IT Inhalt

6. Objektorientierung

- Grundidee und Motivation
- Klassen
- Objekte
- Datenkapselung

7. Datenstrukturen

- Array
- Liste
- Stack
- Queue
- Hash-Tabelle
- Graph
- Baum
- Haufen

Wiederholung

Vorlesung Digitaltechnik

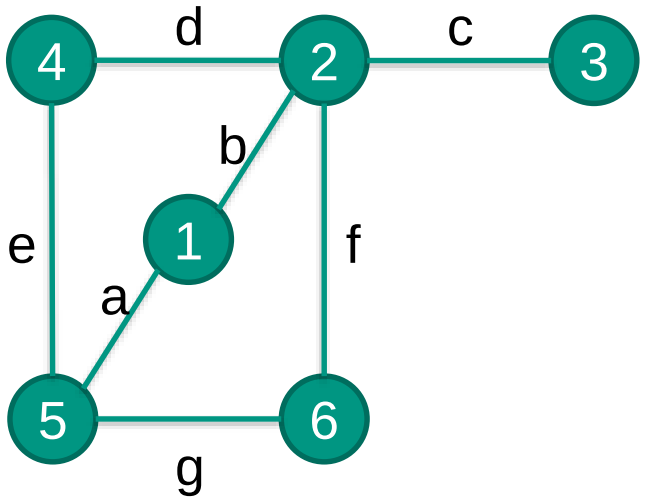
Graphen und Bäume

Graphentheorie: Abstrakter Graph

- Formale mathematische Beschreibung:
  - Abstraktion von der Bedeutung der Darstellungselemente:
    → Verknüpfung mit der Begriffswelt der Mengen und Relationen
  - Graphen können (unabhängig von der Darstellung) durch **zwei Mengen** und **eine Abbildung** beschrieben werden:
    - **V = Menge der Knoten**
    - **E = Menge der Kanten**
    - **Φ (e) ordnet jeder Kante e $\in$ E zwei Knoten aus V zu**
      → diejenigen, die durch die Kante e verbunden sind
  - G (V, E, Φ) wird abstrakter Graph genannt

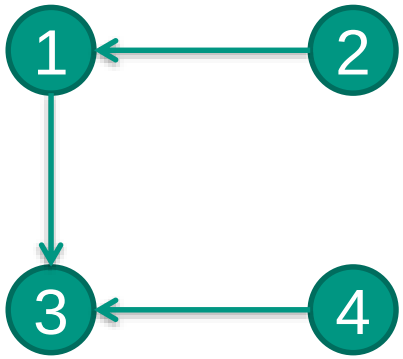
Graphentheorie: Abstrakter Graph

- **Beispiel:**
  - V = { 1, 2, ..., 6 }
  - E = { a, b, ..., g }
  - Φ: E $\rightarrow$ { v, w } v, w $\in$ V
    - Φ (a) = { 1, 5 }
    - Φ (b) = { 1, 2 }
    - Φ (c) = { 2, 3 }
    - Φ (d) = { 2, 4 }
    - Φ (e) = { 4, 5 }
    - Φ (f) = { 2, 6 }
    - Φ (g) = { 5, 6 }



Graphentheorie: Gerichteter Graph

- **Gerichteter Graph:**
  - Für manche Probleme benutzt man auch **gerichtete Graphen**
    → **Kanten** haben eine **festgelegte Richtung**
  - Bei gerichteten Graphen gilt:
    Φ bildet **Kanten** auf geordnete **Knoten-Tupel** aus **V x V** ab
  - Ein gerichteter Graph muss mindestens eine Kante zwischen zwei Knoten (g,h) besitzen, so dass keine Kante in umgekehrter Richtung (h,g) existiert
  - Gerichtete Graphen werden auch als **Digraphen** bezeichnet

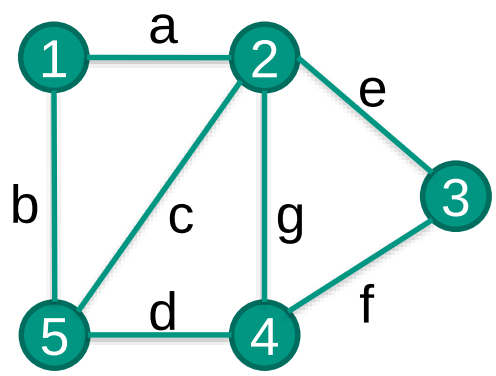


Graphentheorie: Inzident
(Kanten orientiert)

- Verbindet die Kante e die Knoten g und h, so sagt man: „Die Kante e ist inzident zu g bzw. zu h“ und schreibt:
  - Φ (e) = (g, h) für gerichtete Graphen
  - Φ (e) = { g, h } = { h, g } für ungerichtete Graphen
- Φ wird daher Inzidenzabbildung genannt
- **Inzidenzmatrix:**
  - Eine **Inzidenzmatrix** ist eine n x m-Matrix (Kante x Knoten)
    - Der Eintrag in der *i*-ten Spalte und *j*-ten Zeile gibt an, ob die *i*-te Kante den *j*-ten Knoten enthält.
  - Ist der Knoten ein Endknoten, so erhält das entsprechende Matrixelement den Wert 1, sonst 0 (oder leer).
    - Remark: *Bei ungerichteten Graphen verbindet eine Kante 2 Endknoten.*
  - Bei gerichteten Graphen wird ein Ausgangsknoten für eine Kante mit -1 gekennzeichnet.
    - Der Endknoten wird weiter mit 1 gekennzeichnet.

