Grundlagen der Algorithmen und Datenstrukturen

Kapitel 8

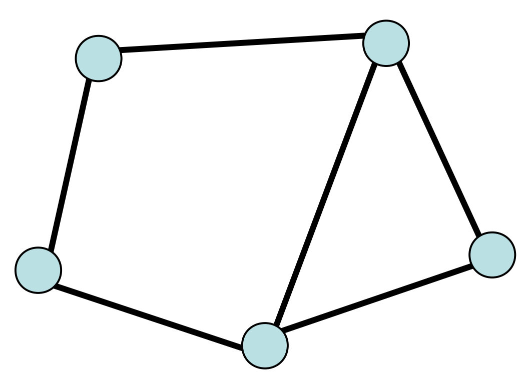
Christian Scheideler + Helmut Seidl

SS 2009

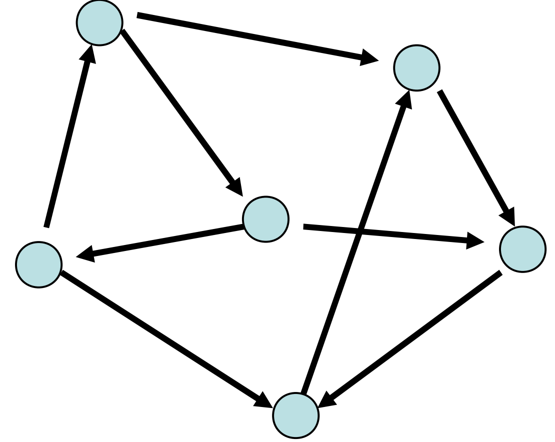
Graphen

Graph $G=(V,E)$ besteht aus

- Knotenmenge V
- Kantenmenge E ——



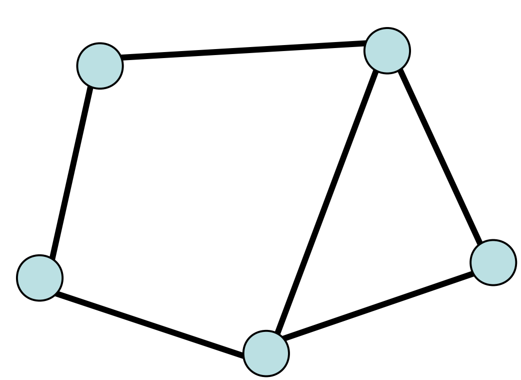
ungerichteter Graph



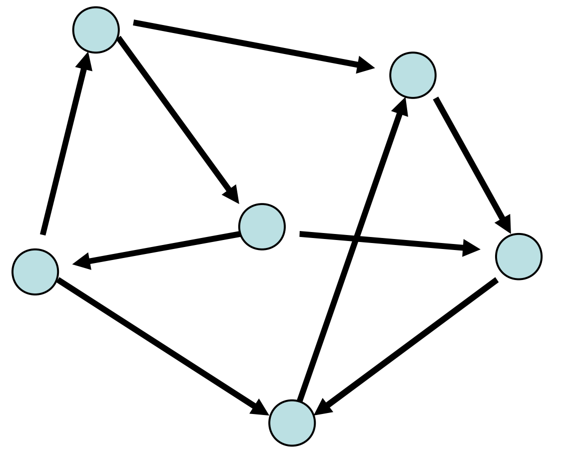
gerichteter Graph

Graphen

- Ungerichteter Graph: Kante repräsentiert durch Teilmenge $\{v,w\} \subset V$
- Gerichteter Graph: Kante repräsentiert duch Paar $(v,w) \in V \times V$ (bedeutet $v \longrightarrow w$)



ungerichteter Graph



gerichteter Graph

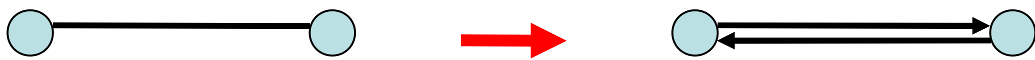
Graphen

Anwendungen:

- Ungerichtete Graphen: Symmetrische Beziehungen jeglicher Art (z.B. $\{v,w\} \in E$ genau dann, wenn Distanz zwischen v und w maximal 1 km)
- Gerichtete Graphen: asymmetrische Beziehungen (z.B. $(v,w) \in E$ genau dann, wenn Person v Person w mag)

Graphen

Im folgenden: nur gerichtete Graphen.
Modellierung eines ungerichteten Graphen als gerichteter Graph:



Ungerichtete Kante ersetzt durch zwei gerichtete Kanten.

