Informatik, TU Dortmund

16. VO *2. Teil* DAP2 SS 2009 18. Juni 2009

Motivation

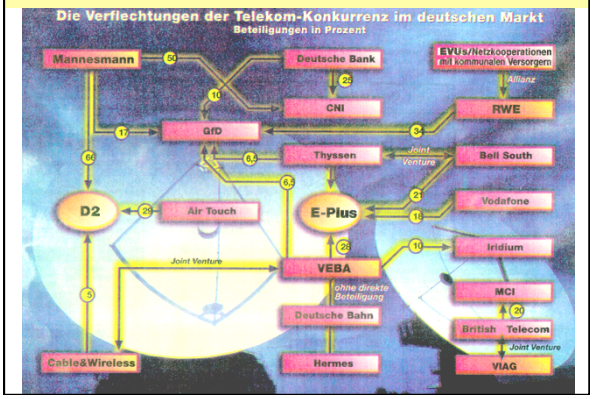
„Warum soll ich heute hier bleiben?"
Graphen sind wichtig und machen Spaß!

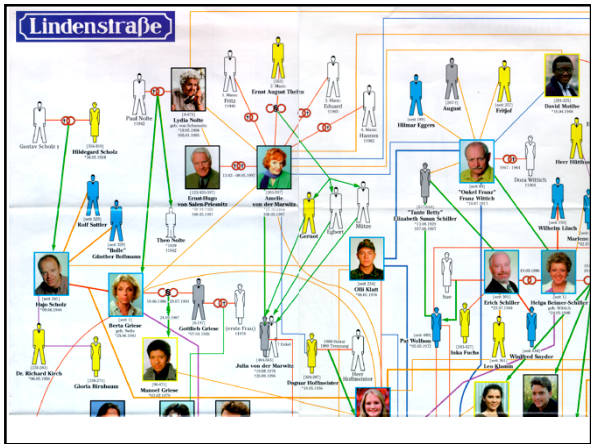
„Was gibt es heute Besonderes?"
schöne Bilder

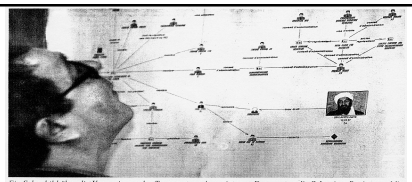
Überblick

- Motivation und Einführung
- Datenstrukturen
- Traversieren von Graphen:
  - Breitensuche (BFS)
  - Tiefensuche (DFS)

Graph: Objekte und Beziehungen zwischen den Objekten





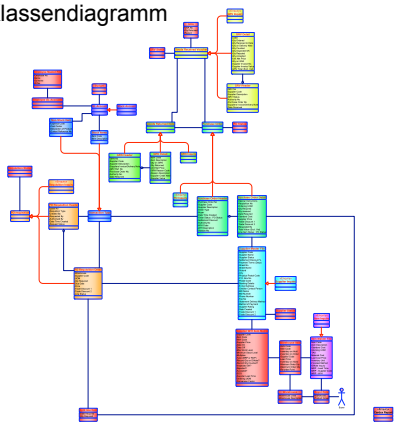


US-Präsident Bush gelingt Schlag gegen Bin Ladens Finanznetz

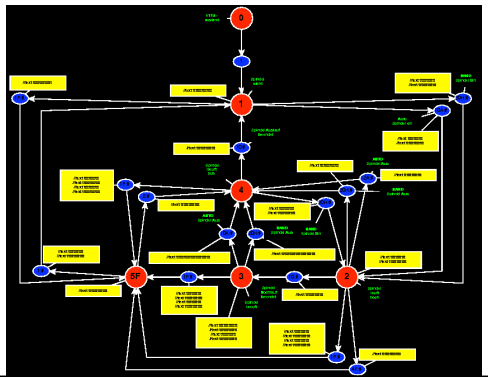
In der Schweiz und in Somalia wird gegen Verdächtige ermittelt

