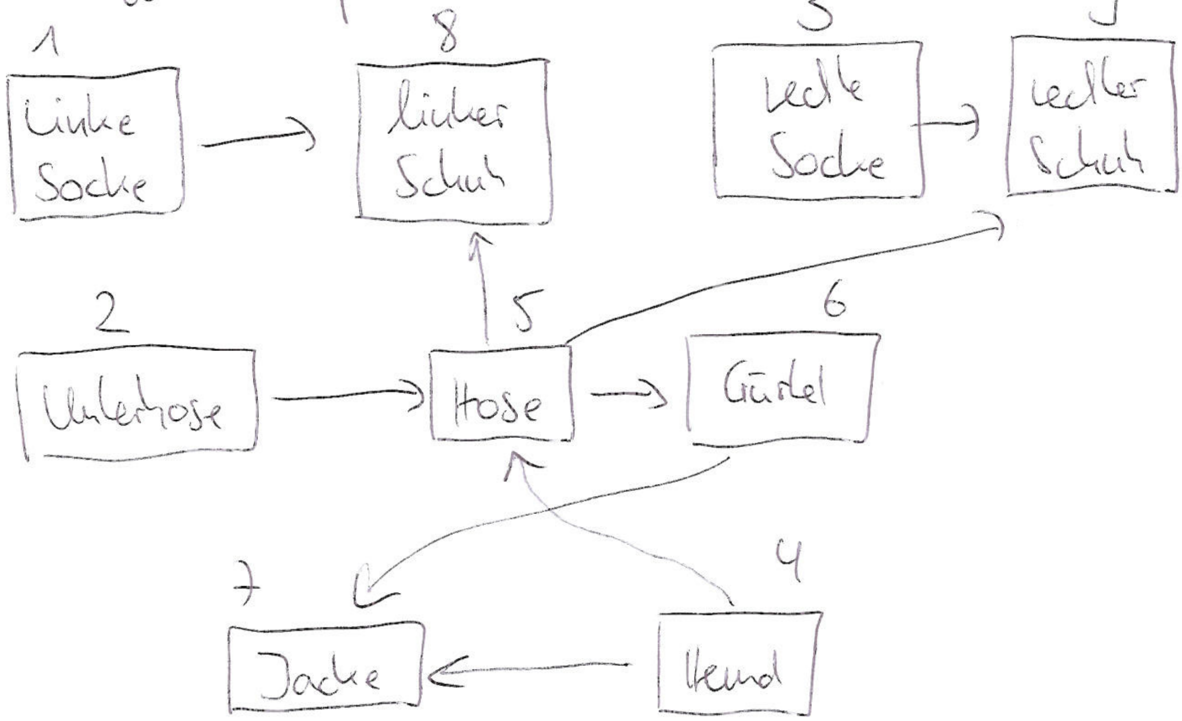
Algorithmen und Datenstrukturen

Vorlesung 3 vom 5. Oktober 2017

Thema: Graphenalgorithmen

Letztes Mal: Modellierung von Abhängigkeiten (etwa beim Anziehen) durch Graphen

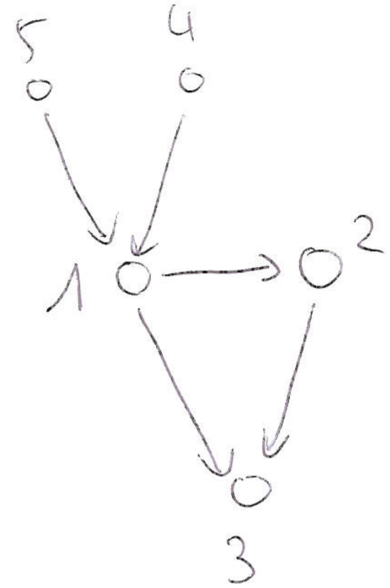


Gesucht: Reihenfolge, die alle Abhängigkeiten berücksichtigt

Alle Kanten zeigen von kleinerer zu größerer Zahl

⇒ Topologische Sortierung

Definition: Folge $v_1, \dots, v_n$ von Knoten ist eine topologische Sortierung von $G = (V, E)$, falls für jede Kante $(v_i, v_j) \in E$ gilt, dass $i < j$, und $V = \{v_1, \dots, v_n\}$.



