

Algorithmen I

Tutorium 1 - 8. Sitzung

Dennis Felsing

`dennis.felsing@student.kit.edu`
`www.stud.uni-karlsruhe.de/~ubcqr/algo`

2011-06-06



Überblick

- 1 **Graphrepräsentation**
 - Allgemeines
 - Adjazenzliste
 - Adjazenzmatrix
 - Adjazenzfeld
 - Aufgaben

- 2 **Graphenalgorithmen**
 - Tiefensuche
 - Topologisches Sortieren

Graphen

Begriffe

Graph $G = (V, E)$, **Knoten** V , **Kanten** $E \subseteq V \times V$

Gerichteter Graph, Ungerichteter Graph

Grad $d(v)$: Anzahl Kanten an Knoten v

Ausgangsgrad $d^+(v)$: Anzahl ausgehender Kanten an $v \in V$

Eingangsgrad $d^-(v)$: Anzahl eingehender Kanten an $v \in V$

Pfad (Weg): Folge von Knoten $v_1 \dots v_n$ mit $(v_k, v_{k+1}) \in E$

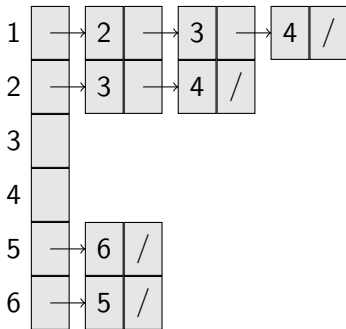
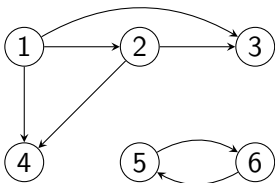
Zusammenhängender Graph: $\forall v, w \in V : \exists$ Weg von v nach w

Zyklus: Weg mit gleichem Anfangs- und Endknoten

Eulerkreis: Zyklus, der alle Kanten genau ein mal benutzt

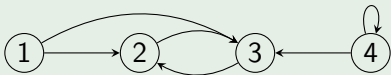
...

Adjazenzliste



Aufgabe

Gib die Adjazenzliste zu folgendem Graphen an:



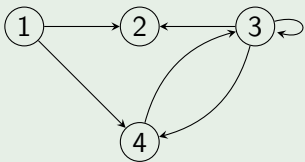
Adjazenzmatrix

Definition

Die **Adjazenzmatrix** eines Graphen $G = (V, E)$ ist eine Matrix mit

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

Beispiel



	1	2	3	4
1	0	1	0	1
2	0	0	0	0
3	0	1	1	1
4	0	0	1	0

Adjazenzmatrix

Definition

Die **Adjazenzmatrix** eines Graphen $G = (V, E)$ ist eine Matrix mit

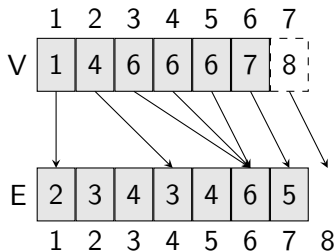
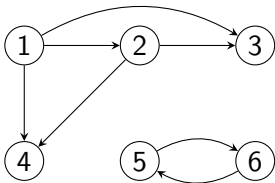
$$A_{ij} = \begin{cases} 1 & \text{falls } (i, j) \in E \\ 0 & \text{falls } (i, j) \notin E \end{cases}$$

Aufgabe

Gib die Adjazenzmatrix zu folgendem Graphen an:



Adjazenzfeld



Aufgabe

Gib das Adjazenzfeld zu folgendem Graphen an:



Quiz

Sei $G = (V, E)$ ein gerichteter Graph mit $n := |V|$ und $m := |E|$ und A die zugehörige Adjazenzmatrix.

- Der Speicherverbrauch von Adjazenzliste liegt in $\Theta(n + m)$
- Der Speicherverbrauch von Adjazenzmatrix liegt in $\Theta(n^2)$
- Der Speicherverbrauch von Adjazenzfeld liegt in $\Theta(n + m)$
- Einfügen von Kanten geht in Adjazenzlisten in $\Theta(1)$ ✓
- Suchen nach Kanten geht in Adjazenzlisten in $\Theta(1)$ ✗
- Was bedeutet der Wert $n = (A^k)_{ij}$? Es gibt n Wege der Länge k von i nach j .
- A ist symmetrisch $\Leftrightarrow G$ ist ungerichtet
- Einfügen von Kanten geht in Adjazenzfeldern in $\Theta(1)$ ✗

Quiz

DAG (Directed Acyclic Graph) bezeichnet einen gerichteten Graphen ohne Zykel

- DAGs lassen sich in einer Adjazenzmatrix als obere Dreiecksmatrix repräsentieren ✓