- n: aktuelle Anzahl Knoten
- m: aktuelle Anzahl Kanten

Operationen auf Graphen

$G=(V,E)$: Graph-Variable

- Node: DS für Knoten, Edge: DS für Kanten

Operationen:

- G.insert(Edge e): $E=E \cup \{e\}$;
- G.remove(Key i, Key j): $E=E \setminus \{e\}$; für die Kante $e=(v,w)$ mit $Key(v)=i$ und $Key(w)=j$
- G.insert(Node v): $V=V \cup \{v\}$;
- G.remove(Key i): sei $v \in V$ der Knoten mit $Key(v)=i$. $V:=V \setminus \{v\}$, $E:=E \setminus \{(x,y) \mid x=v \vee y=v\}$
- G.find(Key i): gib Knoten v aus mit $Key(v)=i$
- G.find(Key i, Key j): gib Kante (v,w) aus mit $Key(v)=i$ und $Key(w)=j$

Operationen auf Graphen

Anzahl der Knoten oft fest. In diesem Fall:

- $V=\{0,\ldots,n-1\}$ (Knoten hintereinander nummeriert, identifiziert durch ihre Keys)

Relevante Operationen:

- G.insert(Edge e): $E=E \cup \{e\}$;
- G.remove(Key i, Key j): $E=E \setminus \{e\}$; für die Kante $e=(i,j)$
- G.find(Key i, Key j): gib Kante $e=(i,j)$ aus

Operationen auf Graphen

Anzahl der Knoten variabel:

- Hashing kann verwendet werden, um Keys von n Knoten in den Bereich $\{0, \ldots, O(n)\}$ zu hashen.
- Damit kann variabler Fall auf den Fall einer statischen Knotenmenge reduziert werden. (Nur $O(1)$-Vergrößerung gegenüber statischer Datenstruktur)

Operationen auf Graphen

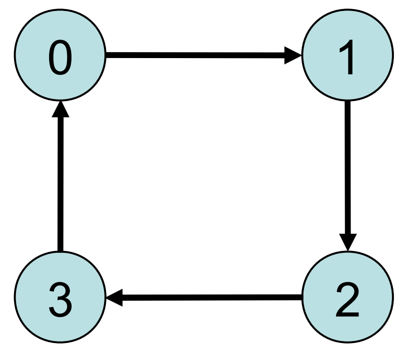
Im folgenden: Konzentration auf statische Anzahl an Knoten.

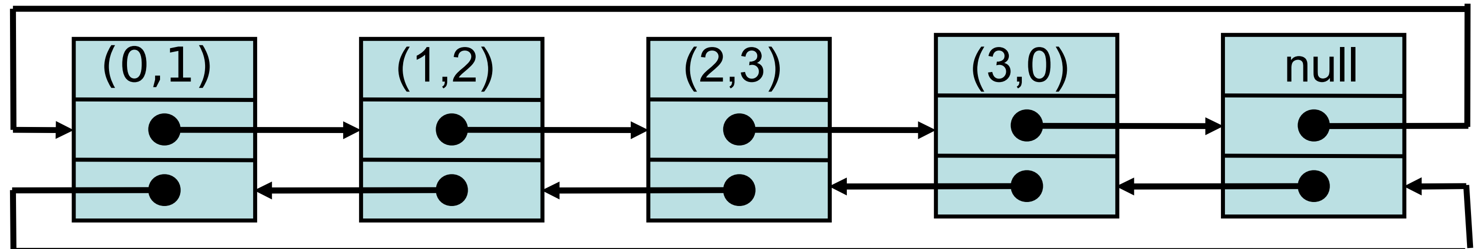
Parameter für Laufzeitanalyse:

- n: Anzahl Knoten
- m: Anzahl Kanten
- d: maximaler Knotengrad (maximale Anzahl ausgehender Kanten von Knoten)

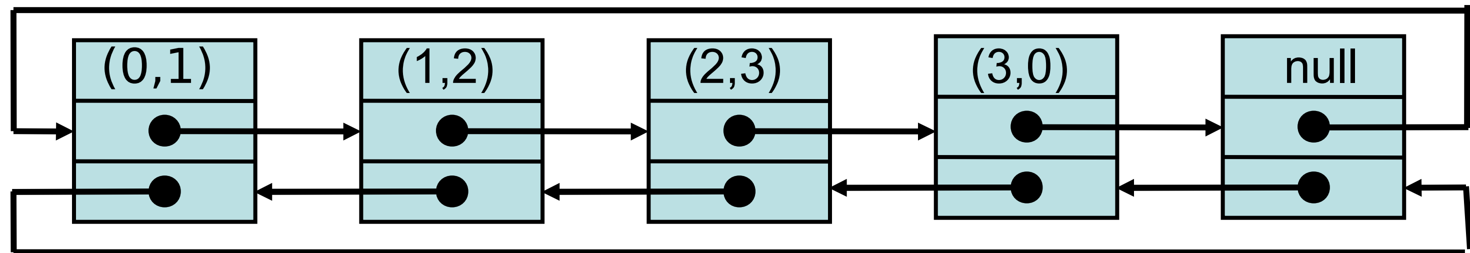
Graphrepräsentationen

Kapitel 8.1: Sequenz von Kanten