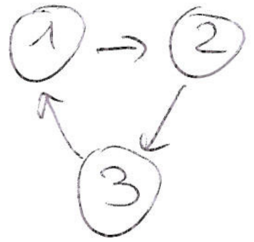
Sort.: 5 4 1 2 3

Anm.: Zahlen links geben nicht Reihenfolge, sondern Namen des Knotens an.

Beobachtung: Topologische Sortierung i. Allg. nicht eindeutig.

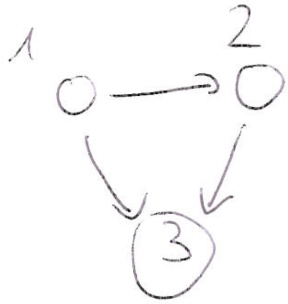
Theorem: Graph topologisch sortierbar $\Leftrightarrow$ kein Zyklus enthalten.

Bsp.:



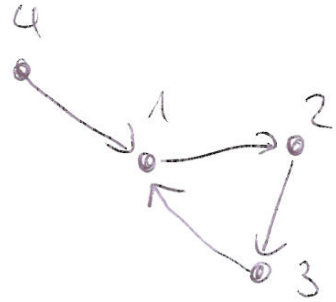
$\langle 1, 2, 3, 1 \rangle$ ist Zyklus.
Klar: Keine topologische Sortierung. 1 muss vor 2 kommen, 2 vor 3, 3 vor 1. ↯

Lemma: Jeder azyklische Graph hat eine Senke (mit Ausgangsgrad 0).



3 ist Senke

Beweis: Betrachte beliebigen azyklischen Graphen $G=(V,E)$. Behauptung: Jeder Weg hat Länge $\leq n$, $n = \#$Knoten von G. Gäbe es einen Weg mit Länge $>n$, würde nach Schubfachprinzip mindestens ein Knoten mehrfach besucht und G hätte einen Zyklus.



Weg $\langle 4, 1, 2, 3, 1 \rangle$ besucht 1 doppelt, Graph hat Zyklus.

Sei $v_1, \dots, v_k$ ein Weg maximaler Länge (existiert immer, da Graph zyklenfrei). Der letzte Knoten dieses Wegs ist eine Senke: Ansonsten gäbe es einen Nachfolger und der Weg wäre nicht maximal lang $\blacksquare$

__Beweis__ (jeder azyklische Graph ist topologisch sortierbar).

Induktion über Knotenanzahl $n \geq 1$.

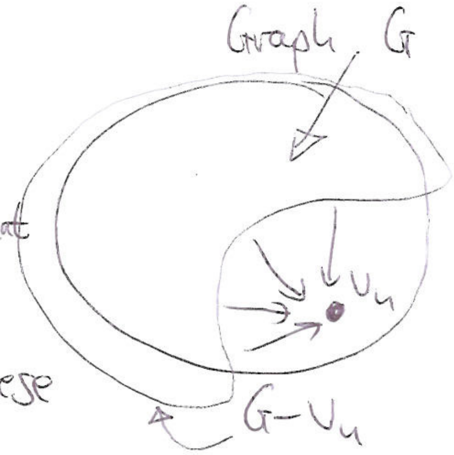
__Induktionsanfang__ $(n=1)$:

 ist topologisch sortierbar.

 nicht, ist aber auch nicht azyklisch (Schleife zählt als Zyklus).

__Induktionshypothese:__ Angenommen, die Aussage gilt für Graphen mit $n-1$ Knoten. $(n \geq 2)$.

__Induktionsschritt__ $(n-1 \to n)$:



Sei v_n Senke.

Entfernen wir v_n, dann hat der verbleibende Graph gemäss Induktionshypothese eine topologische Sortierung $v_1, \dots, v_{n-1}$.

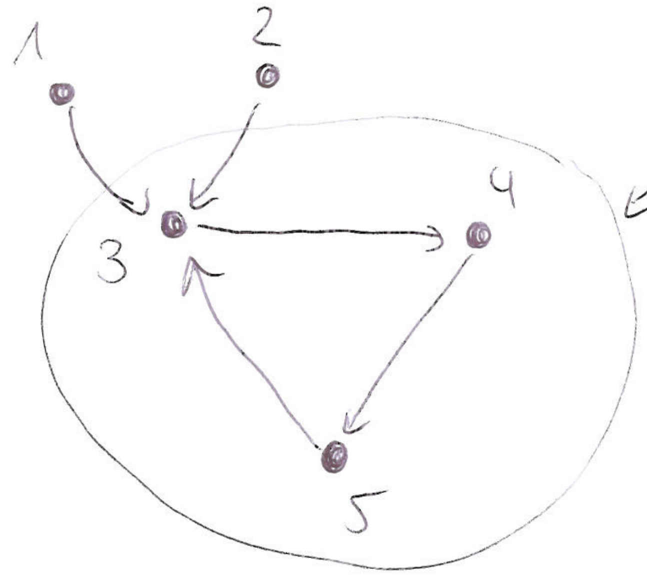
In G kommen alle Kanten zu v_n von einem Knoten v_i mit $i < n$.