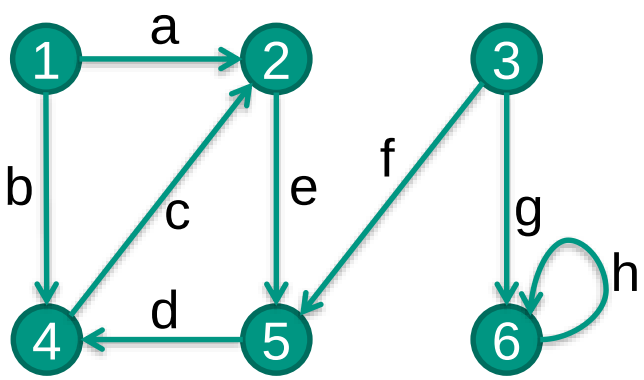
Graphen: Darstellung Inzidenzmatrix

- Inzidenzdarstellung für einen ungerichteten Graphen:



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

- Inzidenzdarstellung für einen gerichteten Graphen:



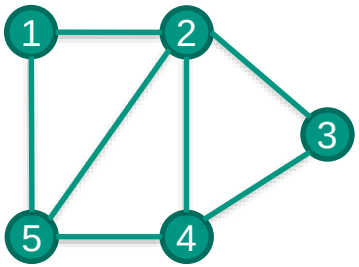
	a	b	c	d	e	f	g	h
1	-1	-1						
2	1		1		-1			
3						-1	-1	
4		1	-1	1				
5				-1	1	1		
6							1	1

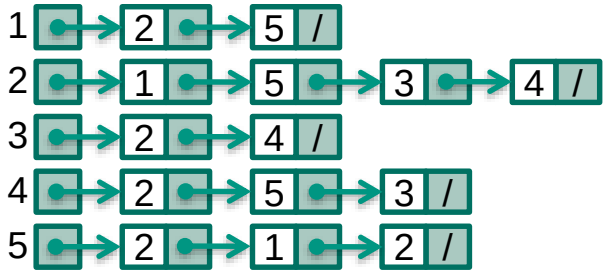
Graphentheorie: Adjazent
(Knoten orientiert)

- Zwei durch eine Kante e verbundene Knoten g und h heißen adjazent zur Kante e
- **Adjazenzmatrix (Nachbarschaftsmatrix)**
  - speichert, welche Knoten des Graphen durch eine Kante verbunden sind.
  - besitzt für jeden Knoten eine Zeile und eine Spalte, woraus sich für *n* Knoten eine n x n-Matrix ergibt.
- Gerichteter Graph
  - Gibt es eine Kante von Knoten g zu Knoten h, wird in der Matrix in der g-ten Zeile an der h-ten Stelle eine 1 eingetragen.
- Ungerichteter Graph
  - Es muss eigentlich nur die Hälfte gespeichert werden, da sich die andere Hälfte durch Spiegelung ergibt.

Graphen: Darstellung Adjazenzmatrix
(Knoten orientiert)

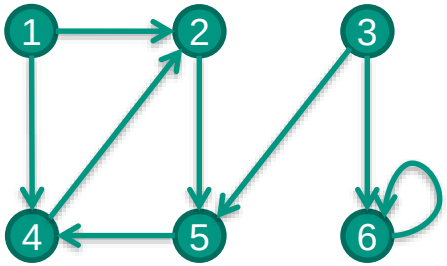
- Adjazenzdarstellung für einen ungerichteten Graphen:

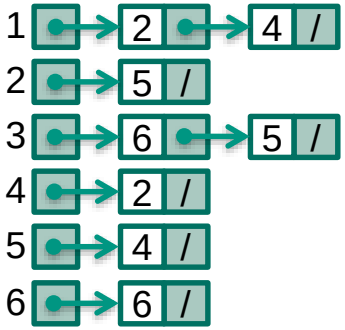




	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

- Adjazenzdarstellung für einen gerichteten Graphen:





	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

Graphentheorie: Isomorphie

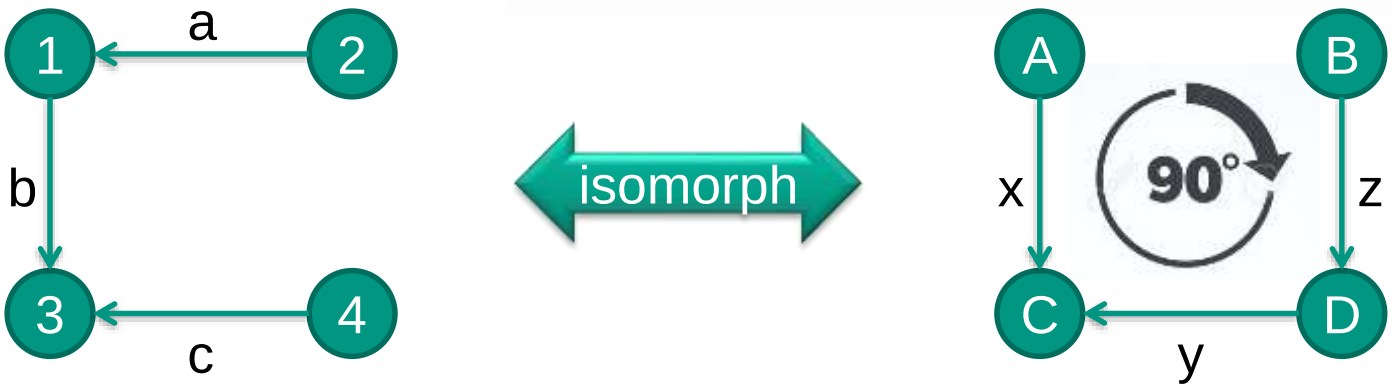
- Vergleichbarkeit zweier Graphen:
  → es werden Abbildungen zwischen Knoten und Kanten gesucht, so dass die Inzidenzbeziehungen erhalten bleiben

- Sind diese Zuordnungen bijektiv (eineindeutig)
  → so wird der Graph isomorph genannt

- Isomorphie von Graphen:
  → Strukturen isomorpher Graphen sind gleich
  → wichtige Eigenschaft in der Vergleichbarkeit und formalen Verifikation digitaltechnischer Schaltungen und Systeme

Isomorphie: Beispiel

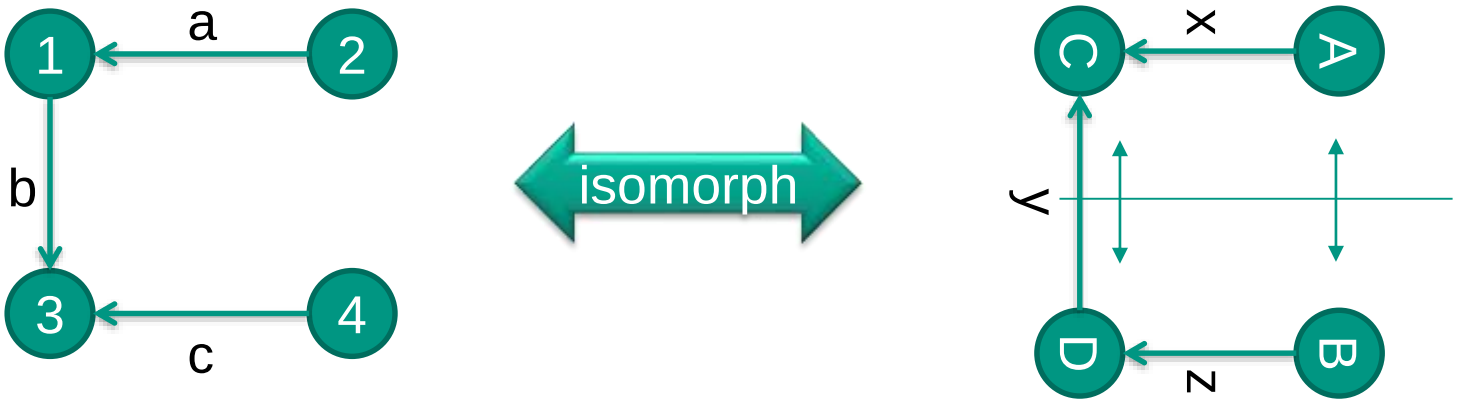
- **Inzident:**
  - Verbindet die Kante e die Knoten g und h, so sagt man: „e ist inzident zu g bzw. zu h“ und schreibt:
    - Φ (e) = (g, h) für gerichtete Graphen
    - Φ (e) = { g, h } = { h, g } für ungerichtete Graphen



- **Zuordnung der Knoten φ:** (1,2,3,4) $\Rightarrow$ (D,B,C,A)
- **Zuordnung der Kanten ψ:** (a,b,c) $\Rightarrow$ (z,y,x)
- **Inzidenzbeziehungen:**

Φ(a) = (2,1)
Φ(b) = (1,3)
Φ(c) = (4,3)