Sequenz von Kanten

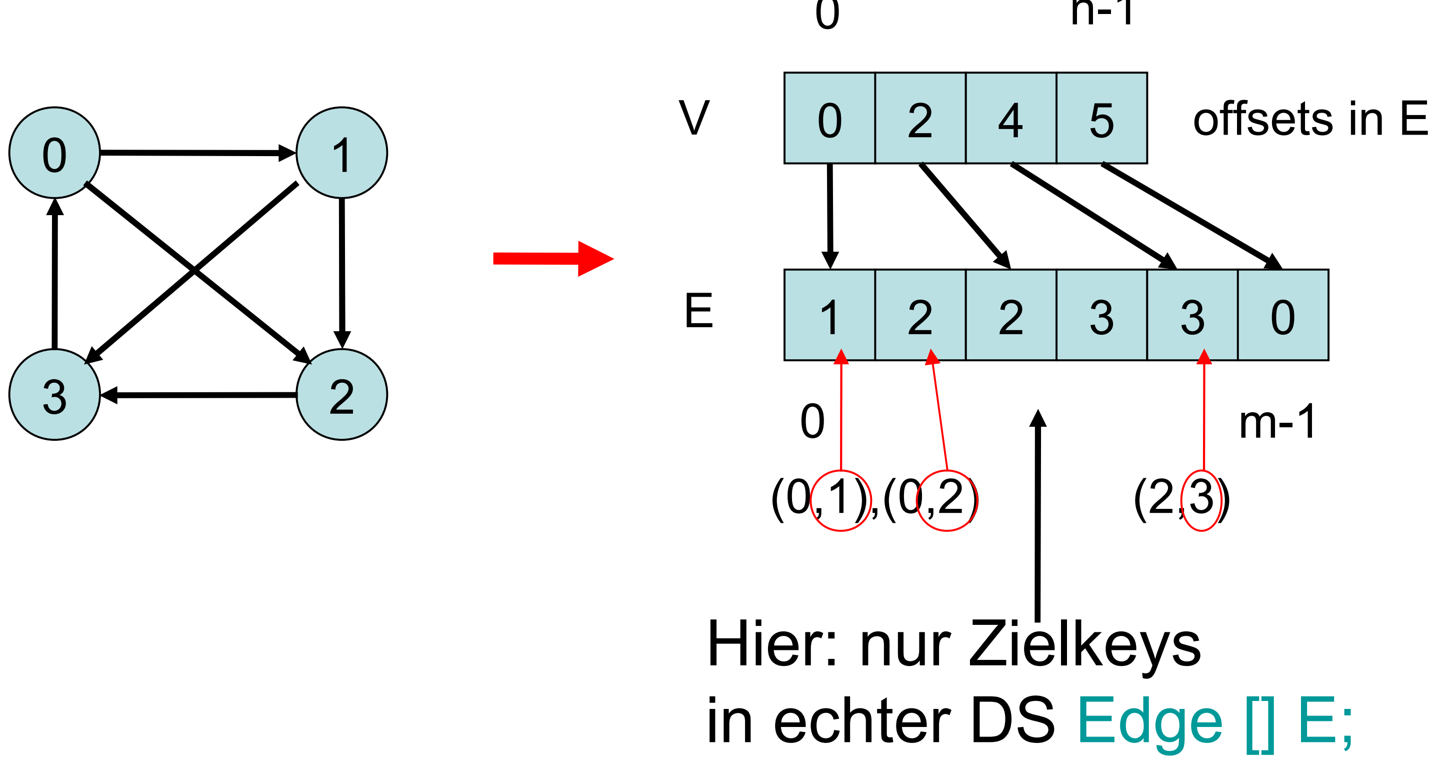


Zeitaufwand:

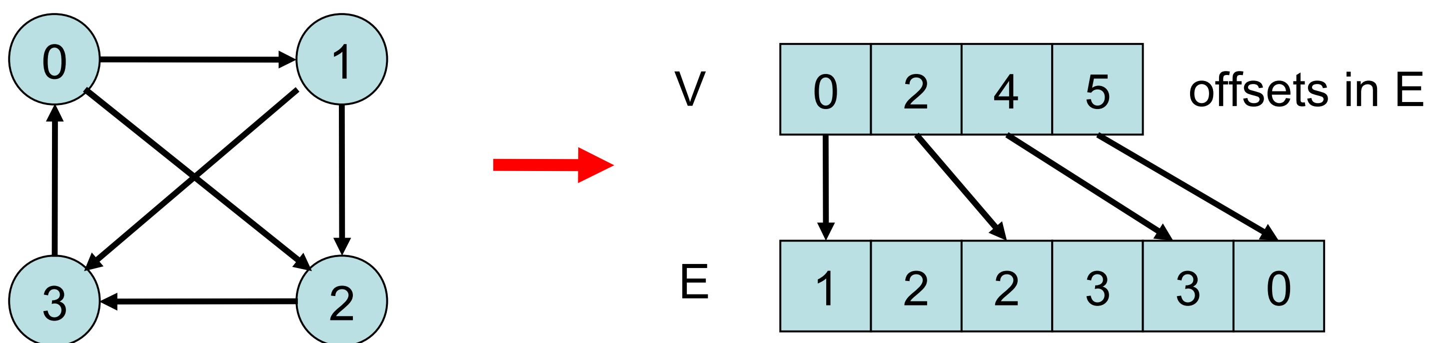
- G.find(Key i, Key j): Θ (m) im worst case
- G.insert(Edge e): O(1)
- G.remove(Key i, Key j): Θ (m) im worst case

Graphrepräsentationen

Kapitel 8.2: Adjazenzfeld



Adjazenzfeld

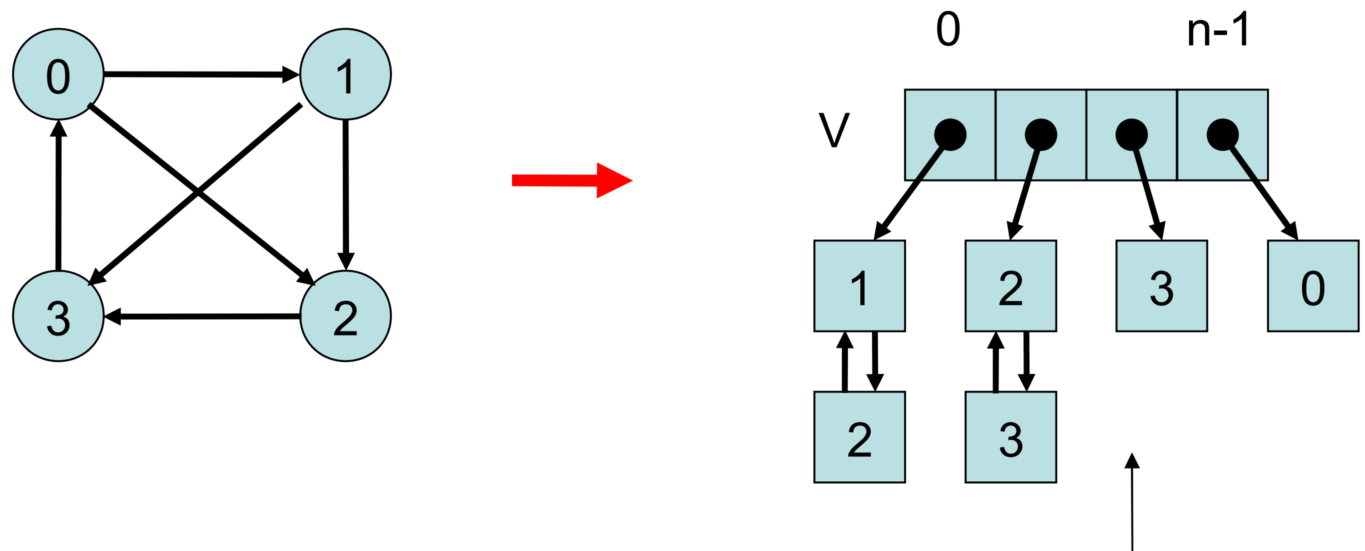


Zeitaufwand:

- G.find(Key i, Key j): Zeit O(d)
- G.insert(Edge e): Zeit O(m) (worst case)
- G.remove(Key i, Key j): Zeit O(m) (worst case)

Graphrepräsentationen

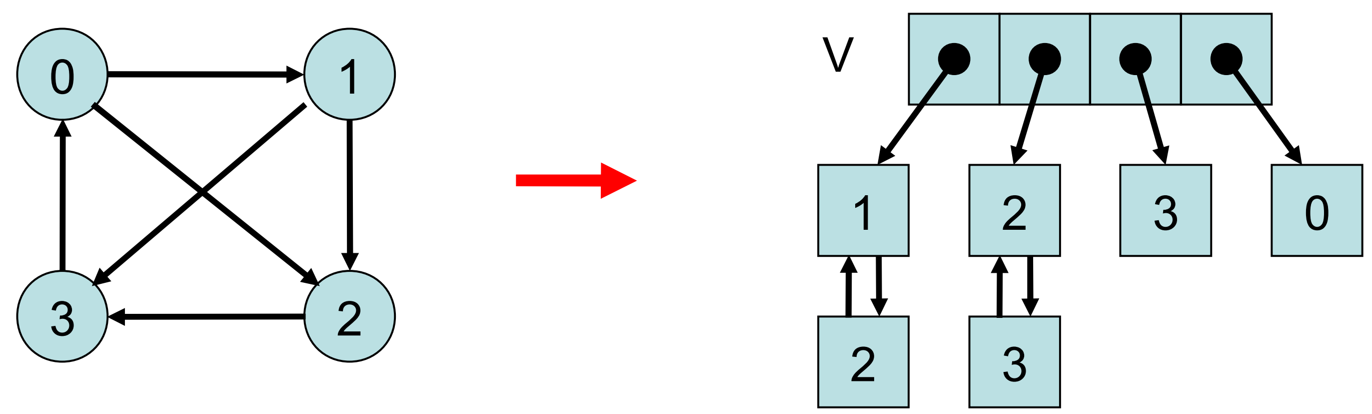
Kapitel 8.3: Adjazenzliste



Hier: nur Zielkeys

In echter DS: DList<Edge> [] V;

Adjazenzliste



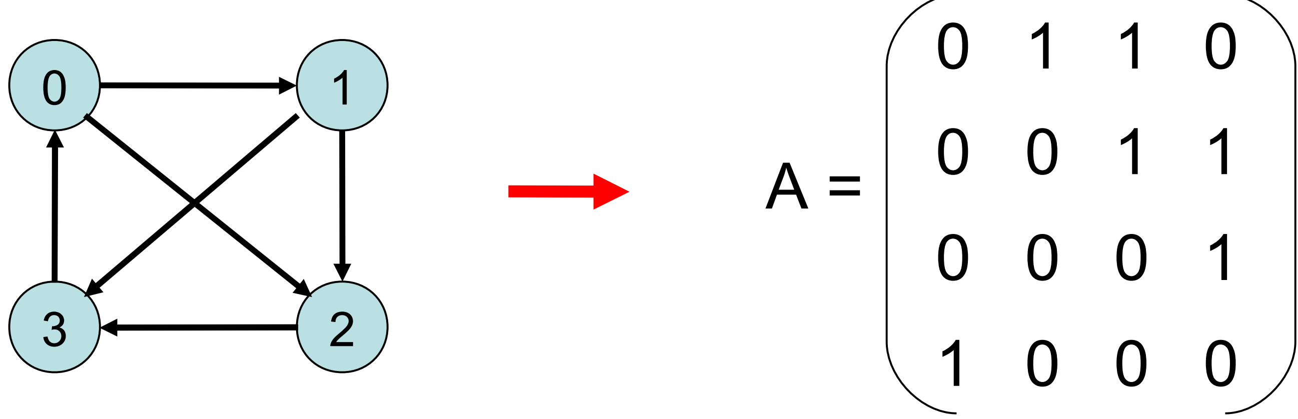
Zeitaufwand:

- G.find(Key i, Key j): Zeit O(d)
- G.insert(Edge e): Zeit O(d)
- G.remove(Key i, Key j): Zeit O(d)

Problem: d kann groß sein!

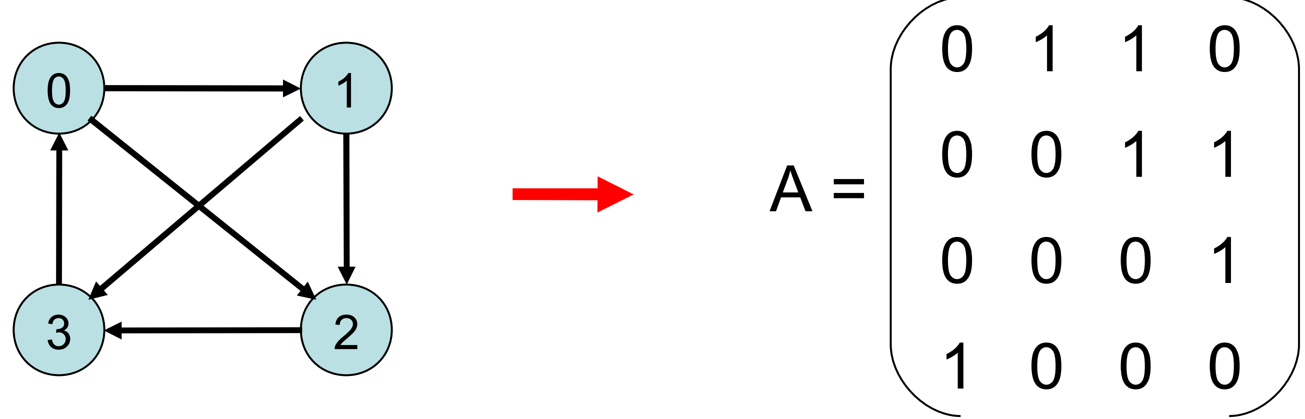
Graphrepräsentationen

Kapitel 8.4: Adjazenzmatrix



- A[i][j] $\in$ {0,1} (bzw. Zeiger auf Edge)
- A[i][j]=1 genau dann, wenn (i,j) $\in$ E

Adjazenzmatrix



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

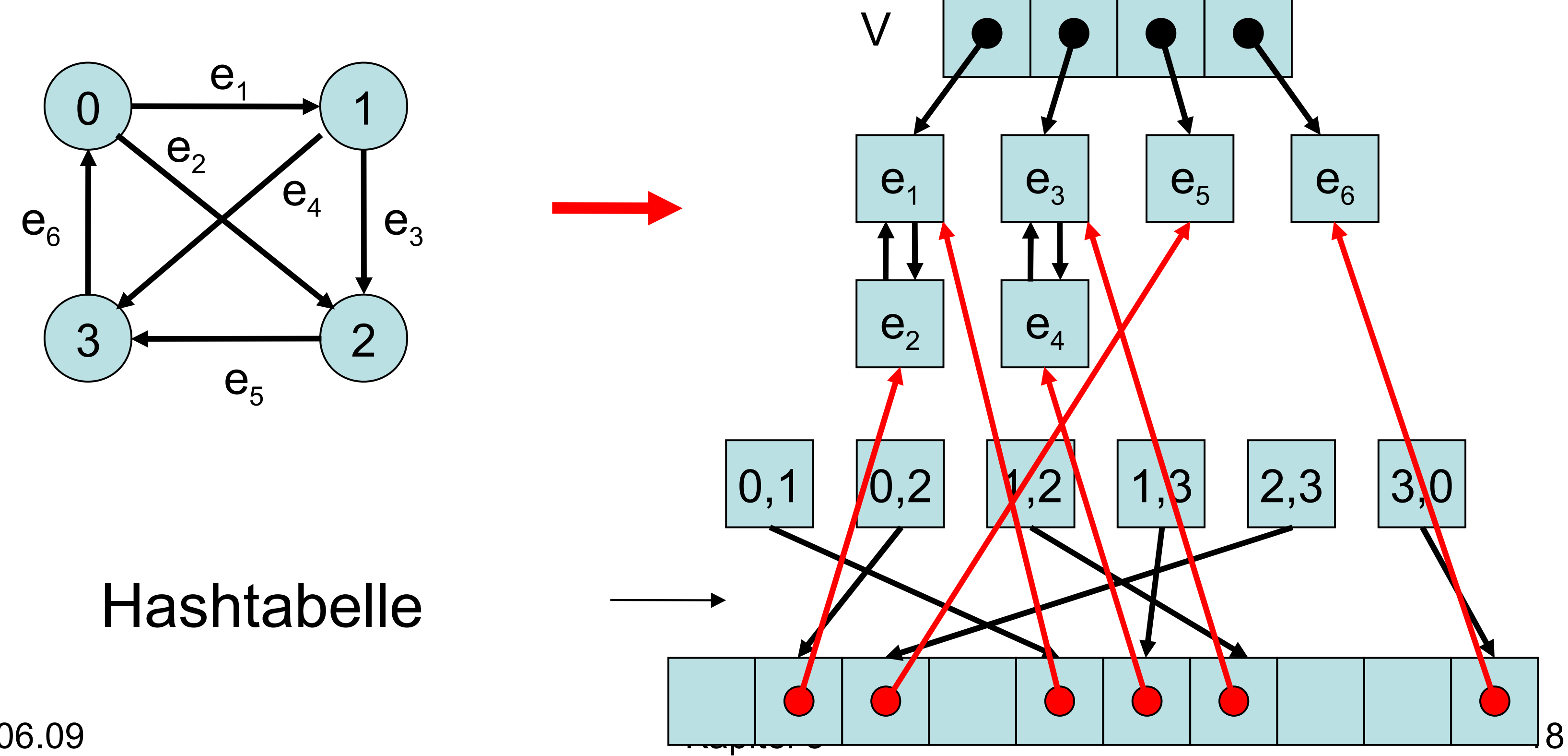
Zeitaufwand:

- G.find(Key i, Key j): Zeit $O(1)$
- G.insert(Edge e): Zeit $O(1)$
- G.remove(Key i, Key j): Zeit $O(1)$