Also ist $v_1, \dots, v_n$ eine topologische Sortierung von G. ∎

Zusammenhang

$G=(V,E)$ gerichteter oder ungerichteter Graph.

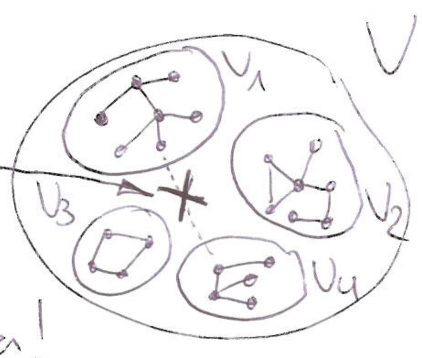
Knotenmenge $W \subseteq V$ ist (stark) zusammenhängend in G, falls für alle Knoten $u, v \in W$ gilt, dass ein Weg von u nach v existiert.



(stark) zusammenhängende Knotenmenge.

Begriff: Graph $G=(V,E)$ zusammenhängend falls V zusammenhängend.

Theorem: In ungerichteten Graphen $G=(V,E)$ existiert Partitionierung $V_1, \dots, V_k$ von V, sodass jeder Teil zusammenhängend ist und jede Kante von G in genau einem Teil V_i verläuft.

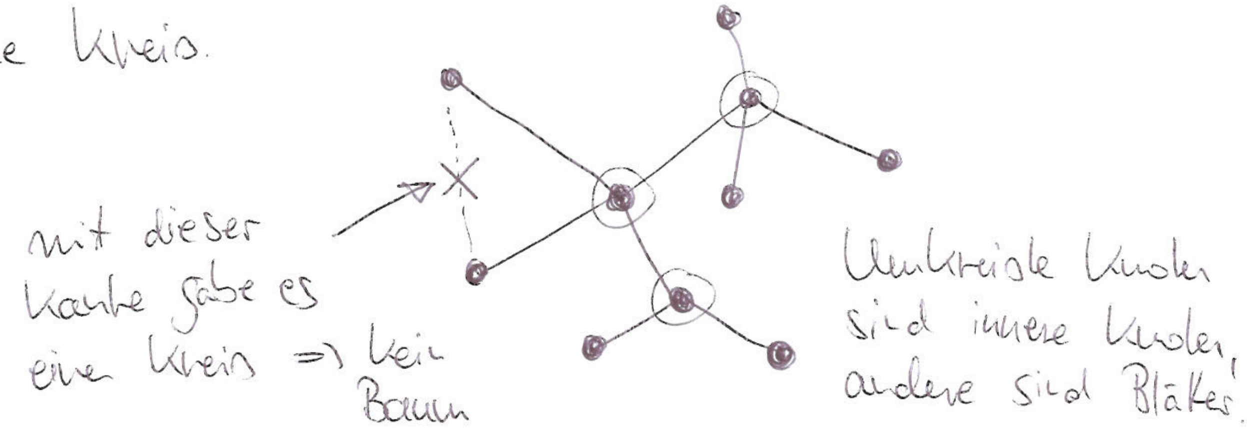


Kanten zwischen verschiedenen Zusammenhangskomponenten V_i und V_j sind verboten!

Bäume

Wichtige Klasse zusammenhängender Graphen.

Bäume sind zusammenhängende, ungerichtete Graphen ohne Kreis.



Wald: Ungerichteter Graph ohne Kreis, Zusammenhangskomp. sind Bäume.

Blatt: Knoten mit Grad 1

Innerer Knoten: Knoten, die kein Blatt sind.

