Karlsruhe Institute of Technology

Tutorium 9: Graphrepräsentation und erste Graphalgorithmen

Matthias Schimek | 23. Juni 2017

TUTORIUM ZUR VORLESUNG ALGORITHMEN I IM SS17

Gliederung

1. Übungsklausur
2. Graphrepräsentation
3. Graphtraversierung

Übungsklausur

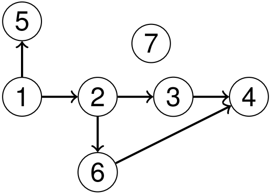
- Größe der Hashtabelle angeben $\rightarrow$ wichtig für Laufzeitgarantien
- Universelle Hashfamilien bieten i.A. kein perfektes Hashing

Graphen

Graphen in der Informatik:

- Inzidenzstrukur (=Nachbarschaftsstruktur) bestehend aus *Kanten* und *Knoten*
- $G = (V, E)$
- gängige Konvention: $|V| = n$, $|E| = m$
- man spricht von Linearzeit auf Graphen, wenn Algorithmus in $\mathcal{O}n + m$ Zeit läuft
- Unterscheide: *gerichtete* und *ungerichtete* Graphen

Graphrepräsentationen - Kantenfolge

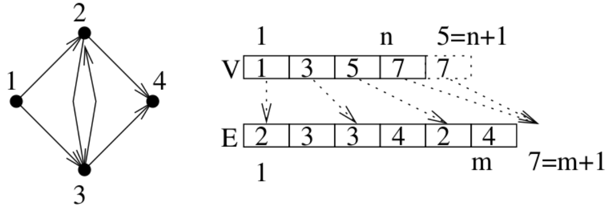


Repräsentation als Liste von Kanten

- $<(1,5),(1,2),(2,6),(2,3),(6,4),(3,4)>$
- Vorteile/Nachteile?

Graphrepräsentation - Adjazenzfeld

- $V = 1..n$ oder $0..n-1$
- Kantenfeld E speichert Ziele und zwar gruppiert nach Startknoten
- V speichert Index der ersten ausgehenden Kante
- Dummy-Eintrag $V[n+1]$ speichert $m+1$



Beispiel: Ausgangsgrad$(v) = \mathtt{V}[v+1] - \mathtt{V}[v]$

[1]

[1] Quelle: Vorlesungsfolien, Algo I, KIT

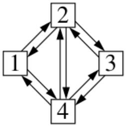
Aufgabe

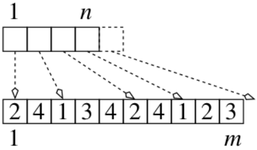
Gegeben sei ein Graph mit einer Repräsentation als Kantenliste.
Entwerfe einen Algorithmus, der diese in ein Adjazenzfeld konvertiert.

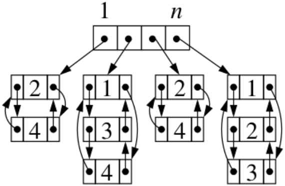
Graphrepräsentation - Adjazenzliste

speichere (doppelt) verkettete Liste adjazenter Kanten für jeden Knoten.

- \+ einfaches Einfügen von Kanten
- \+ einfaches Löschen von Kanten (ordnungserhaltend)
- – mehr Platz (bis zu Faktor 3) als Adjazenzfelder
- – mehr Cache-Misses



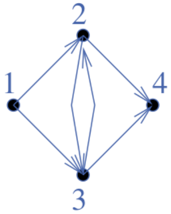




2

[1] Quelle: Vorlesungsfolien, Algo I, KIT

Graphrepräsentation - Adjazenzmatrix



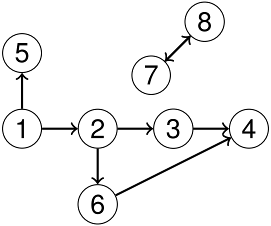
$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

[3]Gegeben Graph $G = (V, E)$

- $n * n$ - Matrix

[3]Quelle: Vorlesungsfolien, Algo I, KIT

Aufgabe



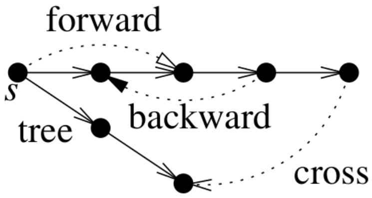
Gib den Graph als

- Kantenfolge
- Adjazenzfeld
- Adjazenzliste
- Adjazenzmatrix

an

Kantenklassifizierung

- Baumkanten: Elemente des Waldes, der bei der Suche gebaut wird
- Vorwärtskanten: verlaufen parallel zu Wegen aus Baumkanten
- Rückwärtskanten: verlaufen antiparallel zu Wegen aus Baumkanten
- Querkanten: alle übrigen



[4]

[4] Quelle: Vorlesungsfolien, Algo I, KIT

Breitensuche

Function bfs(s) :

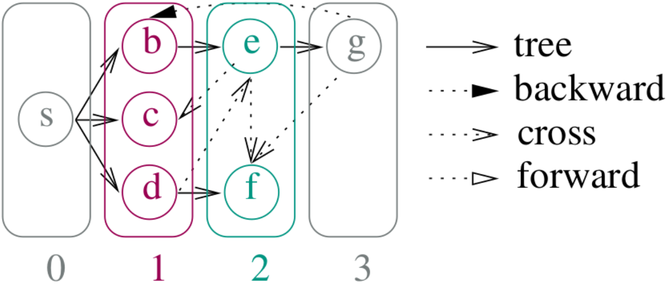
$Q := \langle s \rangle$ // aktuelle Schicht

while $Q \neq \langle\rangle$ **do**

explorierе Knoten in Q

merke dir Knoten der nächsten Schicht in Q'

$Q := Q'$



5

[5] Quelle: Vorlesungsfolien, Algo I, KIT

Aufgabe

Implementiere den Algorithmus der Breitensuche in Pseudocode.

Tiefensuche

[siehe Tafel]

Tiefensuchschema für $G = (V, E)$

```
unmark all nodes;
init
foreach s ∈ V do
    if s is not marked then
        mark s                              // make s a root and grow
        root(s)                             // a new DFS tree rooted at s
        DFS(s, s)


Procedure DFS(u, v : NodeId)                // Explore v coming from u
    foreach (v, w) ∈ E do
        if w is marked then traverseNonTreeEdge(v, w)
        else     traverseTreeEdge(v, w)
                 mark w
                 DFS(v, w)
    backtrack(u, v)          // return from v along the incoming edge
```

[6]

[6] Quelle: Vorlesungsfolien, Algo I, KIT

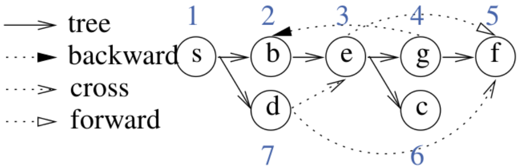
DFS-Nummerierung

init:	dfsPos=1 : $1..n$
root(s):	dfsNum[s]:= dfsPos++
traverseTreeEdge(v, w):	dfsNum[w]:= dfsPos++

$$u \prec v :\Leftrightarrow \mathrm{dfsNum}[u] < \mathrm{dfsNum}[v] \quad .$$

Beobachtung:

Knoten auf dem Rekursionsstapel sind bzgl. $\prec$ sortiert

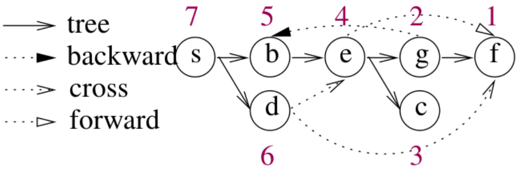


7

[7]Quelle: Vorlesungsfolien, Algo I, KIT

Fertigstellungszeit

init: finishingTime=1 : 1..n
backtrack(u, v): finishTime[v]:= finishingTime++

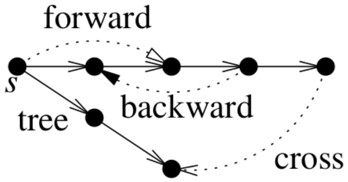


8

[8]Quelle: Vorlesungsfolien, Algo I, KIT

Kantenklassifizierung bei DFS

type (v, w)	$\text{dfsNum}[v] < \text{dfsNum}[w]$	$\text{finishTime}[w] < \text{finishTime}[v]$	w is marked
tree	yes	yes	no
forward	yes	yes	yes
backward	no	no	yes
cross	no	yes	yes



9

[7] Quelle: Vorlesungsfolien, Algo I, KIT

Aufgabe

Es sei $G = (V, E)$ ein gerichteter azyklischer Graph (DAG) mit endlich vielen und mindestens einem Knoten.

Zeige, dass G mindestens einen Knoten mit Eingangsgrad 0 besitzt.