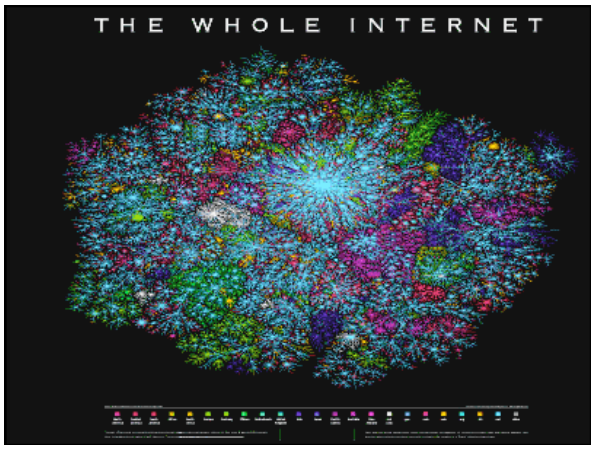
Kölner Stadtanzeiger
09.11.2001

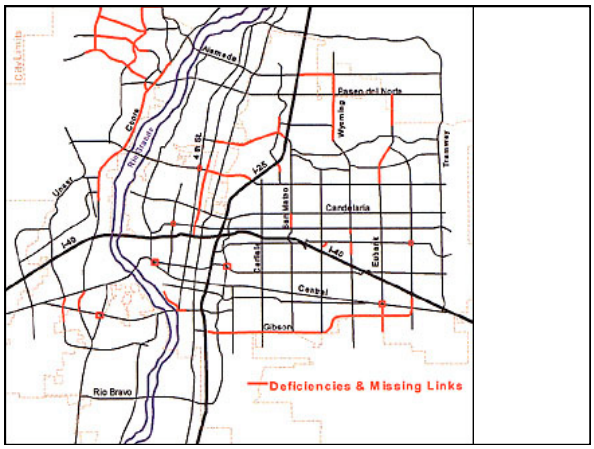
UML-Klassendiagramm



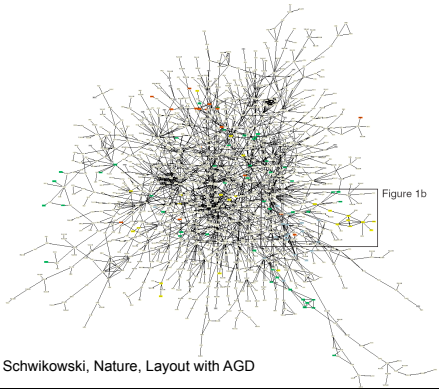
Endlicher Automat





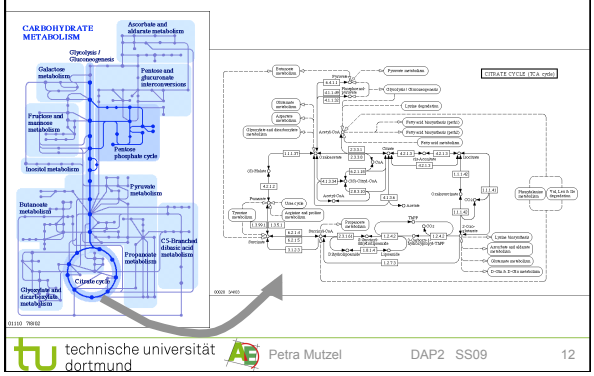


Protein-Protein Interaction Network

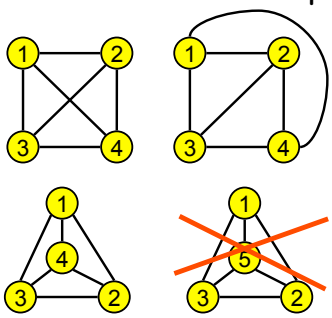


Benno Schwikowski, Nature, Layout with AGD

Metabolic Pathways



Verschiedene Zeichnungen des selben Graphen



Graph G=(V,E) mit V={1,2,3,4} und E gegeben durch E={(1,2),(1,3),(1,4), (2,3),(2,4),(3,4)}

Motivation

- Graphen modellieren diskrete Strukturen
- hilfreich zur Analyse und Optimierung

- Straßen-, Bahnnetze: kürzeste Wege
- Modellierung von Prozessen, z.B. Geschäftsprozesse, Betriebsabläufe
- Proteininteraktionsnetzwerke in der molekularen Biologie

Kap. 6.1 Definition (Graph)

Graph G=(V,E) besteht aus
- einer Menge V von Knoten
- einer (Multi-)menge E von Kanten, die Paaren von Knoten entsprechen.

