

Kap. 6.4: Elementare Graphalgorithmen
 Kap. 6.5: Minimale Spannbäume

Carsten Gutwenger
 Lehrstuhl für Algorithm Engineering, LS11

19. VO 13. Juni 2006

Ankündigung

“Kleine” Sammelübung

immer
 Montags, 16-18Uhr
 GB 4, R113

2

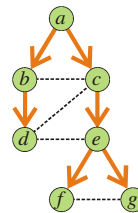
Überblick

- Wiederholung:
 - BFS & DFS
 - Komponenten eines Graphen
 - Kreise in Graphen

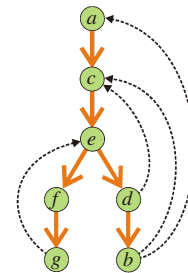
- Topologisches Sortieren
- Minimale Spannbäume

3

BFS & DFS



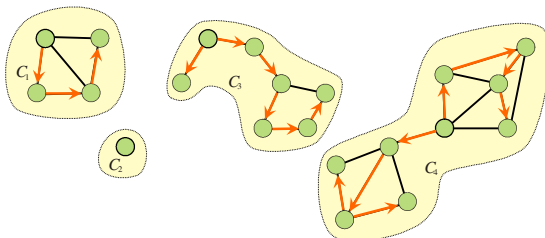
BFS-Baum



DFS-Baum

4

Komponenten eines Graphen



→ T-Kanten des DFS-Baums

5

Kreise in Graphen

```

(1) function ISACYCLIC(Graph G=(V,E)) : bool {
(2)   B := ∅
(3)   for all v∈V do { marked[v]:=0; π(v):=nil }
(4)   for all v∈V do {
(5)     if not marked[v] then
(6)       DFS-ACYCLIC(v)
(7)   }
(8)   if B ≠ ∅ then return false else return true
(9) }
  
```

6

Kreise in Graphen (2)

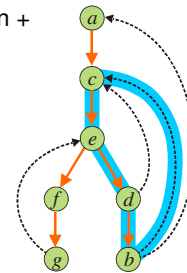
```

(1) procedure DFS-ACYCLIC(Node v) {
(2)   marked[v] := true
(3)   for all w ∈ N(v) do
(4)     if not marked[w] then
(5)       π[w] := v
(6)       DFS-ACYCLIC(w)
(7)     else if π[v] ≠ w then
(8)       B := B ∪ (v,w)
(9)   }
(10) }
  
```

7

Welche Kreise werden gefunden?

Pfad von T-Kanten +
eine B-Kante



8

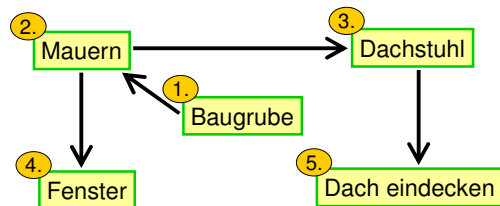
Kap. 6.4.3 Topologisches Sortieren

Achtung: in diesem Abschnitt
gerichtete Graphen!



9

Modellierung von Abhängigkeiten



→ gerichteter Graph
Kante (x,y) ⇒ Aufgabe x muss abgeschlossen
werden, bevor mit y angefangen werden kann

10

DAG

Grundvoraussetzung:

Graph ist azyklisch!
(also keine zyklischen Abhängigkeiten)

Definition:

Ein *DAG* (*directed acyclic graph*) ist ein gerichteter Graph, der keinen (gerichteten) Kreis enthält.

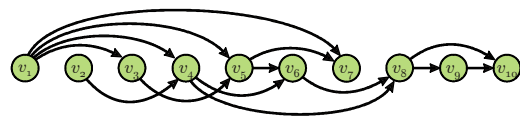
11

Topologisches Sortieren

Topologisches Sortieren

Gegeben: DAG $G = (V, A)$

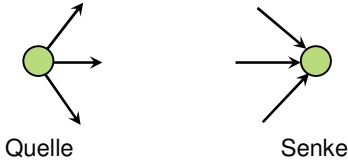
Gesucht: eine Sortierung v_1, \dots, v_n der Knoten von G mit $i < j$ für alle $(v_i, v_j) \in A$



12

Quellen & Senken

- Eine *Quelle* ist ein Knoten ohne eingehende Kanten
- Eine *Senke* ist ein Knoten ohne ausgehende Kanten



13

Beobachtung:

Jeder (nicht-leere) DAG $G=(V,A)$ hat mind. eine Quelle und eine Senke.

Beweis: Annahme: G hat *keine* Senke

- Verfolge von u_1 an immer ausgehende Kanten

$$p_i := u_1, u_2, \dots, u_i$$

- Falls $i > |V| \Rightarrow$ ein Knoten zweimal auf p_i

• \Rightarrow Kreis! **Widerspruch!**

- Analog für Quelle

14

Algorithmus



Idee:

Wähle immer Quelle und entferne sie dann

```
while G ist nicht leer do
  Wähle eine Quelle  $s$  in  $G$  und gib sie aus
   $G := G - s$ 
end while
```

15

Verbesserungen

Wie finden wir effizient eine Quelle?

- Wird v gelöscht, dann können nur Zielknoten w von Kanten (v,w) zu Quellen werden.
- Genügt ausgehende Nachbarmenge $N^+(v)$ zu betrachten.

Müssen wir Knoten wirklich löschen?

- Nein!
Verwalte Eingangsgrad in Knotenfeld *indeg*.

16

Algorithmus TopSort

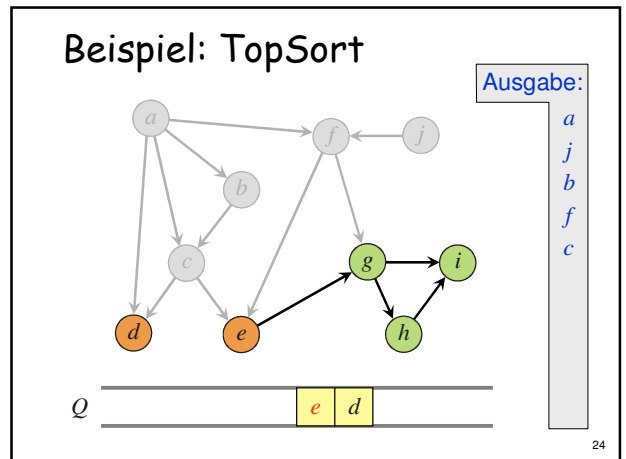
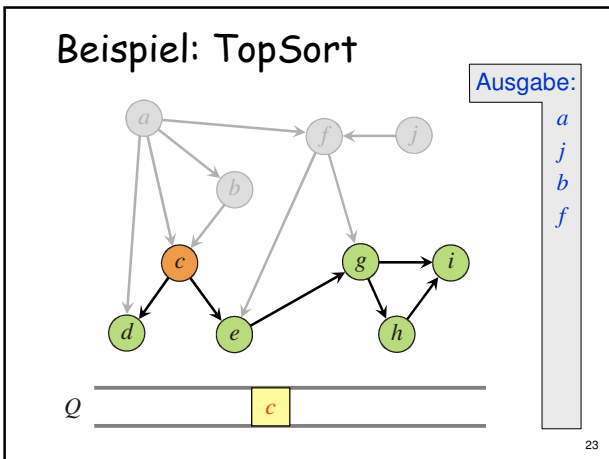
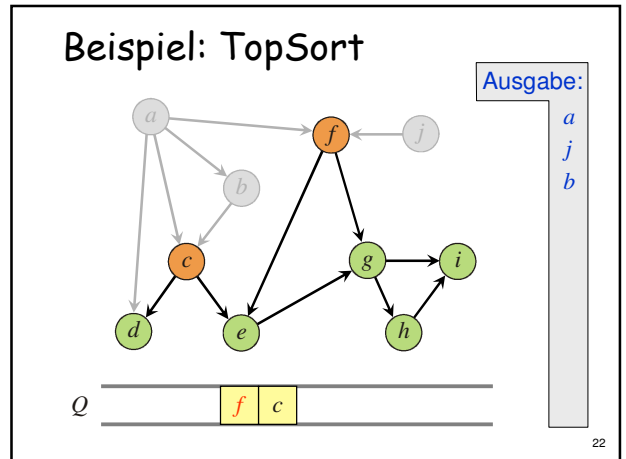
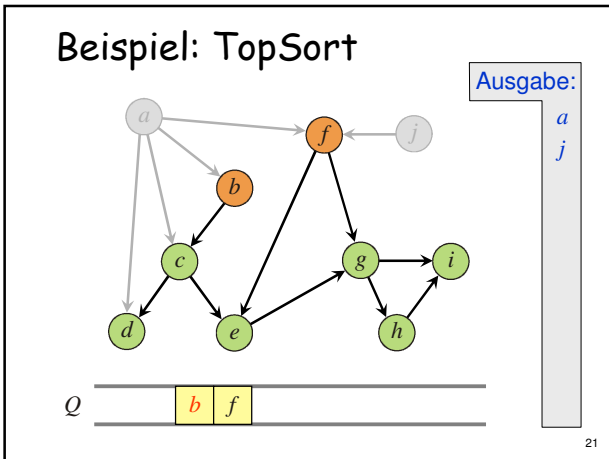
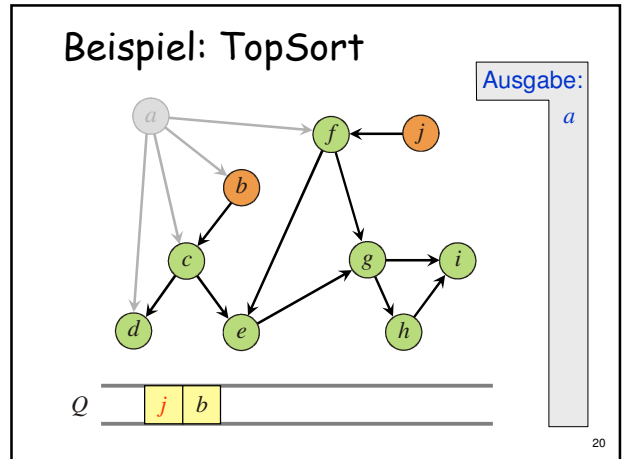
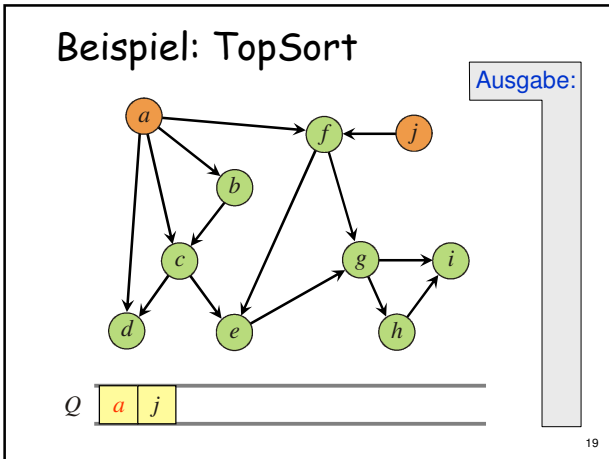
```
(1) var Queue Q
(2) var int indeg[V]
(3) ▷ Initialisierung
(4) for all  $v \in V$  do {
(5)    $indeg[v] := d^+(v)$ 
(6)   if  $indeg[v] = 0$  then Q.PUT( $v$ )
(7) }
```

