Algorithmen & Komplexität

Angelika Steger
Institut für Theoretische Informatik

steger@inf.ethz.ch

Wdh: Wie misst man die Laufzeit??

Anzahl arithmetischer Operationen

bis auf einen konstanten Faktor genau

Worst Case Analyse:

$T_{\mathcal{A}lg}(n) := \max\{T(I) \mid I \text{ ist Eingabe der Länge } n\}$

Beispiel: BubbleSort hat Laufzeit $O(n^2)$

MergeSort hat Laufzeit $O(n \log n)$

Algorithmen-Klassifikation

linearer Algorithmus,	$T_{\mathcal{A}lg}(n) = \mathcal{O}(n)$,
quasi-linearer Algorithmus,	$T_{\mathcal{A}lg}(n) = \mathcal{O}(n \log n)$,
quadratischer Algorithmus,	$T_{\mathcal{A}lg}(n) = \mathcal{O}(n^2)$,
polynomieller Algorithmus,	$T_{\mathcal{A}lg}(n) = \mathcal{O}(n^k)$ für ein $k \in \mathbb{N}$,
exponentieller Algorithmus,	$T_{\mathcal{A}lg}(n) = 2^{\mathcal{O}(n)}$.

Graphenalgorithmen

Szenario

Gegeben:

Flugplan einer Airline

Aufgabe:

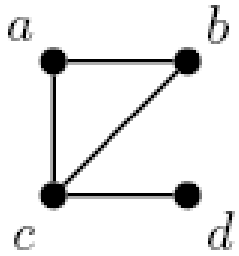
Schreibe ein Programm, das Anfragen der Form

„Kann man von A nach B mit höchstens einmal umsteigen gelangen?“

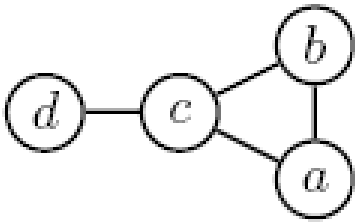
beantwortet.

Modellierung

Graph



oder auch so



Ein Graph G ist ein Tupel (V,E), wobei V eine (endliche) nichtleere Menge von Knoten ist.

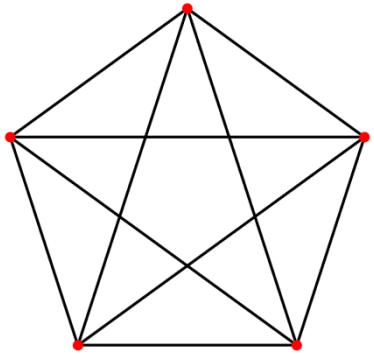
Die Menge E ist eine Teilmenge der zweielementigen Teilmengen von V, also $E \subseteq \{ \{x,y\} : x,y \in V, x \neq y\}$.

Die Elemente der Menge E bezeichnet man als Kanten.

Einige spezielle Graphenklassen (I)

- Vollständiger Graph K_n

  $\#edges(K_n) = n(n-1)/2$



K_5

- Kreis C_n

  $\#edges(C_n) = n$

C_6

- Pfad P_n

  $\#edges(P_n) = n$

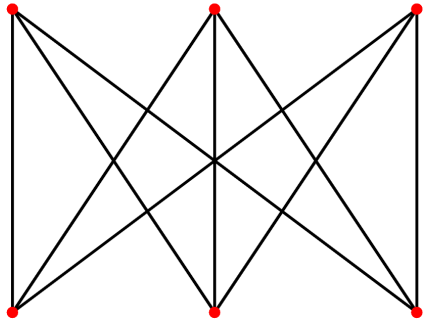
  $\#vertices(P_n) = n+1$

P_4

Einige spezielle Graphenklassen (II)