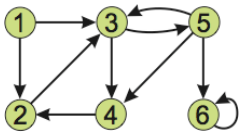
- Bei Multimenge kann ein Paar (v,w) mehrfach in E vorkommen → Mehrfachkanten
- Eine Kante (v,v) heißt Schleife (self-loop)

- Annahmen:
- V und E sind endliche Mengen
- Mehrfachkanten erlaubt

Gerichtete Graphen

- Sind die Paare in E geordnet: E ⊆ V x V → gerichteter Graph (Digraph)
- Kanten heißen dann: gerichtete Kanten (Bögen, directed edges, arcs)

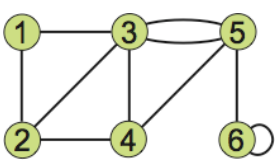
- Maximale Kantenanzahl eines Digraphen ohne Schleifen und Mehrfachkanten: $|E| \leq$



Ungerichtete Graphen

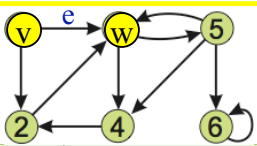
- Sind die Paare in E ungeordnet → (ungerichteter) Graph
- Kanten heißen dann: Kanten (edges)

- Maximale Kantenanzahl ohne Schleifen und Mehrfachkanten: $|E| \leq$



Definitionen (Nachbarn)

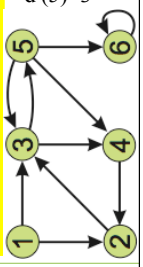
- Sei e=(v,w) eine Kante in E, dann sagen wir:
- v und w sind adjazent
- v (bzw. w) und e sind inzident
- v und w sind Endpunkte von e
- v und w sind Nachbarn
- e ist eine ausgehende Kante von v und eine eingehende Kante von w (falls G Digraph)



Definitionen für gerichtete Graphen G=(V,A)

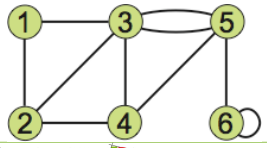
- Eingehende Nachbarmenge von $v \in V$: $N^+(v) := \{u \in V \mid (u,v) \in A\}$
- Ausgehende Nachbarmenge von $v \in V$: $N^-(v) := \{w \in V \mid (v,w) \in A\}$
- $A^+(v)$:=Menge der eingehenden Kanten von v
- $A^-(v)$:=Menge der ausgehenden Kanten von v
- $A(v) := A^+(v) \cup A^-(v)$
- Eingangsgrad $d^+(v) := |A^+(v)|$
- Ausgangsgrad $d^-(v) := |A^-(v)|$
- Knotengrad $d(v) := d^+(v) + d^-(v)$

$N^+(5)=\{3\}$
$N^-(5)=\{3,4,6\}$
$d^+(5)=1$
$d^-(5)=3$



Definitionen für ungerichtete Graphen G=(V,E)

- Nachbarmenge von $v \in V$: $N(v) := \{w \in V \mid (v,w) \in E\}$
- Menge der zu v inzidenten Kanten $E(v) := \{(u,v) \mid (u,v) \in E\}$
- Knotengrad $d(v)$ ist die Anzahl der zu v inzidenten Kanten, wobei eine Schleife 2 Mal gezählt wird



$N(5)=\{3,4,6\}$
$d(5)=4$

$d(6)=3$

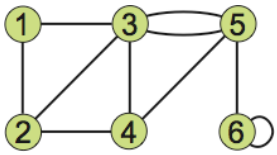
Lemma (gerade Knotengrade)

- In einem ungerichteten Graphen $G=(V,E)$ ist die Anzahl der Knoten mit ungeradem Knotengrad gerade.
- Summiert man über alle Knotengrade, so zählt man jede Kante genau zweimal:

$$\sum_{v \in V} d(v) = 2\,|E|$$

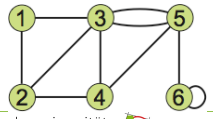
L.S.: gerade ← R.S.: gerade

also auch die Anzahl der ungeraden Summanden



Definitionen (Wege)

- Sei $G=(V,E)$ gerichtet oder ungerichtet:
- Ein Kantenzug (walk) der Länge k ist eine nicht-leere Folge $v_0,e_1,v_1,e_2,\ldots,e_k,v_k$ von abwechselnd Knoten und Kanten aus G mit $e_i=(v_{i-1},v_i)$ für $i=1,\ldots,k$.
- Man schreibt auch: $v_0,v_1,\ldots,v_k$
- Ein Weg (path) ist ein Kantenzug in dem alle Knoten verschieden sind.