Aber: Speicheraufwand $O(n^2)$

Graphrepräsentationen

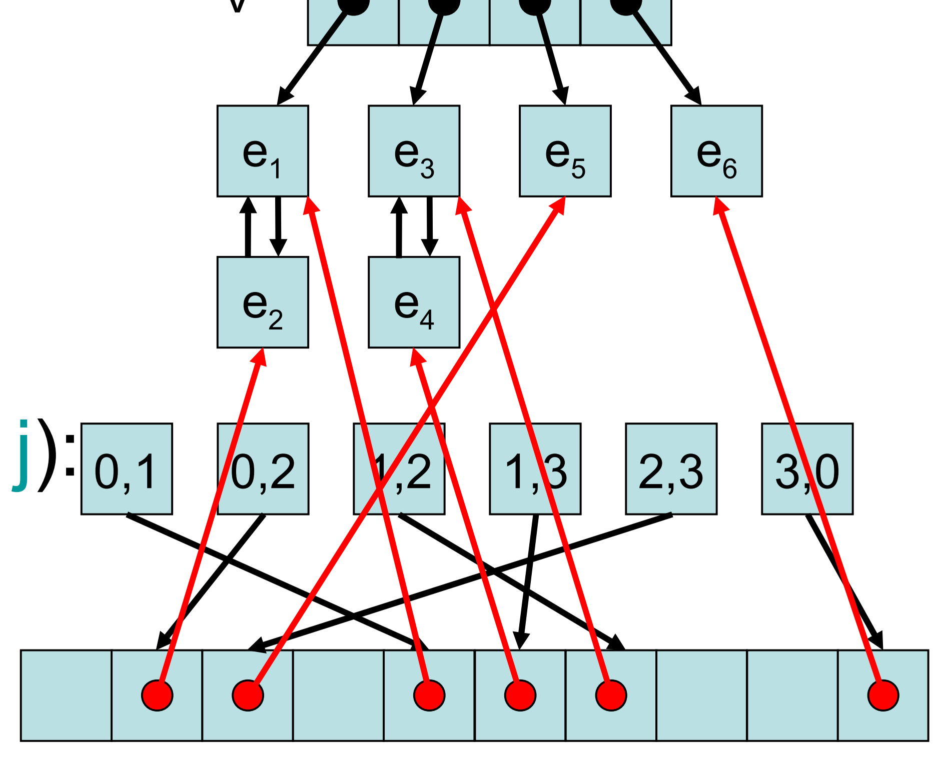
Besser: Adjazenzliste + Hashtabelle



Adjazenzliste+Hashtabelle

Zeitaufwand:

- G.find(Key i, Key j):
  O(1) (worst case)
- G.insert(Edge e):
  O(1) (im Mittel)
- G.remove(Key i, Key j):
  O(1) (im Mittel)
- Speicher: O(n+m)



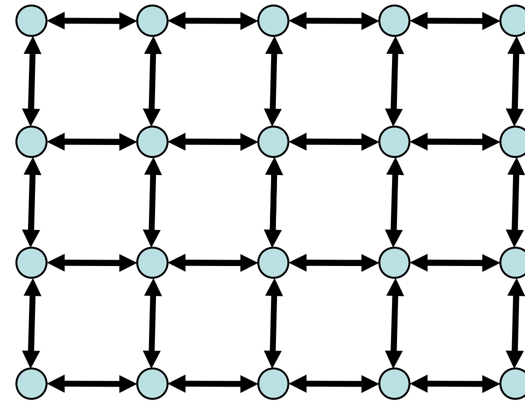
Graphrepräsentationen

Kapitel 8.5: Implizite Repräsentationen

(k,l)-Gitter $G=(V,E)$:

- $V=[k]\times[l]$ ($[a]=\{0,\dots,a-1\}$ für $a\in\mathbb{N}$)
- $E=\{((v,w),(x,y)) \mid (v=x \wedge |w-y|=1) \vee (w=y \wedge |v-x|=1)\}$

Beispiel: (5,4)-Gitter



Graphrepräsentationen

Kapitel 8.5: Implizite Repräsentationen

(k,l)-Gitter $G=(V,E)$:

- $V=[k]\times[l]$ ($[a]=\{0,\dots,a-1\}$ für $a\in IN$)
- $E=\{((v,w),(x,y)) \mid (v=x \wedge |w-y|=1) \vee (w=y) \wedge |v-x|=1)\}$

- Speicheraufwand: $O(\log k + \log l)$
  (speichere Kantenregel sowie k und l)
- Find-Operation: $O(1)$ Zeit (reine Rechnung)

Zusammenfassung

Verschiedene Graphrepräsentationen:

- Kantenliste
- Adjazenzfeld
- Adjazenzliste
- Adjazenzmatrix
- Adjazenzliste + Hashtabelle
- Implizite Graphrepräsentation

Weiter mit Graphdurchlauf (Kapitel 9)