Eigenschaften (hier ohne Beweis)

- Jeder Baum hat mindestens ein Blatt. ($n \geq 2$)
- Jeder Baum hat $n-1$ Kanten. ($n = \#$Knoten)
- Jeder zusammenhängende Graph enthält mindestens einen Baum, einen sog. Spannbaum.

Jetzt: Algorithmen

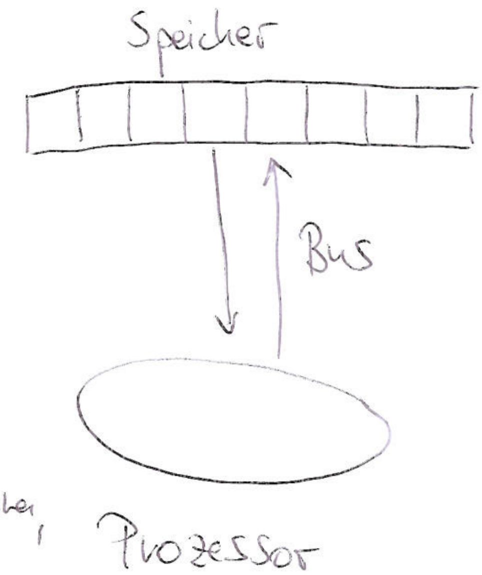
Unterschied zwischen effizienten Algorithmen (z.B. Gale-Shapley), und ineffizienten Algorithmen (z.B. alle Möglichkeiten auszuprobieren).

Genaue Laufzeit eines Computerprogramms hängt von vielen Details ab (Architektur, Speicher, Compiler, ...)

Aber: Diese konkreten Details sind zur Unterscheidung zwischen effizienten und ineffizienten Algorithmen nicht wichtig!

Abstraktion: Vereinfachtes Berechnungsmodell.

Komponenten des Modells:



- Speicher: unbegrenzt viele adressierbare Speicherzellen. Speichern bestimmte Daten, z.B. Bits, Zahlen, ...
- Prozessor: Elementare Operationen wie Rechenoperationen, Vergleichsoperationen, Lese- und Schreibzugriff auf Speicher.
- Bus: Verbindet Prozessor und Speicher.

Berechnungsmodell erlaubt Definition der Laufzeit eines Algorithmus für eine Eingabe: Anzahl der ausgeführten Operationen während der Berechnung.

Beobachtung: Verschiedene Laufzeiten für verschiedene Eingaben möglich. Laufzeit hängt von vielen Details der Eingabe ab. Bsp. Gale-Shapley: Es ist möglich, dass insgesamt nur n Anträge gemacht werden.

$\Rightarrow$ Beschränkung der Laufzeit als Funktion der Eingabegrösse.

Definition: Algorithmus A hat Laufzeit f falls $f(n) =$ maximale Laufzeit von A über alle Eingaben der Größe n.

Eingabegröße kann von mehreren Parametern abhängen, z.B. # Knoten und # Kanten bei Graphen.

Weitere Vereinfachung: Asymptotisches Wachstum reicht. Modell ist sowieso stark vereinfacht, in der Realität ist etwa Multiplizieren viel langsamer als Addieren. Speicherzugriffe dauern auch u.U. unterschiedlich lange

$\Rightarrow$ Konstante Faktoren ignorieren.

Schreibweise: $f(n) \leq O(g(n))$ falls Konstante $C>0$ (unabhängig von n) existiert, sodass $f(n) \leq C \cdot g(n)$ für alle $n \geq 1$ gilt.

Bsp.: $10\,000 \cdot n \leq O(n)$ mit $C = 10\,000$

(hier $f(n) = 10\,000n$, $g(n) = n$)

$1\,000\,000 \cdot n + n^2 \leq O(n^2)$ mit $C = 1\,000\,001$

gilt, da $10^6 \cdot n + n^2 \leq 10^6 \cdot n^2 + n^2 = (10^6+1) \cdot n^2 = C \cdot n^2$ ✓

Weitere Beispiele: $n^2 \not\leq O(10000 \cdot n)$, $n^2 \leq O(n^3)$.

Beispiele für Laufzeiten:

- Gale-Shapley hat Laufzeit $\leq O(n^2)$ ($n = \#$ Männer)
- Naiver Algorithmus zur Färbung mit k Farben hat Laufzeit $\leq O(k^n \cdot (n+m))$ für Graphen mit n Knoten und m Kanten.