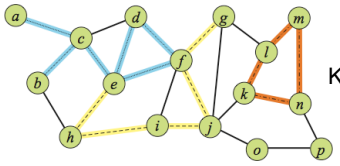


Kantenzug: 1,3,2,4,3,5

Weg: 1,3,2,4,5

Definitionen (Kreis)

- Sei $G=(V,E)$ gerichtet oder ungerichtet:
- Ist $v_0,e_1,v_1,e_2,\ldots,e_{k-1},v_{k-1}$ ein Weg mit $k \geqq 3$ und $e_k=(v_{k-1},v_0)$ eine Kante aus G, dann ist $v_0,e_1,v_1,e_2,\ldots,e_{k-1},v_{k-1},e_k,v_0$ ein Kreis der Länge k in G.



Kantenzug: a,c,e,f,d,e,c,b

Weg: g,f,j,i,h,e

Kreis: k,l,m,n,k

Darstellung von Graphen im Rechner: Statische Graphen

- Im Folgenden sei $V=\{v_1,v_2,\ldots,v_n\}$
- 1. Möglichkeit: Adjazenzlisten
- **Idee:** Speichere für jeden Knoten seine Nachbarmenge in einer Liste
- Realisierung: z.B. Knoten in Array und Nachbarkanten jedes Knotens als einfach verkettete Liste

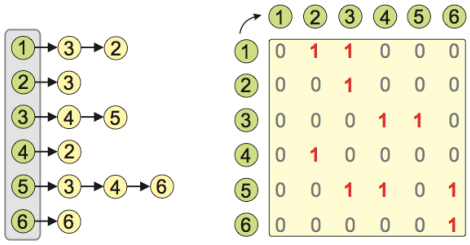
Darstellung von Graphen im Rechner: Statische Graphen

- 2. Möglichkeit: Adjazenzmatrix

- **Idee:** Eine VxV Matrix enthält 0/1-Einträge für jedes Knotenpaar $\{u,v\}$

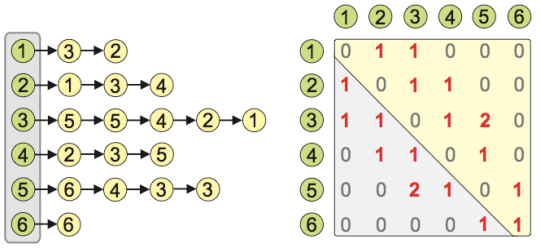
- Realisierung:
- Sei $M=(m_{i,j})$ eine n x n Matrix mit $m_{ij}:=1$ falls $(v_i,v_j)\in E$, und $m_{ij}:=0$ sonst.
- bei Mehrfachkanten schreibe statt 1 die Anzahl der Kanten

Darstellung gerichteter Graphen



Adjazenzlisten Adjazenzmatrix

Darstellung ungerichteter Graphen



Adjazenzlisten Adjazenzmatrix

ist symmetrisch: nur Speicherung der oberen Hälfte

Diskussion

HIER: ab jetzt Adjazenzlisten

für **dünne** Graphen vorzuziehen!

Adjazenzliste:
- Speicherplatzverbrauch: linear: $\Theta(|V|+|E|)$
- Zeit für Aufbau: linear: Θ
- Abfrage, ob Kante (u,v) existiert: Θ
- Iteration über alle Nachbarn von $v\in V$: Θ

Adjazenzmatrix:
- Speicherverbrauch immer quadratisch:
- Zeit für Aufbau: immer quadratisch: Θ
- Abfrage, ob Kante (u,v) existiert: Θ
- Iteration über alle Nachbarn von $v\in V$: Θ

Definitionen

- Die Dichte (density) eines Graphen G ist das Verhältnis $|E| / |V|$.
- G heißt dünn, falls seine Dichte $O(1)$ ist
- G heißt dicht, falls seine Dichte $\Omega(|V|)$ ist.

