CoNe Freiburg

IIF INSTITUT FÜR INFORMATIK FREIBURG

Graphentheorie
05 – Wege, Kreise, Kreisfreie Graphen

Dr. Sven Köhler
Rechnernetze und Telematik
Technische Fakultät
Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

Wege

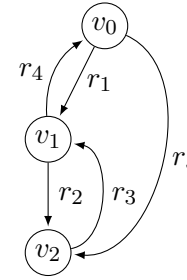
Definition 3.1

Ein *Weg/Pfad* in einem Graph G ist eine endlich Folge $P = (v_0, r_1, v_1, r_2, v_2, \ldots, r_k, v_k)$ mit $k \geq 0$ und

- $v_0, v_1, \ldots, v_k \in V(G)$
- $r_1, r_2, \ldots, r_k \in R(G)$
- $\forall i \in \{1, 2, \ldots, k\} : \alpha(r_i) = v_{i-1} \wedge \omega(r_i) = v_i$

bzw. eine endliche Folge $P = (v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k)$ mit $k \geq 0$ und

- $v_0, v_1, \ldots, v_k \in V(G)$
- $e_1, e_2, \ldots, e_k \in E(G)$
- $\forall i \in \{1, 2, \ldots, k\} : \gamma(e_i) = \{v_{i-1}, v_i\}$



$P = (v_0, r_1, v_1, r_2$
v_2, r_3, v_1, r_4
$v_0, r_5, v_2)$

Wege – ungerichtet

Sei $P = (v_0, r_1, v_1, \ldots, r_k, v_k)$ bzw. $P = (v_0, e_1, v_1, \ldots, e_k, v_k)$

Definition 3.1 (Forsetzung)

- *Startknoten* $\alpha(P) = v_0$
- *Endknoten* $\omega(P) = v_k$
- *Länge* $|P| = k$ ist die Anzahl der Kanten in P
- *Spur* $s(P) = (v_0, v_1, \ldots, v_k)$

P heißt *Kreis* wenn $\alpha(P) = \omega(P)$ und $|P| \geq 1$.

Wir sagen:

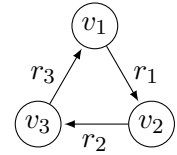
- P *verbindet* v_0 mit v_k
- P *berührt* v_i mit $0 \leq i \leq k$.

Einfache und Elementare Wege

Sei $P = (v_0, r_1, v_1, \ldots, r_k, v_k)$ bzw. $P = (v_0, e_1, v_1, \ldots, e_k, v_k)$

Definition 3.1 (Fortsetzung)

Weg P heißt *einfach* wenn $r_i \neq r_j$ bzw. $e_i \neq e_j$ für alle $i \neq j$.
Weg P heißt *elementar* wenn er einfach ist und keinen Knoten mehrmals berührt (Ausnahme: $\alpha(P) = \omega(P)$).



$P = (v_1, r_1, v_2, r_2, v_3, r_3, v_1)$

Beobachtung

Für jeden elementaren Weg P gilt $|P| \leq |V(G)|$.
Falls P kein Kreis ist, dann gilt sogar $|P| \leq |V(G)| - 1$.

Wege – Komposition

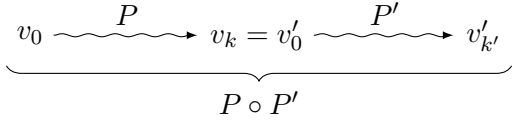
Definition

Seien P und P' zwei Wege mit

- $P = (v_0, r_1, v_1, \ldots, r_k, v_k)$
- $P' = (v'_0, r'_1, v'_1, \ldots, r'_{k'}, v'_{k'})$
- $\omega(P) = \alpha(P')$

Dann bezeichnet $P \circ P'$ die *Komposition* von P und P'.

$$P \circ P' := (v_0, r_1, \ldots, r_k, v_k = v'_0, r'_1, \ldots, r'_{k'}, v'_{k'})$$



Existenz von Kreisen

Lemma 3.3

Sei G ein gerichteter endlicher Graph mit $g^+(v) \geq 1$ für alle $v \in V(G)$.
Dann existiert ein elementarer Kreis.

Falls G einfach ist und $\exists g \geq 1 \forall v \in V(G) : g^+(v) \geq g$,
dann existiert ein elementarer Kreis der Länge $\geq g + 1$.

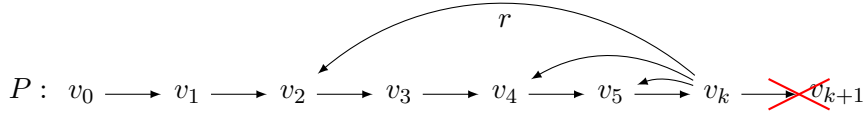
Erste Aussage ist Folgerung von zweiter:

- Entferne Schlingen und Parallelen um den Graphen einfach zu machen.
- Wende zweite Aussage an, erste Aussage folgt.

Wir beweisen nur den zweiten Teil.

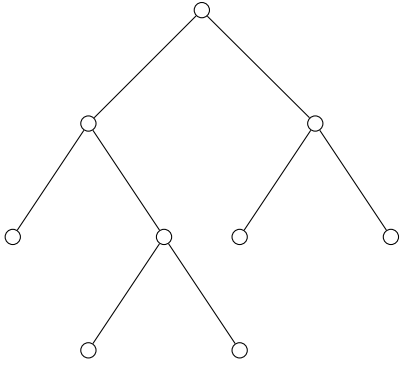
Existenz von Kreisen

Beweis.
Sei $P = (v_0, r_1, v_1, \ldots, r_k, v_k)$ ein **längster** elementarer Weg (aber kein Kreis!).
Es gehen $g^+(v_k) \geq g$ Pfeile von v_k aus.
Diese gehen zu Knoten, welche der Pfad schon berührt $(v_0, v_1, \ldots v_{k-1})$,
da sonst ein längerer elementarer Pfad existiert (Widerspruch zur Annahme).
Damit muss $k \geq g$ gelten, da G einfach ist.

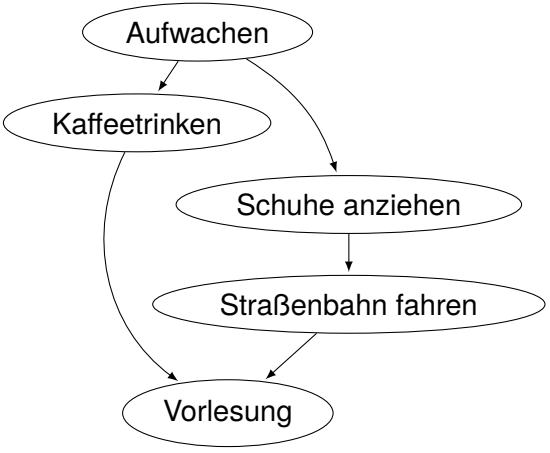


Wähle i minimal so dass $v_i = \omega(r)$ mit $r \in \delta^+(v_k)$.
Dann ist $P' = (v_i, r_{i+1}, \ldots, r_k, v_k) \circ (v_k, r, v_i)$ ein Kreis.
Es gilt $i \leq k - g$ und damit hat P' die Länge $k - i + 1 \geq g + 1$. □

Kreisfreie Graphen



Ein Baum



Ein Prozessgraph

Kreisfreie Graphen

Definition 3.6
Ein Graph heißt *kreisfrei*, wenn er keine einfachen Kreise enthält.

Gerichtete kreisfreie Graphen werden auch DAG (Directed Acyclic Graph) genannt.

Definition 3.7
Sei $G = (V, R, \alpha, \omega)$ ein gerichteter Graph.
Eine topologische Sortierung von G ist eine bijektive Abbildung

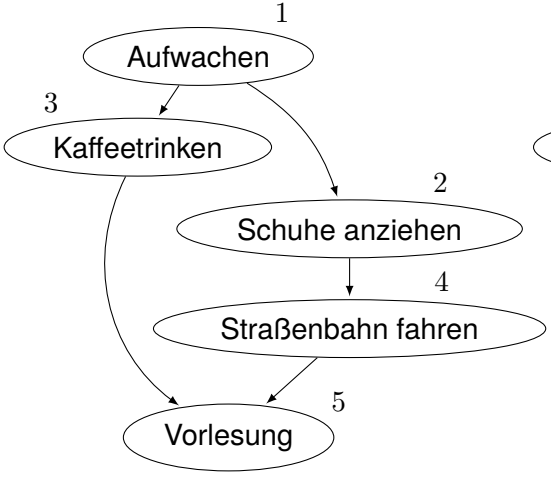
$$\sigma : V(G) \to \{1, 2, \ldots, n\}$$

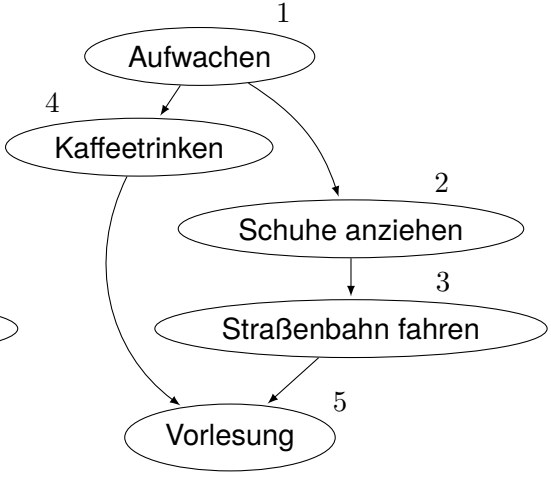
mit

$$\forall r \in R : \sigma(\alpha(r)) < \sigma(\omega(r)) \quad .$$

Viele Graphen haben erlauben mehrere topologische Sortierungen.

Topologische Sortierung – Beispiel





Topologische Sortierung

Satz 3.8

Ein gerichteter Graph ist genau dann kreisfrei, wenn er eine topologische Sortierung besitzt.

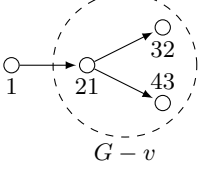
Beweis, Teil 1.

"$\Rightarrow$" Wenn G kreisfrei, dann existiert eine topologische Sortierung.
Wir verwenden vollständige Induktion über $n = |V|$.

Fall $n = 0$: offensichtlich

Fall $n > 0$:
Sei $v \in V(G)$ mit $g^-(v) = 0$. Einen solchen Knoten gibt es, da G kreisfrei ist.
Sei σ' ein topologische Sortierung von $G - v$.
Dann ist σ eine topologische Sortierung von G:

$$\sigma(u) := \begin{cases} 1 & \text{wenn } u = v \\ 1 + \sigma'(u) & \text{sonst} \end{cases}$$



□

Topologische Sortierung

Satz 3.8
Ein gerichteter Graph ist genau dann kreisfrei, wenn er eine topologische Sortierung besitzt.

Beweis, Teil 2.
"$\Leftarrow$" Wenn es eine topologische Sortierung gibt, dann ist der Graph kreisfrei.

Annahme:
Es gibt einen Kreis $C = (v_0, r_1, v_1, \ldots, r_k, v_k)$, wobei $v_k = v_0$. Da es eine topologische Sortierung σ gib, muss gelten:

$$\sigma(v_0) < \sigma(v_1) < \ldots < \sigma(v_k)$$

und damit

$$\lightning \quad \sigma(v_0) < \sigma(v_k) = \sigma(v_0) \quad .$$

Das ist ein Widerspruch. □

Topologische Sortierung – Vorbereitung

Algorithmus 2.2 Berechnung aller Innengrade

Eingabe: gerichteter Graph $G = (V, R)$

for each $v \in V$ **do**
  $inGrad[v] := 0$

for each $v \in V$ **do**
  for each $w \in N^+(v)$ **do** ▷ Iteriere durch $ADJ[v]$
    $inGrad[w] := inGrad[w] + 1$

return $inGrad$

Algorithmus 2.2 hat Laufzeit $\mathcal{O}(n + m)$.

Topologische Sortierung

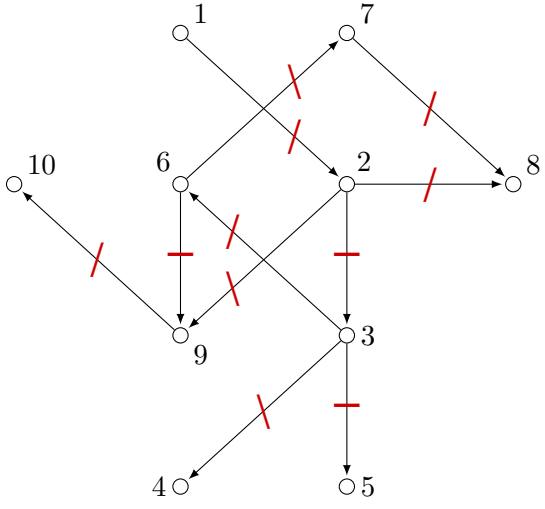
Algorithmus 3.1

Eingabe: gerichteter Graph $G = (V, R)$
Idee: entferne Knoten mit Innengrad 0 aus G, bis G leer ist.

Berechne Hilfsarray $inGrad$ mit Alg. 2.2
$L_0 := \{v \in V \mid inGrad[v] = 0\}$
for $i = 1, 2, \ldots, n$ **do**
    Wähle beliebiges $v \in L_0$
    $L_0 := L_0 \setminus \{v\}$
    $\sigma(v) := i$
    for each $w \in N^+(v)$ **do**
        $inGrad[w] := inGrad[w] - 1$
        if $inGrad[w] = 0$ **then**
            $L_0 := L_0 \cup \{w\}$

return σ

Topologische Sortierung – Beispiel



Topologische Sortierung

Satz 3.9
Algorithmus 3.1 berechnet eine topologische Sortierung in Laufzeit $\mathcal{O}(n+m)$.

Beweis.
Initialisierung:

- Laufzeit $\mathcal{O}(n+m)$ siehe Algo. 2.2

Innere for-each-Schleife:

- Laufzeit $\mathcal{O}(g^+(v))$

Äußere for-Schleife:

- Besucht jeden Knoten einmal
- Benutzt jede Kante einmal Erinnerung: $|R| = \sum_{v \in V} g^+(v)$
- Laufzeit $\mathcal{O}(n+m)$

□

Annahme: G liegt in Adjazenzlisten-Repräsentation vor