$f(n) \leq O(g(n))$: $\exists C > 0. \forall n \geq 1. f(n) \leq C \cdot g(n)$

Frage: Gilt $100 \cdot n + n^2 \leq O(n^2)$?

$C = 1$: "$\forall n \geq 1.\ 100 \cdot n + n^2 \leq 1 \cdot n^2$" gilt nicht.

$C = 2$: " —— " —— $\leq 2 \cdot n^2$" gilt nicht.

$C = 101$: "$\forall n \geq 1.\ 100 n + n^2 \leq 100 n^2$" gilt.

$$100 \cdot n + n^2 \leq 100 n \cdot n + n^2 = 101 n^2.$$

Gilt $n^2 \leq O(1000 \cdot n)$?

Beh.: "$\exists C > 0.\ \forall n \geq 1.\ n^2 \leq C \cdot 1000 \cdot n$" gilt nicht.

Dazu argumentieren wir, dass es für jedes $C > 0$ ein $n \geq 1$ gibt, sodass $n^2 > C \cdot 1000 \cdot n$ gilt.

Gleichheit gilt bei $n^2 = C \cdot 1000 \cdot n \Leftrightarrow n = C \cdot 1000$, also wählen wir $n = C \cdot 1000 + 1$ und erhalten $n^2 > C \cdot 1000 \cdot n$

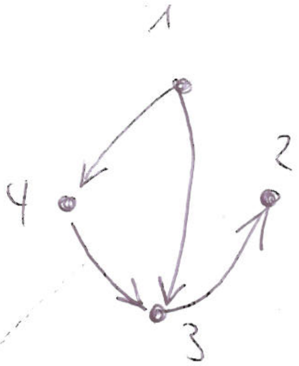
Repräsentation von Graphen

Gerichteter Graph $G=(V,E)$ mit $V=\{v_1, \dots, v_n\}$.
Repräsentation im Speicher?

<u>Definition</u> (Adjazenzmatrix von G):

Tabelle A mit Eintrag $A_{ij} \in \{0,1\}$. Eintrag $1 \Leftrightarrow$ Kante $(v_i, v_j) \in E$

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



Adjazenzmatrix kann im Speicher abgelegt werden. Pro Zelle 1 Bit. Direkte Abfrage, ob Kante von v_i nach v_j existiert, möglich. Laufzeit dieser Operation ist $O(1)$ (konstant).

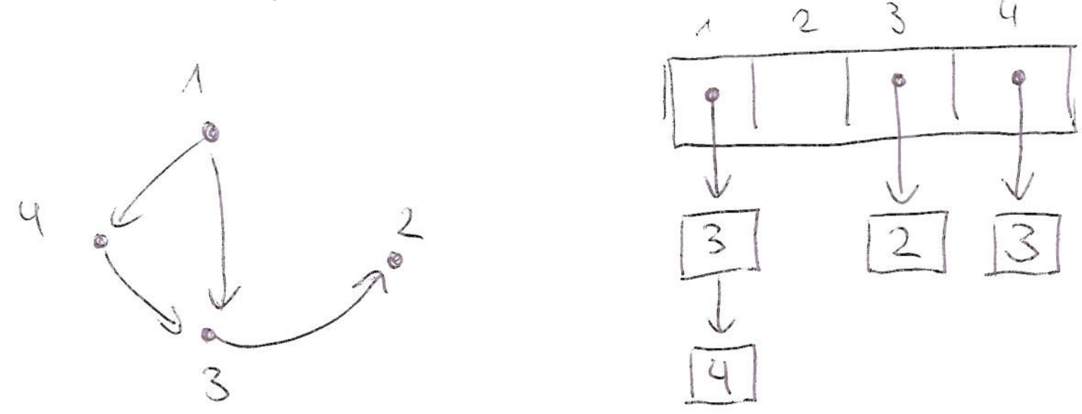
Andere Operation:

- Finde alle Nachfolger eines Knotens v_i. Zeit $O(n)$.

Weitere Darstellung: Adjazenzlisten.

Tabelle mit n Einträgen $(A_1, \dots, A_n)$, sodass

$$A_i = \text{Liste der Nachfolger von } v_i$$



Realisierung der A_i durch sog. einfach verkettete Listen.

Test, ob bestimmte Kante existiert ist $O(\deg^+(v_i))$.

Bestimmung aller Nachfolger von v_i ebenso.

Hinweis: $\deg^+(v)$ ist der Ausgangsgrad eines Knotens v.