- Vollständiger bipartiter Graph $K_{n,n}$

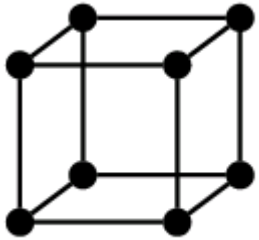
$\#edges(K_{n,n}) = n^2$



$K_{3,3}$

- Hyperwürfel Q_d

$\#edges(Q_d) = ½\ d\ 2^d$



Q_3

Nachbarschaft, Grad

Definitionen: Graph $G=(V,E)$, $v \in V$

- Nachbarschaft: $\Gamma(v) = \{u \in V \mid \{u,v\} \in E\}$
- Grad: $\deg(v) = |\Gamma(v)|$
- G heisst *k-regulär*, wenn $\deg(v)=k \quad \forall v \in V$
- Sprechweise für $e=\{u,v\} \in E$:

  - *u* und *v* sind adjazent, und
  - *u* und *e* sind inzident.

Grad

Lemma:

Für jeden Graphen $G=(V,E)$ gilt

$$\sum_{v \in V} \deg(v) = 2\,|E|$$

Beweis:

„Doppeltes Abzählen“

Pfade, Zusammenhang

Definition: Sei $G=(V,E)$ ein Graph

- Ein *Pfad* in G ist eine Folge $(v_0,\ldots,v_n)$ mit

  $\{v_i,v_{i+1}\} \in E$ und $v_i \neq v_j$ für $i \neq j$

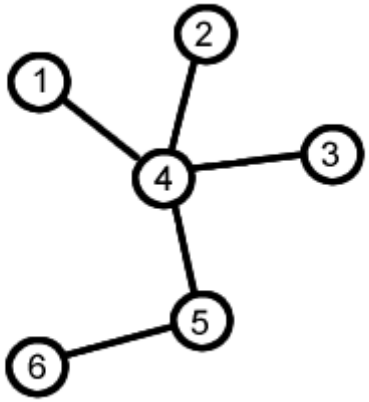
- G heisst *zusammenhängend*, wenn für alle $u,v \in V$ ein u-v-Pfad existiert.

- Die zusammenhängenden Teile von G heissen die *Komponenten* von G.

Definition:

Einen zusammenhängenden, kreisfreien Graphen $T=(V,E)$ nennt man einen Baum.

Ein Knoten $v \in V$ mit $\deg(v) = 1$ heisst Blatt.



Eigenschaften von Bäumen

Satz: Jeder Baum $T=(V,E)$ mit $|V| \geq 2$ enthält mindestens zwei Blätter.

Eigenschaften von Bäumen

Satz: Ist $T=(V,E)$ ein Baum mit $|V| \geq 2$, so ist der Graph, der durch Entfernen eines Blattes entsteht, ebenso ein Baum.

Eigenschaften von zshgd Graphen

Lemma: $G=(V,E)$ zshgd. Graph, C Kreis in G

Dann gilt: $G_e=(V,E\backslash e)$ zshgd. $\forall e \in C$

Eigenschaften von Bäumen

Lemma: Für jeden zshgd. Graphen $G=(V,E)$ gilt

$$|E| \geq |V|-1$$

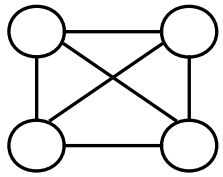
Eigenschaften von Bäumen

Satz: Ein zshgd. Graph $G=(V,E)$ ist genau dann ein Baum, wenn $|E|=|V|-1$.

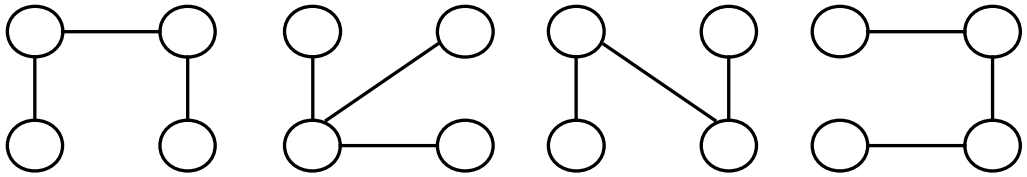
Teilgraph, Spannbaum

Graph $G=(V,E)$

- $H=(V_H,E_H)$ mit $V_H \subseteq V$, $E_H \subseteq E$ heisst ein *Teilgraph* (engl. *subgraph*) von G.
- Gilt für einen Teilgraph $V=V_H$, nennt man ihn einen *spannenden Teilgraphen.*
- Ist H zusätzlich ein Baum und G zshgd, nennt man ihn einen *Spannbaum von G.*



G



4 Spannbäume von G

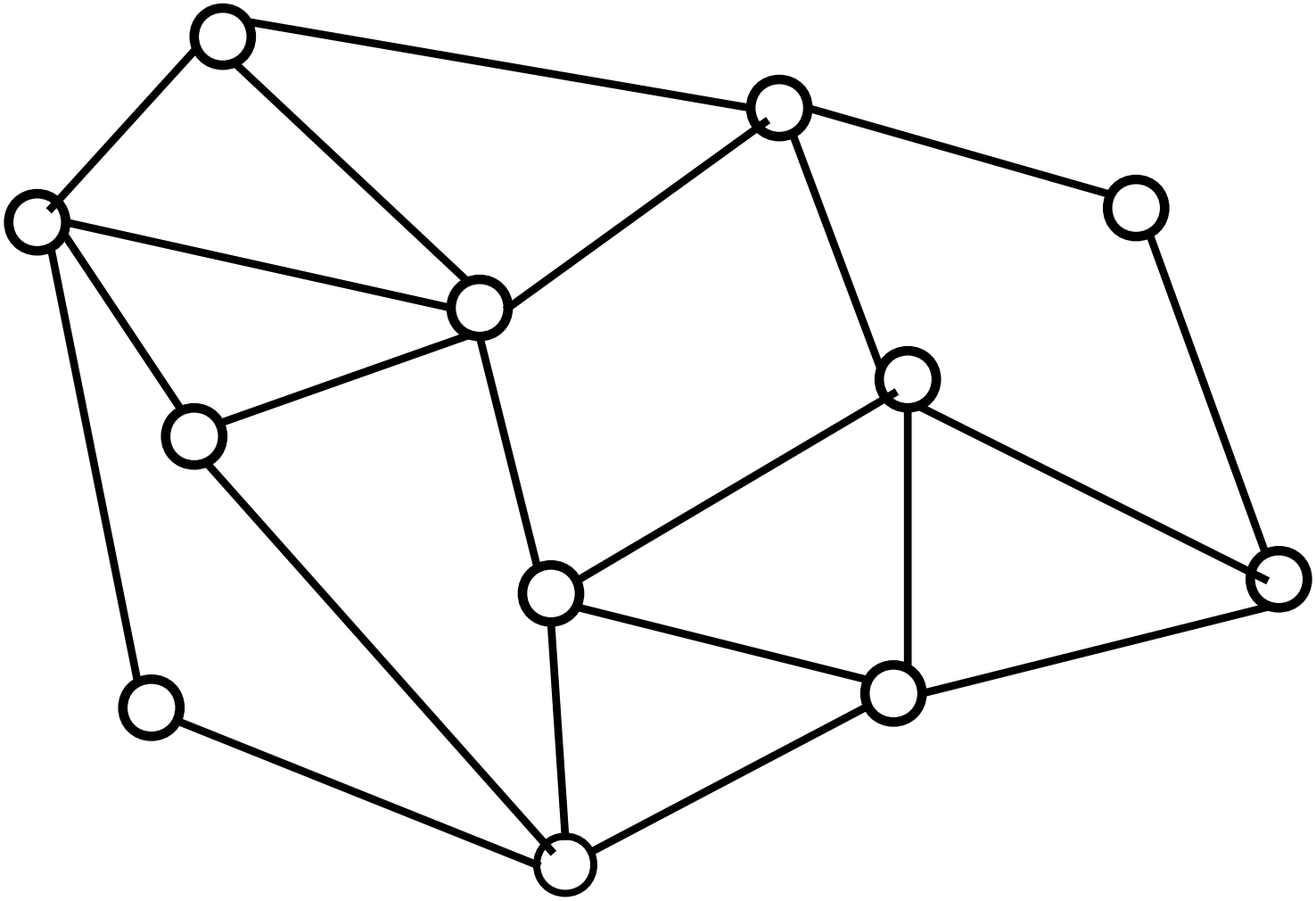
Breitensuche, Tiefensuche

Wir besprechen nun zwei grundlegende Verfahren, alle Knoten