Darstellung von Graphen im Rechner: Dynamische Graphen

- Dynamisch unter den Operationen:
  - Hinzufügen neuer Knoten und Kanten
  - Entfernen von Knoten und Kanten

- **Idee:** für gerichtete Graphen:
- Inzidenzlisten: speichere ein- und ausgehende Knoten bei v
- Knoten in doppelt verketteter Liste (damit Entfernen in konstanter Zeit)
- Inzidenzlisten in doppelt verketteten Listen

Realisierung Dynamische Graphen: Liste für die Knoten

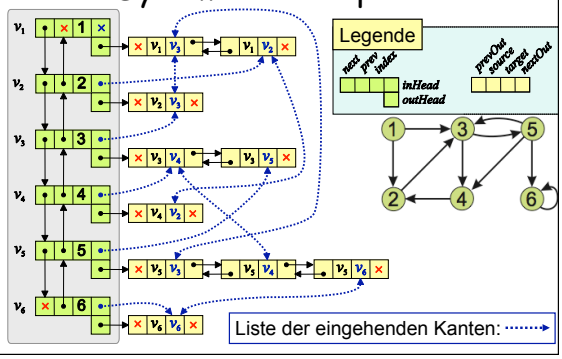
```
struct Node
  var Node prev        // Vorgänger Knotenliste
  var Node next        // Nachfolger in Knotenliste
  var Edge outHead     // Listenanfang ausgeh. Kanten
  var Edge inHead      // Listenanfang eingeh. Kanten
  var int index        // fortlaufender Index
end struct
```


Realisierung Dynamische Graphen: Liste für die Kanten

```
struct Edge
  var Edge prevOut  //Vorgänger in Liste ausg. Kanten
  var Edge nextOut  //Nachfolger in Liste ausg. Kanten
  var Edge prevIn   // Vorgänger in Liste eing. Kanten
  var Edge nextIn   // Nachfolger in Liste eing. Kanten
  var node source   // Anfangsknoten der Kante
  var node target   // Endknoten der Kante
end struct
```

Achtung: jeder Kanteneintrag ist genau einmal in Liste enthalten

Darstellung von Graphen im Rechner: Dynamische Graphen

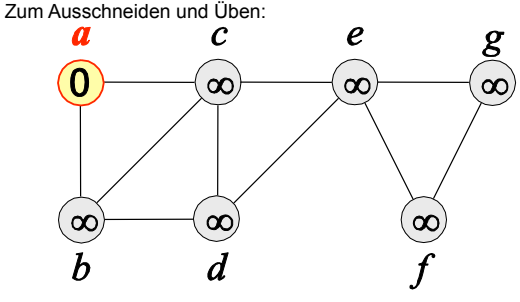


Analyse Dynamischer Graphen

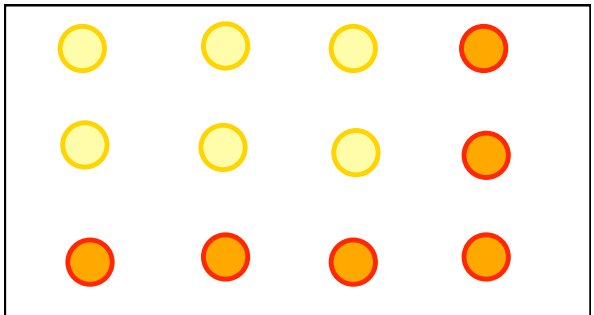
- Speicherplatzverbrauch: linear: Θ
- Zeit für Aufbau: linear: Θ
- Abfrage, ob Kante (u,v) existiert: Θ
- Iteration über alle Nachbarn von $v \in V$: Θ
- Iteration über alle ausg. Kanten von $v \in V$: Θ
- Iteration über alle eing. Kanten von $v \in V$: Θ
- Einfügen eines Knotens bzw. Kante: Θ
- Entfernen einer Kante: Θ
- Entfernen eines Knotens: Θ

Man geht davon aus, dass man jeweils Zeiger auf die Knoten und beim Entfernen auch auf die Kanten gegeben hat

Zum Ausschneiden und Üben:



Q a b c d e f g



Q: