

# Algorithmen und Datenstrukturen

Sommersemester 2021

Stefan Näher

Universität Trier

naeher@uni-trier.de

**Vorlesung 13**

20. Mai 2021

## Kapitel 4: Graphen und Graphalgorithmen

**Graphen** werden verwendet um Strukturen aus Objekten zu beschreiben, die untereinander in Relation zu einander stehen.

Die Objekte werden als **Knoten** des Graphen dargestellt und Relationen zwischen den Objekten durch (gerichtete) **Kanten** zwischen den Knoten.

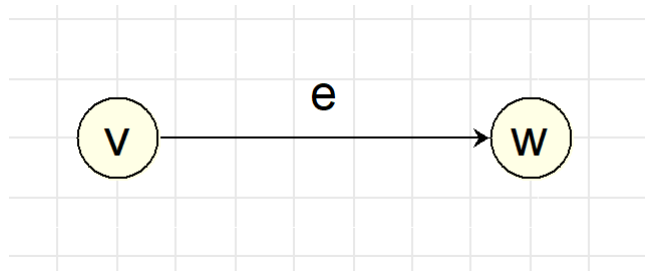
### Beispiele:

- Verkehrsnetzwerke (Knoten = Orte, Kanten = Verkehrswege)
- Soziale Netzwerke (Knoten = Personen, Kanten = Beziehungen)
- Produktionsabläufe (Knoten = Produkte, Kanten = Konstruktionsabläufe)

## Formale Definition

Ein gerichteter Graph  $G = (V, E)$  besteht aus einer Menge von **Knoten**  $V$  und einer Menge von **Kanten**  $E \subseteq V \times V$ .

$e = (v, w) \in E$  heißt Kante von  $v$  nach  $w$

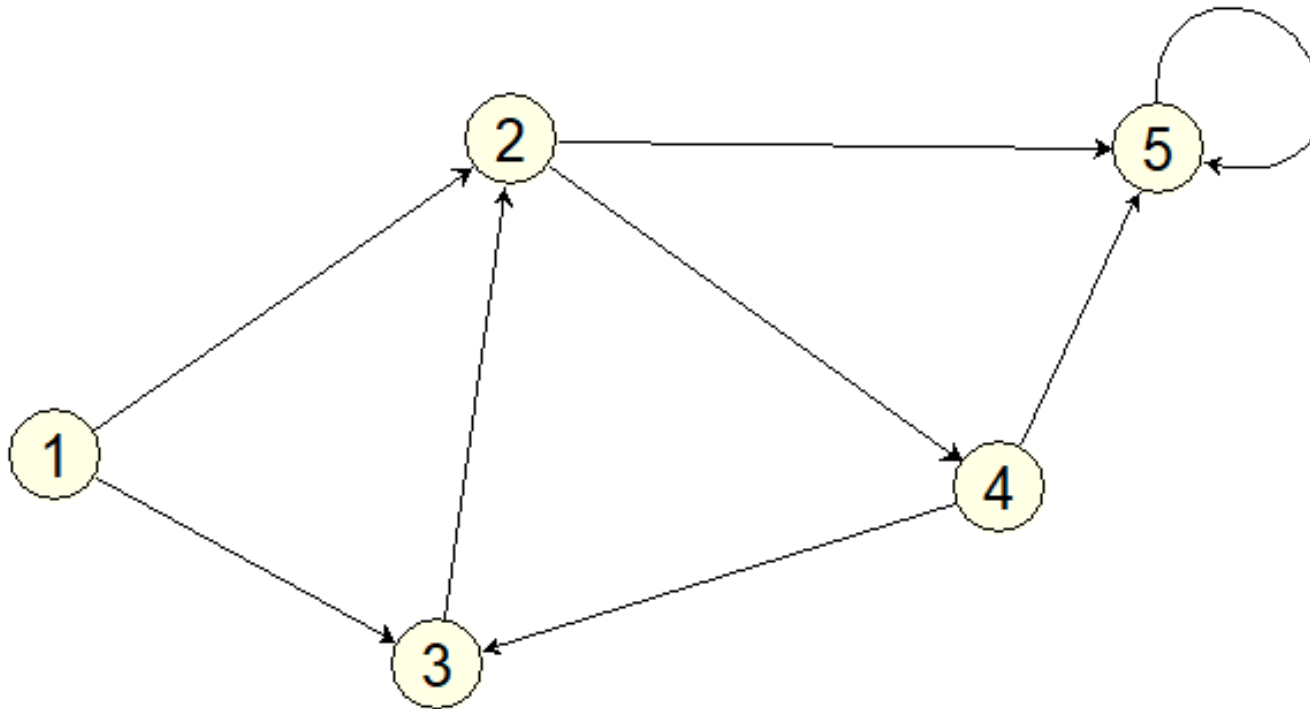


$w$  ist dann **adjazent** (oder benachbart) zu  $v$ .

$v$  heißt *Source*- und  $w$  *Target*-Knoten von  $e$ .

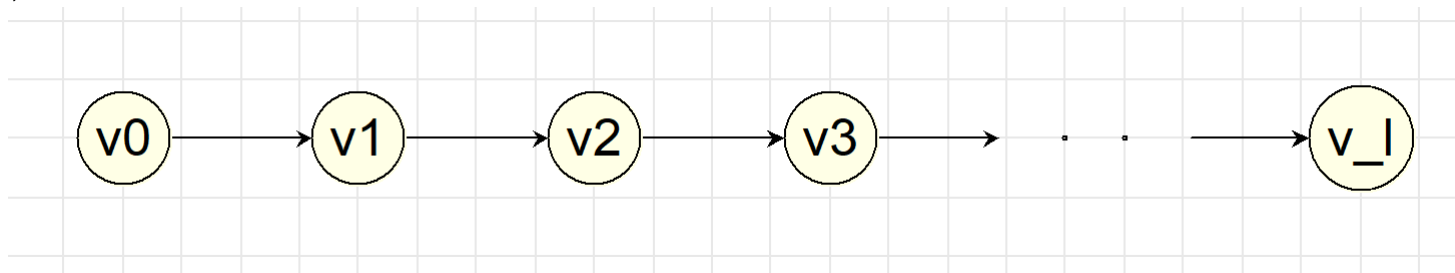
**Ein Beispiel:**  $G = (V, E)$  mit

$V = \{1, 2, 3, 4, 5\}$      $E = \{(1, 2), (1, 3), (2, 5), (2, 4), (3, 2), (4, 3), (4, 5), (5, 5)\}$



Ein **Pfad**  $P$  vom Knoten  $v$  zum Knoten  $w$  ist eine Folge  $v_0, v_1, \dots, v_\ell$  mit

1.  $v_0 = v$  und  $v_\ell = w$
2.  $(v_i, v_{i+1}) \in E$  für  $0 \leq i \leq \ell - 1$



$\ell$  ist die **Länge** von  $P$

Ein Pfad der Länge 0 heißt leerer Pfad.

Ein Pfad heißt **einfach**, wenn  $v_i \neq v_j$  für  $i \neq j$ .

$P$  ist ein **Kreis**, wenn  $v = w$

Ein Graph  $G$  ist **zyklisch**, wenn er einen Kreis enthält, sonst ist er **acyklisch**.

**Ausgangsgrad** (Out-Degree) eines Knotens  $v$

$\text{outdeg}(v)$  = Anzahl der ausgehenden Kanten

$$= |\{ w \in V \mid (v, w) \in E \}|$$

**Eingangsgrad** (In-Degree) eines Knotens  $v$

$\text{indeg}(v)$  = Anzahl der eingehenden Kanten

$$= |\{ u \in V \mid (u, v) \in E \}|$$

**Beobachtung:**

$$\sum_{v \in V} \text{outdeg}(v) = \sum_{v \in V} \text{indeg}(v) = |E|$$

## Darstellung von Graphen im Rechner

Sei  $V = 1, \dots, n$  d.h.  $n$  = Anzahl der Knoten

1. Die **Adjazenzmatrix** ist eine  $n \times n$  Matrix  $a_{i,j} \in \{0, 1\}$  mit

		1	2	3	4	5	
$a_{i,j} =$	1, falls $(i, j) \in E$	1	0	1	1	0	0
		2	0	0	0	1	1
	0, falls $(i, j) \notin E$	3	0	1	0	0	0
		4	0	0	1	0	1
		5	0	0	0	0	1

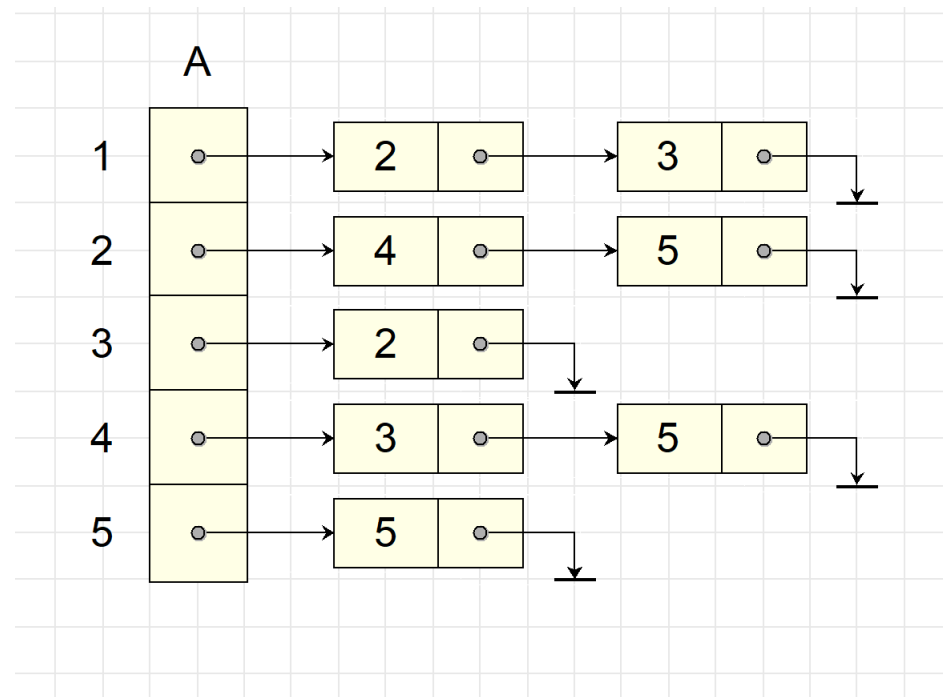
An Position  $i, j$  (Zeile  $i$  und Spalte  $j$ ) steht also genau dann eine **1**, wenn es eine Kante von  $i$  nach  $j$  gibt.

**Nachteil:** Platzbedarf  $\mathcal{O}(n^2)$ .

## 2. Adjazenzlisten

Verwende ein Feld  $A[1..n]$  von Listenköpfen (siehe Bucketsort).

$A[i]$  ist Kopf der einfach-verketteten Liste aller Nachbarknoten von  $i$ .



**Platzbedarf:**  $\mathcal{O}(n + m)$      $n = |V|$  und  $m = |E|$ .



## Adjazenzlisten-Elemente

Realisierung durch eine Struktur oder Klasse (analog zu Listen)

```
class AdjElem {  
    int vertex;      // Knoten  
    AdjElem next;   // Referenz auf nächstes Element  
} ;
```

## Iteration über alle Nachbarknoten eines Knotens

### Pseudo-Code

**forall**  $w \in V$  mit  $(v, w) \in E$  **do**

...

**od**

### Iteration über die Adjazenzliste

$p \leftarrow A[v];$

**while**  $p \neq \text{null}$  **do**

$w \leftarrow p.\text{vertex};$

...

$p \leftarrow p.\text{next};$

**od**

**Laufzeit:**  $\mathcal{O}(1 + \text{outdeg}(v)).$

## Topologisches Sortieren

### Definition

Sei  $G = (V, E)$  ein gerichteter Graph mit  $n = |V|$ .

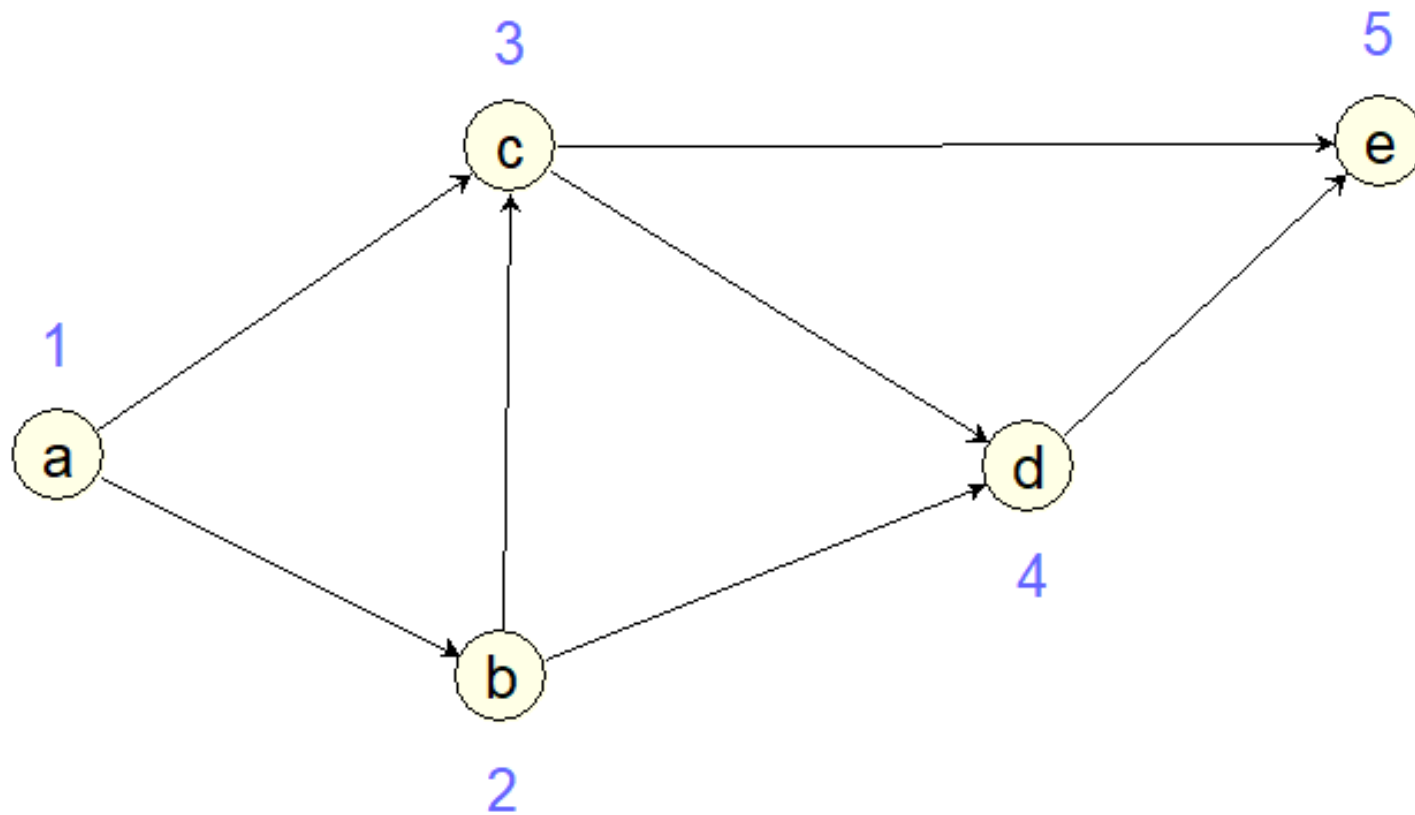
Eine **topologische Sortierung** von  $G$  ist eine *injektive* Abbildung  $\text{ord} : V \longrightarrow \{1, \dots, n\}$  mit  $\text{ord}(v) < \text{ord}(w)$  für alle Kanten  $(v, w) \in E$ .