eines Graphen zu durchlaufen

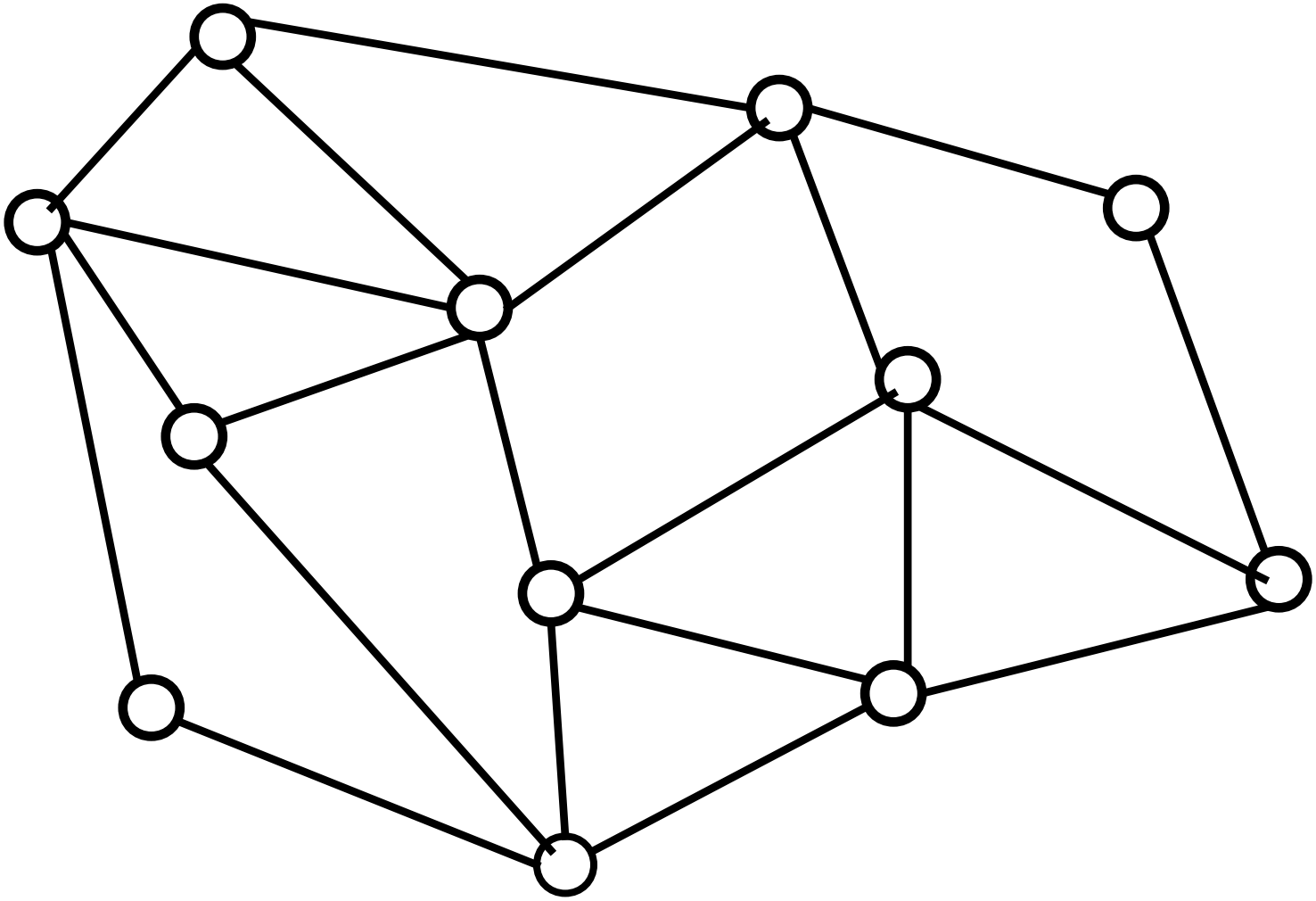
Breitensuche	Tiefensuche
(„*breadth first search*“, BFS)	(„*depth first search*“, DFS)