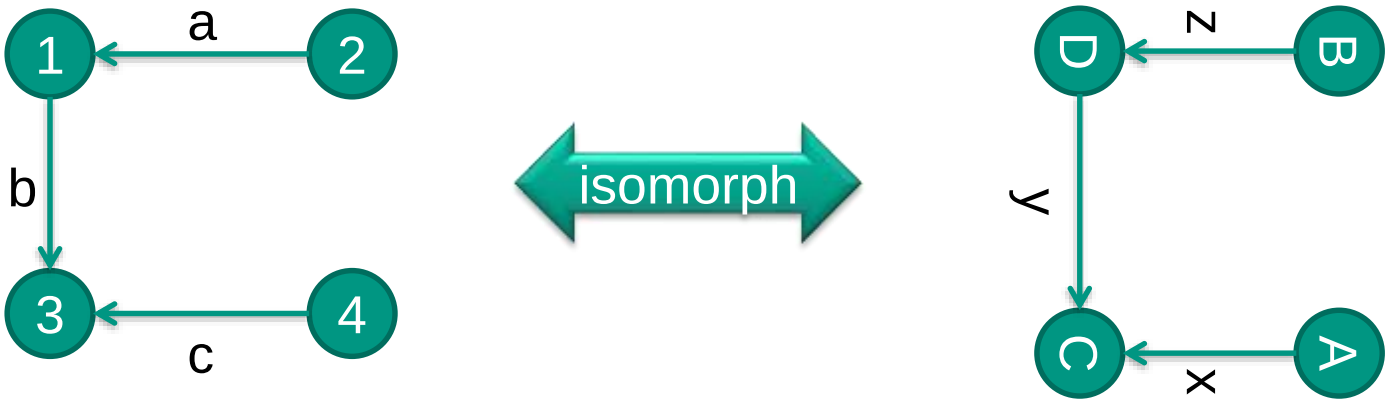
Isomorphie: Beispiel



- **Zuordnung der Knoten φ:** (1,2,3,4) ⇒ (D,B,C,A)
- **Zuordnung der Kanten ψ:** (a,b,c) ⇒ (z,y,x)
- **Inzidenzbeziehungen:**

$$\Phi(a) = (2,1) \quad \Leftrightarrow \quad \Phi(\psi(a)) = \Phi(z) = (B,D) = (\varphi(2), \varphi(1))$$
$$\Phi(b) = (1,3) \quad \Leftrightarrow \quad \Phi(\psi(b)) = \Phi(y) = (D,C) = (\varphi(1), \varphi(3))$$
$$\Phi(c) = (4,3) \quad \Leftrightarrow \quad \Phi(\psi(c)) = \Phi(x) = (A,C) = (\varphi(4), \varphi(3))$$

Isomorphie: Beispiel



- **Zuordnung der Knoten φ:** (1,2,3,4) ⇒ (D,B,C,A)
- **Zuordnung der Kanten ψ:** (a,b,c) ⇒ (z,y,x)
- **Inzidenzbeziehungen:**

$$\Phi(a) = (2,1) \quad \Leftrightarrow \quad \Phi(\psi(a)) = \Phi(z) = (B,D) = (\varphi(2), \varphi(1))$$
$$\Phi(b) = (1,3) \quad \Leftrightarrow \quad \Phi(\psi(b)) = \Phi(y) = (D,C) = (\varphi(1), \varphi(3))$$
$$\Phi(c) = (4,3) \quad \Leftrightarrow \quad \Phi(\psi(c)) = \Phi(x) = (A,C) = (\varphi(4), \varphi(3))$$

Graphentheorie: Begriffe

Zusammenhängende Graphen

Streng zusammenhängende Graphen

Lokale Eigenschaften von Graphen:

Aufspannender Baum

- Definition: Aufspannender Baum
  - Es gilt: bei jedem zusammenhängenden Graphen kann man alle Zyklen durch gezieltes Entfernen von Kanten auflösen
    (→ Kruskal Algorithmus: Minimal Aufspannender Baum, MAB)
    ohne dass der Graph in zwei Teilgraphen zerfällt
  - Auf diese Weise erhält man zu jedem beliebigen Graphen einen zugehörigen aufspannenden Baum

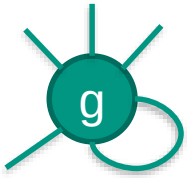
40 | Informationstechnik
Kapitel 7: Datenstrukturen

Institut für Technik der Informationsverarbeitung (ITIV)
Prof. Dr.-Ing. Eric Sax, © 2017

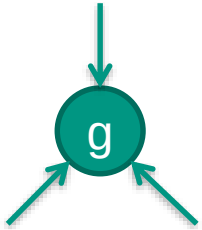
s. Backup

Grad eines Knotens

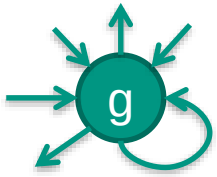
- Betrachtet man einen **Knoten**, so wird die **Anzahl** der damit **inzidenten Kanten** als **Grad d(g)** des Knotens bezeichnet
- Bei **gerichteten Graphen** unterscheidet man zusätzlich
  - abgehende Kanten $\Rightarrow$ **Ausgangsgrad $d^+(g)$**
  - von ankommenden Kanten $\Rightarrow$ **Eingangsgrad $d^-(g)$**
- Beispiele:



d(g) = 6



$d^-(g) = 3$



$d^+(g) = 3$
$d^-(g) = 4$

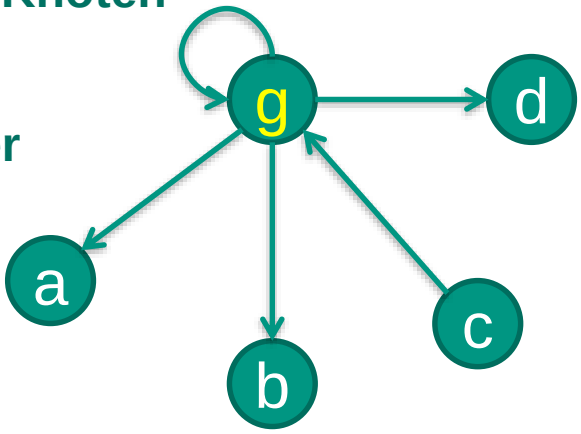
Nachbarschaft von Knoten

- **Unmittelbare Nachbarschaft** von Knoten
  - Sind **zwei Knoten** durch **eine Kante** verbunden
    → so sind die Knoten **unmittelbar benachbart**
  - **Menge** der **unmittelbar benachbarten Knoten**
    → wird mit **V‘(g)** bezeichnet
  - Bei **gerichteten Graphen** gilt:
    → die Menge der benachbarten Knoten wird **entsprechend** der Richtungen der Kanten eingeteilt in:
      → **unmittelbare Vorgänger** $V'_1(g)$ und
      → **unmittelbare Nachfolger** $V'_2(g)$
- **Mittelbare Nachbarschaft** von Knoten
  - Schwächt man die Forderung so ab, dass nur eine **Folge** von **Kanten zwischen zwei Knoten** existieren muss
    → so sind die Knoten **mittelbar benachbart**
  - Nützlich ist die mittelbare Nachbarschaft besonders bei gerichteten Graphen bzgl. der Unterscheidung in
    → **mittelbare Vorgänger** und
    → **mittelbare Nachfolger**

Übung Nachbarschaft

- Bestimmen Sie für den gegebenen Graphen

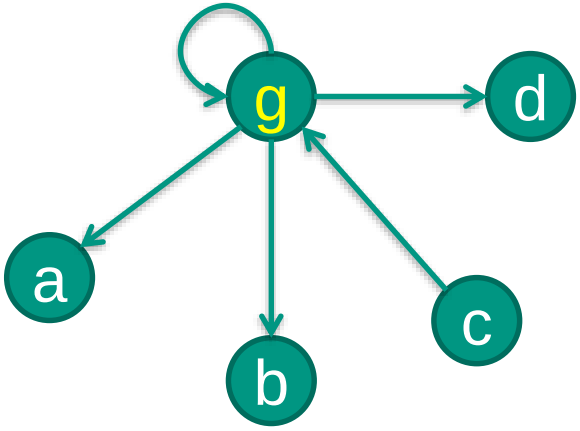
- $V'(g)$ = **Menge** der **unmittelbar benachbarten Knoten**
- Vorgänger: $V'_1(g)$ = **unmittelbare Vorgänger**
- Nachfolger: $V'_2(g)$ = **unmittelbare Nachfolger**
- $d^-(g) = |V'_1(g)|$ = **Eingangsgrad**
- $d^+(g) = |V'_2(g)|$ = **Ausgangsgrad**
- $d(g) = d^-(g) + d^+(g)$ = **Grad des Knotens**



Übung Nachbarschaft

- Bestimmen Sie für den gegebenen Graphen

- $V'(g) = \{ a, b, c, d, g \}$
- Vorgänger: $V'_1(g) = \{ c, g \}$
- Nachfolger: $V'_2(g) = \{ a, b, d, g \}$
- $d^-(g) = |V'_1(g)| = 2$
- $d^+(g) = |V'_2(g)| = 4$
- $d(g) = d^-(g) + d^+(g) = 6$



Kantenfolgen in Graphen

- **Definition:** **Kantenprogression** der **Länge n**
  → **endliche Folge** von **n nicht** notwendigerweise **verschiedenen Kanten (e^i),** die n + 1 nicht notwendigerweise verschiedene Knoten (g^i) verbinden

  $$\Phi(e^i) = (g^i, g^{i+1}) \quad \text{für } i = 1, 2, ..., n$$

- Sind in einer **Kantenprogression alle Knoten (g^i)** voneinander **verschieden** und damit auch alle Kanten, so heißt sie **einfach.**
- Je nachdem, ob der Anfangsknoten gleich dem Endknoten ist ($g^1 = g^{n+1}$), oder ob es sich um einen gerichteten Graphen handelt, verwendet man unterschiedliche Bezeichnungen (s. nächste Folie)

Spezielle Kantenfolgen in Graphen

Ungerichteter Graph	
$g^1 \neq g^{n+1}$	$g^1 = g^{n+1}$
offene Kanten-progression	geschlossene Kanten-progression
Sind alle Kanten einer Progression voneinander verschieden	
Ketten-progression	geschlossene Kantenzug-progression
Berücksichtigt man die Kanten einer Progression ohne Ordnung	
Kette	geschlossener Kantenzug

Spezielle Kantenfolgen in Graphen

Ungerichteter Graph		Gerichteter Graph	
$g^1 \neq g^{n+1}$	$g^1 = g^{n+1}$	$g^1 \neq g^{n+1}$	$g^1 = g^{n+1}$
offene Kanten-progression	geschlossene Kanten-progression	offene Kanten-progression	geschlossene Kanten-progression
Sind alle Kanten einer Progression voneinander verschieden			
Ketten-progression	geschlossene Kantenzug-progression	Weg-progression	Zyklus-progression
Berücksichtigt man die Kanten einer Progression ohne Ordnung			
Kette	geschlossener Kantenzug	Weg	Zyklus

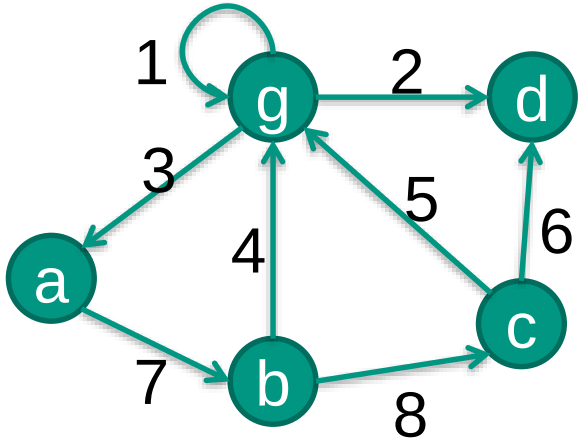
Zyklus

- Begriff des **Zyklus/Zyklen** in Graphen:
  - Für viele Algorithmen, die auf Graphen arbeiten ist es wichtig zu wissen, ob es möglich ist, mit unterschiedlichen Kanten „im Kreis zu laufen“
  - Man bezeichnet einen kompletten **Graphen** als **zyklisch**, wenn
    → wenigstens eine **geschlossene Kantenzugprogression** (= **Zyklus**) in diesem Graphen existiert
    → eine **Schleife** ist ebenfalls ein **Zyklus**
  - Wenn ein **Graph nicht zyklisch** ist
    → nennt man ihn **zyklenfrei** oder **azyklisch**

Übung

Ungerichteter Graph		Gerichteter Graph	
$g^1 \neq g^{n+1}$	$g^1 = g^{n+1}$	$g^1 \neq g^{n+1}$	$g^1 = g^{n+1}$
offene Kanten-progression	geschlossene Kanten-progression	offene Kanten-progression	geschlossene Kanten-progression
Sind alle Kanten einer Progression voneinander verschieden			
Ketten-progression	geschlossene Kantenzug-progression	Weg-progression	Zyklus-progression
Berücksichtigt man die Kanten einer Progression ohne Ordnung			
Kette	geschlossener Kantenzug	Weg	Zyklus

Merkmal
Offene Kantenprogression
Geschl. Kantenprogression
Wegprogresssion
Zyklusprogression
Weg
Zyklus

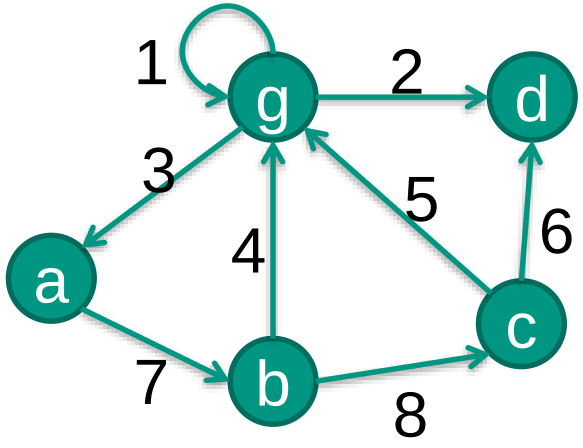


Übung

Ungerichteter Graph		Gerichteter Graph	
$g^1 \neq g^{n+1}$	$g^1 = g^{n+1}$	$g^1 \neq g^{n+1}$	$g^1 = g^{n+1}$
offene Kanten-progression	geschlossene Kanten-progression	offene Kanten-progression	geschlossene Kanten-progression
Sind alle Kanten einer Progression voneinander verschieden			
Ketten-progression	geschlossene Kantenzug-progression	Weg-progression	Zyklus-progression
Berücksichtigt man die Kanten einer Progression ohne Ordnung			
	geschlossener Kantenzug	Weg	Zyklus