d.h. *nummeriere* die Knoten so, dass Kanten immer von kleineren zu größeren Nummern verlaufen.

### Veranschaulichung

Ordne die Knoten des Graphen auf einer horizontalen Linie so an, dass alle Kanten von links nach rechts verlaufen.

## Eine topologische Sortierung



**Frage:** Existiert immer eine topologische Sortierung ?

**Beobachtung:** Nicht für zyklische Graphen !

### **Lemma**

Ein gerichteter Graph  $G = (V, E)$  besitzt genau dann eine topologische Sortierung, wenn  $G$  azyklisch ist.

## Beweis des Lemma

1. Topologische Sortierung  $\Rightarrow$  G azyklisch  
folgt aus der **Beobachtung**.
2. G azyklisch  $\Rightarrow$  Topologische Sortierung  
durch **Induktion** über die Zahl der Knoten  $n$ .

**Induktionsanfang:**  $n = 1$

Dann besteht G nur aus einem einzelnen Knoten  $v$  und  $\text{ord}(v) = 1$  ist eine topologische Sortierung.

**Induktionsschritt**

Sei G ein azyklischer Graph mit  $n > 1$  Knoten.

**Behauptung:** G besitzt einen Knoten  $v$  mit  $\text{indeg}(v) = 0$ .

Betrachte den Graphen  $G' = (V', E')$  der durch das Entfernen von  $v$  aus  $G$  entsteht (d.h.  $G' = G \setminus \{v\}$ ).

Dann ist  $G'$  azyklisch und hat  $n - 1$  Knoten.

Nach **Induktionsannahme** besitzt  $G'$  eine topologische Sortierung  $\text{ord}' : V' \longrightarrow \{1, \dots, n - 1\}$ .

Aus  $\text{ord}'$  kann man leicht eine topologische Sortierung  $\text{ord} : V \longrightarrow \{1, \dots, n\}$  für  $G$  konstruieren:

$$\text{ord}(u) = \begin{cases} 1, & \text{falls } u = v \\ \text{ord}'(u) + 1, & \text{falls } u \neq v \end{cases}$$

## Topologisches Sortieren

Ein erster Algorithmus

1.  $\text{count} \leftarrow 0$ ;
2. **while**  $G$  besitzt einen Knoten mit  $\text{indeg} = 0$  **do**
4.    wähle einen solchen Knoten  $v$
3.     $\text{count} \leftarrow \text{count} + 1$ ;
3.     $\text{ord}[v] \leftarrow \text{count}$ ;
4.     $G \leftarrow G \setminus \{v\}$ ;
5. **od**
6. **if**  $\text{count} \neq n$  **then**
7.    ERROR:  $G$  enthält einen Kreis
8. **fi**