⇒ wichtige Bausteine von „fortgeschrittenen“ Graphenalgorithmen

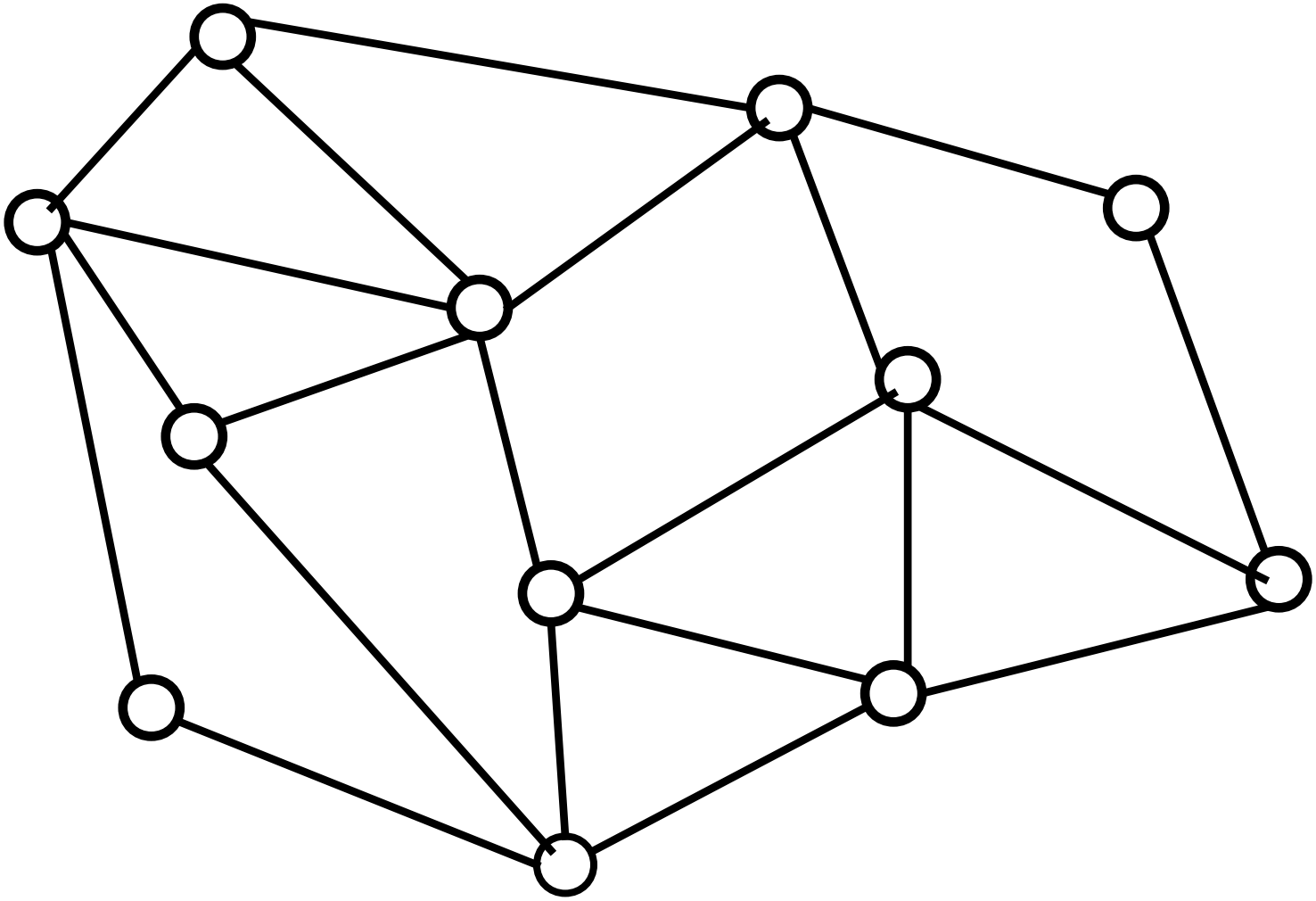
BFS und DFS - Beispiel



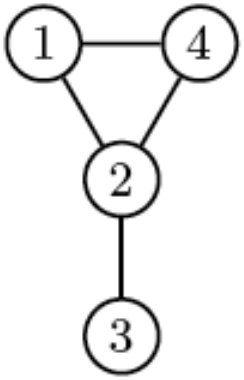
BFS



DFS

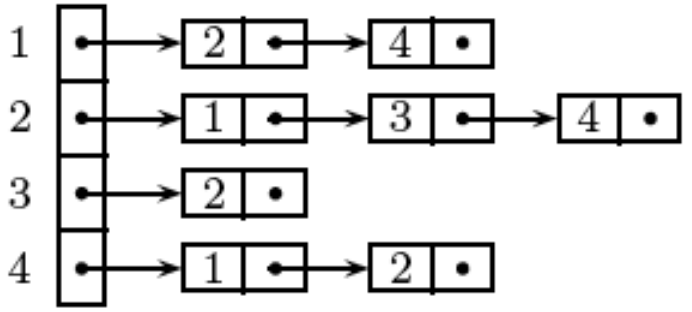


Speicherung eines Graphen



$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Adjazenzmatrix