17

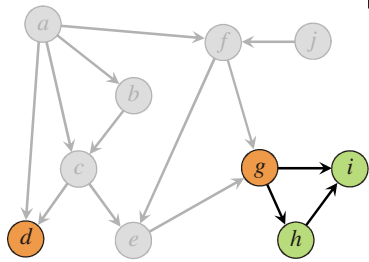
Algorithmus TopSort (2)

```
(8) ▷ Hauptschleife
(9) while not Q.ISEMPTY() do {
(10)   $v := Q.GET()$ 
(11)  Gib  $v$  aus
(12)  for all  $(v,u) \in A^+(v)$  do {
(13)     $indeg[u] := indeg[u] - 1$ 
(14)    if  $indeg[u] = 0$  then Q.PUT( $u$ )
(15)  }
(16) }
```

18



Beispiel: TopSort



Ausgabe:

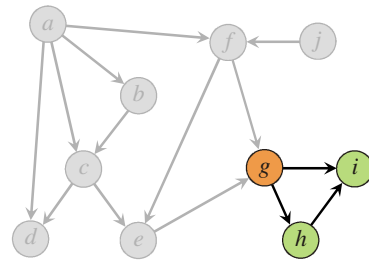
a
j
b
f
c
e

Q

d g

25

Beispiel: TopSort



Ausgabe:

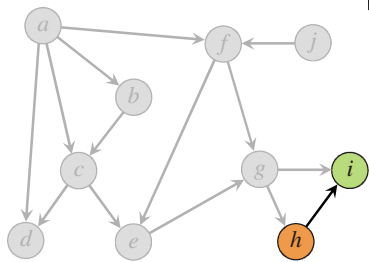
a
j
b
f
c
e
d

Q

g

26

Beispiel: TopSort



Ausgabe:

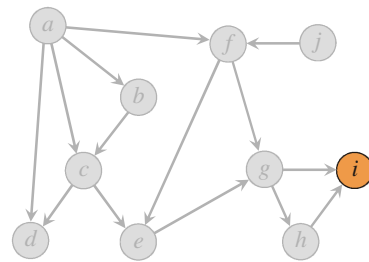
a
j
b
f
c
e
d
g

Q

h

27

Beispiel: TopSort



Ausgabe:

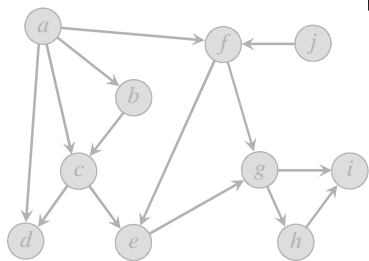
a
j
b
f
c
e
d
g
h

Q

i

28

Beispiel: TopSort



Ausgabe:

a
j
b
f
c
e
d
g
h
i

Q

29

Analyse der Laufzeit

- Initialisierung (Zeilen 1–7): $\Theta(|V|)$
(da $d^+(v)$ in konstanter Zeit abrufbar)
- Jeder Knoten kommt genau einmal in Q
 \Rightarrow **while**-Schleife wird $|V|$ -mal durchlaufen
- Die **for all**-Schleife (Zeile 12) wird insgesamt für jede Kante einmal durchlaufen: $\Theta(|A|)$
- Gesamtaufwand: $\Theta(|V|+|A|)$

30

Analyse TopSort

Theorem:

Der Algorithmus TopSort berechnet eine topologische Sortierung der Knoten eines DAGs $G=(V,A)$ in Zeit $\Theta(|V|+|A|)$.

31

Kap. 6.5 Minimale Spannbäume

Achtung: in diesem Abschnitt **ungerichtete gewichtete** Graphen!



ungerichteter Graph $G=(V,E)$

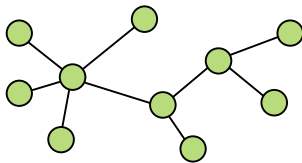
Gewichtsfunktion $w : E \rightarrow \mathbb{R}$

Kantengewichte: Kosten, Länge, ...

32

Bäume

Definition: Ein ungerichteter Graph heißt **Baum** (engl. *tree*), wenn er **zusammenhängend** und **kreisfrei** ist.



auch: *freier Baum* (\leftrightarrow *gewurzelter Baum*)
nicht zusammenhängend: *Wald*

33

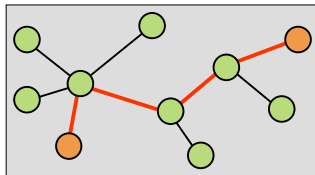
Theorem: Sei $G=(V,E)$ ungerichteter Graph (ohne Mehrfachkanten, Schleifen). Dann sind äquivalent:

- 1) G ist ein Baum
- 2) Jedes Paar von Knoten ist durch einen eindeutigen Weg verbunden.
- 3) G ist zshgd., zerfällt aber durch Entfernen einer beliebigen Kante in zwei Komponenten.
- 4) G ist zshgd. und $|E| = |V|-1$
- 5) G ist kreisfrei und $|E|=|V|-1$
- 6) G ist kreisfrei, aber durch Hinzufügen einer beliebigen Kante entsteht ein Kreis.

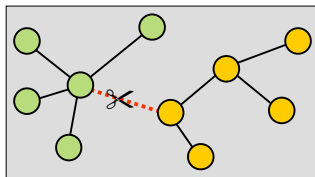
34

Erläuternde Bilder:

zu 2)



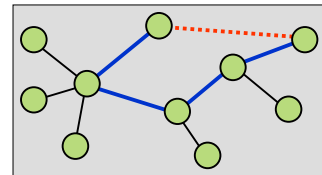
zu 3)



35

Erläuternde Bilder:

zu 6)

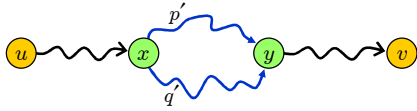


36

Beweis: durch Ringschluss 1) \Rightarrow 2) \Rightarrow ... 6) \Rightarrow 1)

1) \Rightarrow 2)

- Da Baum zshgd. \Rightarrow ex. Weg zwischen u und v
- Annahme: zwei Wege p und q



- $\Rightarrow p' + q'$ bilden Kreis! **Widerspruch**

37

2) \Rightarrow 3)

- zusammenhängend klar
- Annahme: G zerfällt nach Löschen von (u,v) nicht
- \Rightarrow Ex. Weg zwischen u und v , der (u,v) nicht benutzt
- \Rightarrow Weg nicht eindeutig! **Widerspruch**

38

3) \Rightarrow 4)

- zusammenhängend klar
- $\Rightarrow G$ besitzt DFS-Baum mit $|V|-1$ T-Kanten
- \Rightarrow Falls G mehr Kanten besitzt \Rightarrow ex. B-Kante
- Durch Löschen einer B-Kante zerfällt G aber nicht
- $\Rightarrow G$ hat $|V|-1$ Kanten

39

4) \Rightarrow 5)

- Da G zshgd. und $|E| = |V|-1$
 \Rightarrow die T-Kanten eines DFS-Baums sind genau die Kanten von G
- \Rightarrow keine B-Kanten
- $\Rightarrow G$ kreisfrei

40

5) \Rightarrow 6)

- G kreisfrei und besteht aus k Komponenten
- $\Rightarrow G$ hat $|V|-k$ Kanten
- $\Rightarrow k=1$ (d.h. G ist zusammenhängend)
- \Rightarrow Hinzufügen einer Kante erzeugt Kreis

41

6) \Rightarrow 1)

- Sei u,v beliebiges Knotenpaar
- \Rightarrow Entweder $(u,v) \in E$ oder
 - Hinzufügen von (u,v) erzeugt Kreis
 - \Rightarrow ex. Weg von u nach v in G
- $\Rightarrow G$ ist zusammenhängend

42

Spannbaum

Definition: Sei $G=(V,E)$ ein ungerichteter, zshgd. Graph. Ein **Untergraph** $T=(V,E_T)$ von G heißt **Spannbaum** (engl. *spanning tree*) von G , falls T ein **Baum** ist.

Gewicht eines Spannbaums:

$$w(T) := w(E_T) := \sum_{e \in E_T} w(e)$$

43

Minimaler Spannbaum

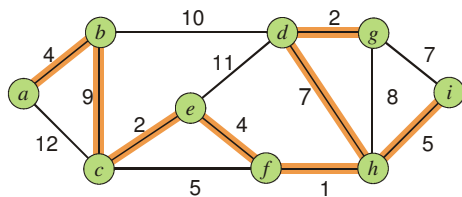
Minimum Spanning Tree (MST)

Gegeben:	ungerichteter, zshgd. Graph $G = (V,E)$ Gewichtsfunktion $w : E \rightarrow \mathbb{R}$
Gesucht:	ein Spannbaum T von G mit minimalem Gewicht $w(T)$

Anwendung: Konstruktion von Netzwerken

44

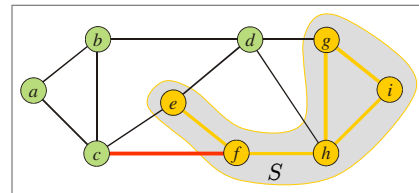
Beispiel für MST



45

Definition:

- Eine Teilmenge E' der Kanten heißt **aussichtsreich**, wenn es einen MST gibt, der alle Kanten aus E' enthält.
- Eine Kante (u,v) **verlässt** eine Knotenmenge $S \subset V$, wenn $u \in S$ und $v \notin S$



46

- $E' := \emptyset$ ist immer aussichtsreich.
- E' aussichtsreich und $|E'| = |V| - 1$
 $\Rightarrow G' := (V, E')$ ist ein MST

47

MST Eigenschaft

Lemma:

Sei $G=(V,E)$ zshgd. mit Gewichtsfunktion $w : E \rightarrow \mathbb{R}$.
Seien:

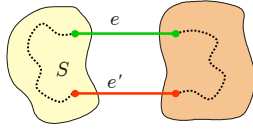
- $S \subset V$
- $E' \subset E$ aussichtsreich und **keine** Kante aus E' verlässt S
- $e \in E$ eine Kante mit minimalem Gewicht, die S verlässt

Dann ist $E' \cup \{e\}$ aussichtsreich.

48

Beweis:

- Ex. MST $T=(V,E_T)$ mit $E' \subseteq E_T$
- Falls $e \in E_T \Rightarrow$ fertig!
- Sonst: $E_T \cup \{e\}$ enthält Kreis:



- \Rightarrow Ex. Kante e' auf Kreis, die S verlässt
- Da $w(e) \leq w(e') \Rightarrow T - e' + e$ ist ein MST
- $\Rightarrow E' \cup \{e\}$ ist aussichtsreich

49