- Der Graph zur Adjazenzmatrix
$$\begin{array}{c} 1 \quad 2 \quad 3 \quad 4 \\ 1 \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$
 ist

zusammenhängend ✗

- Für jeden Knoten n eines gerichteten Graphen ist Ausgangsgrad $d^+(n) = d^-(n)$ ✗
- Für keinen Knoten n eines gerichteten Graphen ist Ausgangsgrad $d^+(n) = d^-(n)$ ✗

Aufgabe

Entwerfe einen Algorithmus, der eine Liste von Kanten in eine entsprechende Darstellung als Adjazenzfeld umwandelt.

Kreativaufgabe - Dynamisiertes Adjazenzfeld

Entwickle eine Graphenrepräsentation mit folgenden Eigenschaften:

- Stabile und eindeutige Knoten-IDs
- Eindeutige Kanten-IDs
- **Wahlfreier Zugriff** auf Knoten und Kanten in konstanter Zeit
- **Navigation** auf Knoten und Kanten in konstanter Zeit
- Amortisiert konstantes **Einfügen** von Knoten und Kanten
- Amortisiert konstantes **Entfernen** von Knoten und Kanten

Tiefensuche

DFS(G)

```
1  for each  $u \in G.V$ 
2       $u.color = \text{WHITE}$ 
3       $u.\pi = \text{NIL}$ 
4   $time = 0$ 
5  for each  $u \in G.V$ 
6      if  $u.color == \text{WHITE}$ 
7          DFS-VISIT( $G, u$ )
```

Tiefensuche

DFS-VISIT(G, u)

```
1  time = time + 1 // Weißer Knoten  $u$  gerade entdeckt
2   $u.d = time$ 
3   $u.color = \text{GRAY}$ 
4  for each  $v \in G.Adj[u]$ 
5      if  $v.color == \text{WHITE}$  // Erkunde Kante  $(u, v)$ 
6           $v.\pi = u$ 
7          DFS-VISIT( $G, v$ )
8   $u.color = \text{BLACK}$  //  $u$  ist fertig
9  time = time + 1
10  $u.f = time$ 
```

Tiefensuche

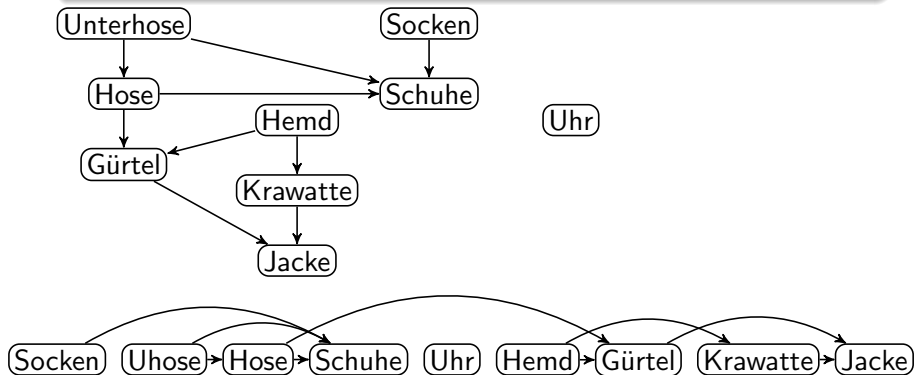
Kantenklassifikation

- **Baumkante** (u, v) : v wurde über (u, v) entdeckt
- **Rückwärtskante** (u, v) : v ist (Baumkanten-)Vorgänger von u
- **Vorwärtskante** (u, v) : u ist (Baumkanten-)Vorgänger von v , aber (u, v) keine Baumkante
- **Querkante** (u, v) : Rest

Topologisches Sortieren

Definition

Eine **Topologische Sortierung** eines DAG $G = (V, E)$ ist eine lineare Anordnung seiner Knoten, so dass für alle Kanten (u, v) u vor v eingeordnet ist.



Topologisches Sortieren

TOPOLOGICAL-SORT(u, v)

- 1 DFS(G) um $v.finished$ für alle $v \in V$ zu berechnen
- 2 Immer wenn *finished* gesetzt wird,
Knoten an Anfang einer Liste anfügen
- 3 **return** Entstandene Knotenliste

Quiz

- Alle Graphen lassen sich topologisch sortieren. ✗

Übersicht

1 Graphrepräsentation

- Allgemeines
- Adjazenzliste
- Adjazenzmatrix
- Adjazenzfeld
- Aufgaben

2 Graphenalgorithmen

- Tiefensuche
- Topologisches Sortieren