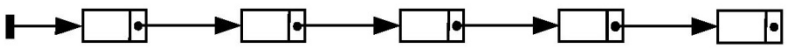
Adjazenzlisten

Elementare Datenstrukturen

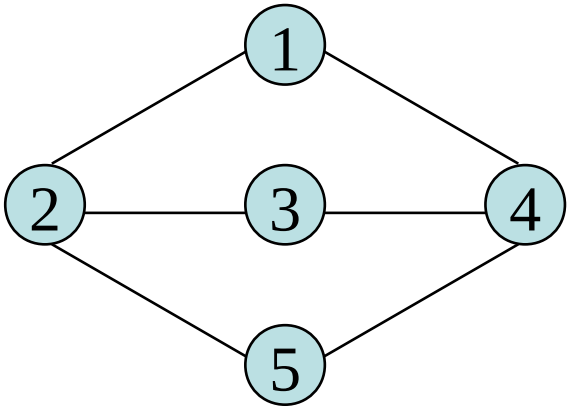
- Felder (engl. arrays):

  a[i,j,k]

- Listen:



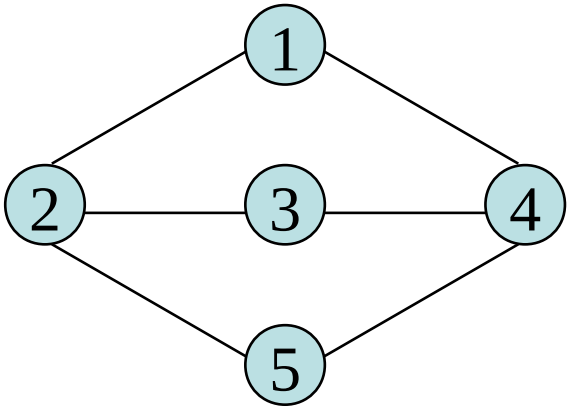
Adjazenzmatrix



Speicheraufwand: $O(|V|^2)$

	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	0	1
3	0	1	0	1	0
4	1	0	1	0	1
5	0	1	0	1	0

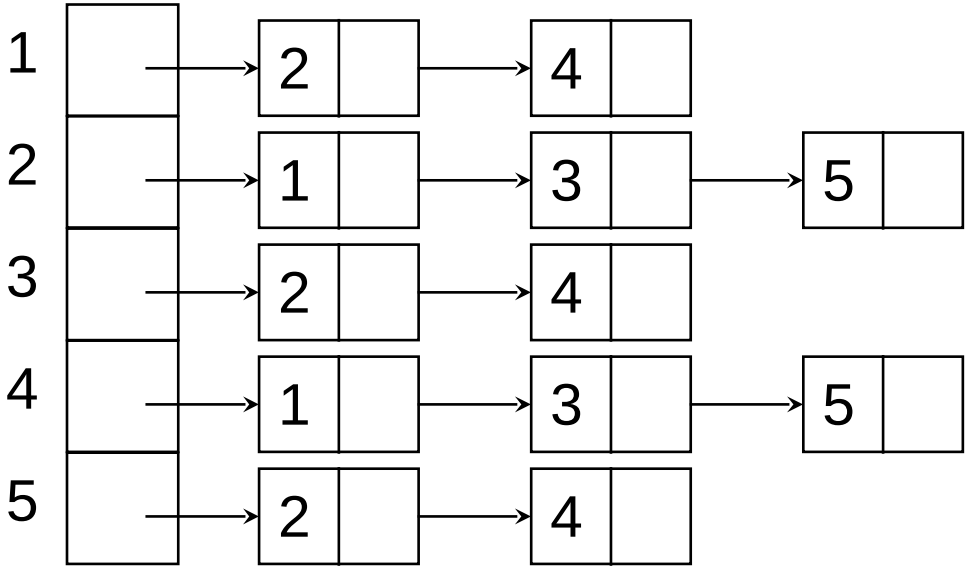
Adjazenzliste



Speicheraufwand: $O(|V|+|E|)$

Ein Nachteil von Adjazenzlisten:

Test auf $\{u,v\} \in E$ dauert $O(\min\{deg(u),deg(v)\})$

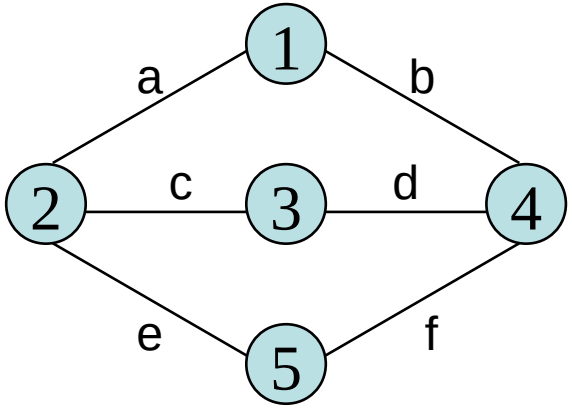


Inzidenzmatrix

Vor allem theoretische Bedeutung: *Inzidenzmatrizen*

⇒ Beziehung zwischen Linearer Algebra und Graphentheorie

Lineare Abhängigkeit, Basis, … ⇔ Kreise, Bäume, …



	a	b	c	d	e	f
1	1	1	0	0	0	0
2	1	0	1	0	1	0
3	0	0	1	1	0	0
4	0	1	0	1	0	1
5	0	0	0	0	1	1

Breitensuche, Tiefensuche

Wir besprechen nun zwei grundlegende Verfahren, alle Knoten eines Graphen zu durchlaufen

Breitensuche („*breadth first search*“, BFS)

Tiefensuche („*depth first search*“, DFS)

Datenstrukturen:

queue

stack