- **Beispiele am gerichteten Graphen:**
  Finden Sie folgende Kantenfolgen:

Merkmal	**Folge**
Offene Kantenprogression	4-3-7-4
Geschl. Kantenprogression	3-7-4-1-1
Wegprogresssion	4-1-2
Zyklusprogression	5-1-3-7-8
Weg	{ 2, 8, 5, 1 }
Zyklus	{ 7, 5, 8, 3 }



IT Inhalt

6. Objektorientierung

- Grundidee und Motivation
- Klassen
- Objekte
- Datenkapselung

7. Datenstrukturen

- Array
- Liste
- Stack
- Queue
- Hash-Tabelle
- Graph
- Baum
- Haufen

Spezieller Graph: Baum

- **Definition: Baum**

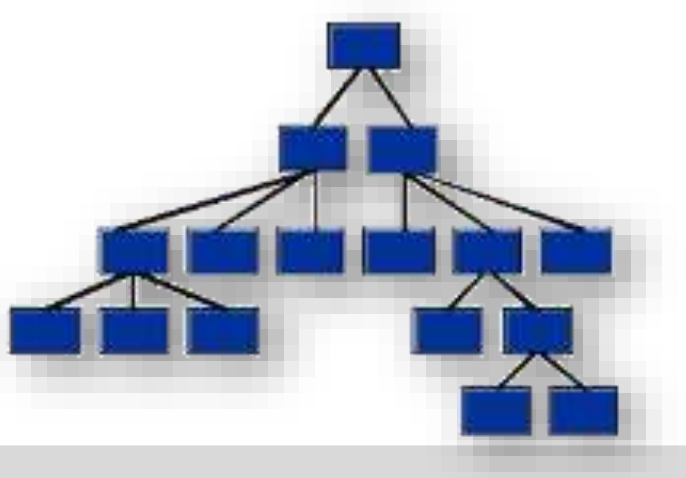
  → ein **zusammenhängender**, **zyklenfreier** Graph

- **Also:**
  - bei einem **ungerichteten Baum**

    → es darf **kein geschlossener Kantenzug** existieren
  - bei einem **gerichteten Baum** gilt:

    → es darf **kein Zyklus** existieren

    → der zugehörige ungerichtete Graph muss **zusammenhängend** sein

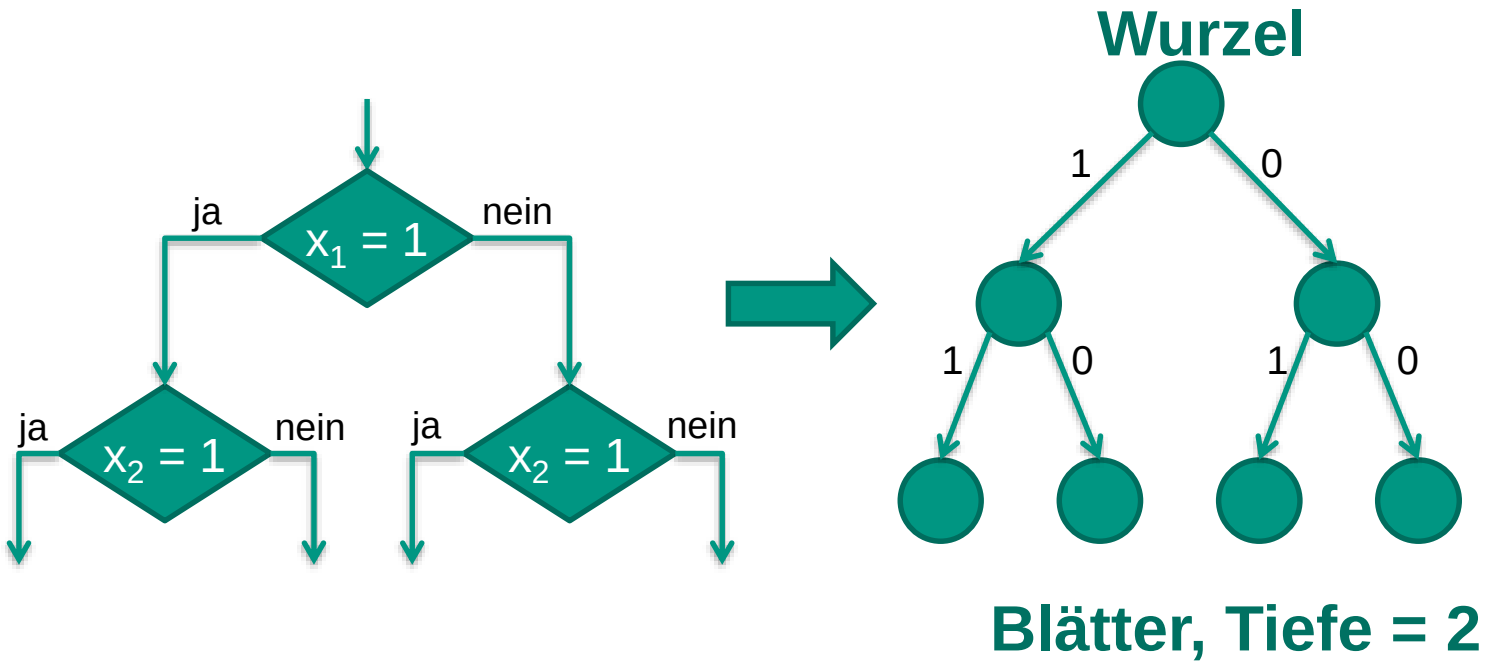


Bäume: Wurzel, Blätter, Tiefe

- Wurzel: In einem gerichteten Baum
  → gibt es genau einen Knoten, der keine Vorgänger hat $d^- = Ø$.
  → dieser Knoten wird Wurzel genannt.
- Blatt: In einem gerichteten Baum
  → heißen die Knoten ohne Nachfolger ($d^+ = Ø$) Blätter.
- Bei einem ungerichteten Baum
  → kann die Wurzel frei gewählt werden
  → daraus ergeben sich die Blätter als übrige Knoten mit Knotengrad 1
- Tiefe:
  → Der Abstand von der Wurzel zu den Blättern wird als Tiefe des Baumes bezeichnet
  - Ist der Abstand von der Wurzel zu den Blättern für alle Blätter identisch
    → so handelt es sich um einen symmetrischen Binärbaum
  - Unterscheiden sich die Abstände um maximal eins
    → so nennt man den Baum ausgeglichen

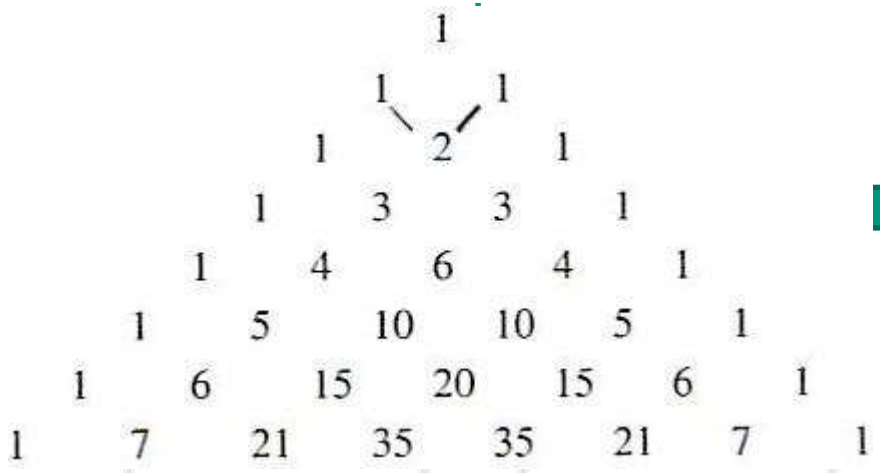
Binärbaum

- Definition: Binärbaum
  - Hat jeder Knoten eines Baumes, außer den Blättern, genau zwei Nachfolger: d+(g) = 2, so heißt ein solcher Baum Binärbaum

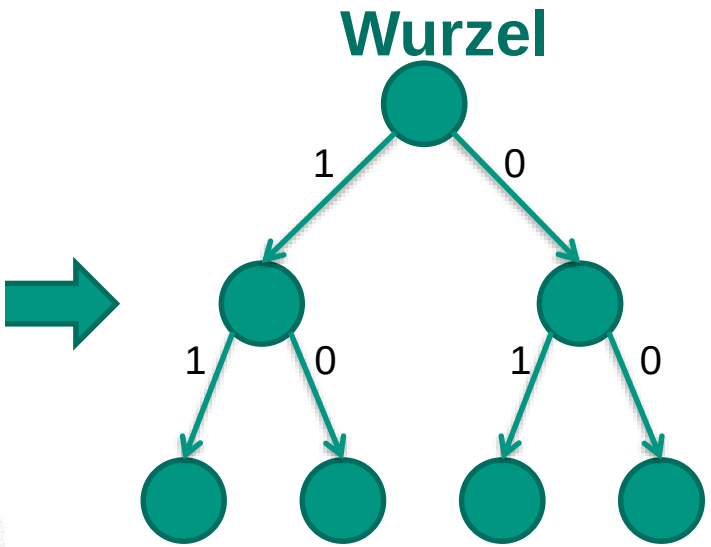


Binärbaum

- Definition: Binärbaum
  - Hat jeder Knoten eines Baumes, außer den Blättern, genau zwei Nachfolger: d+(g) = 2, so heißt ein solcher Baum Binärbaum



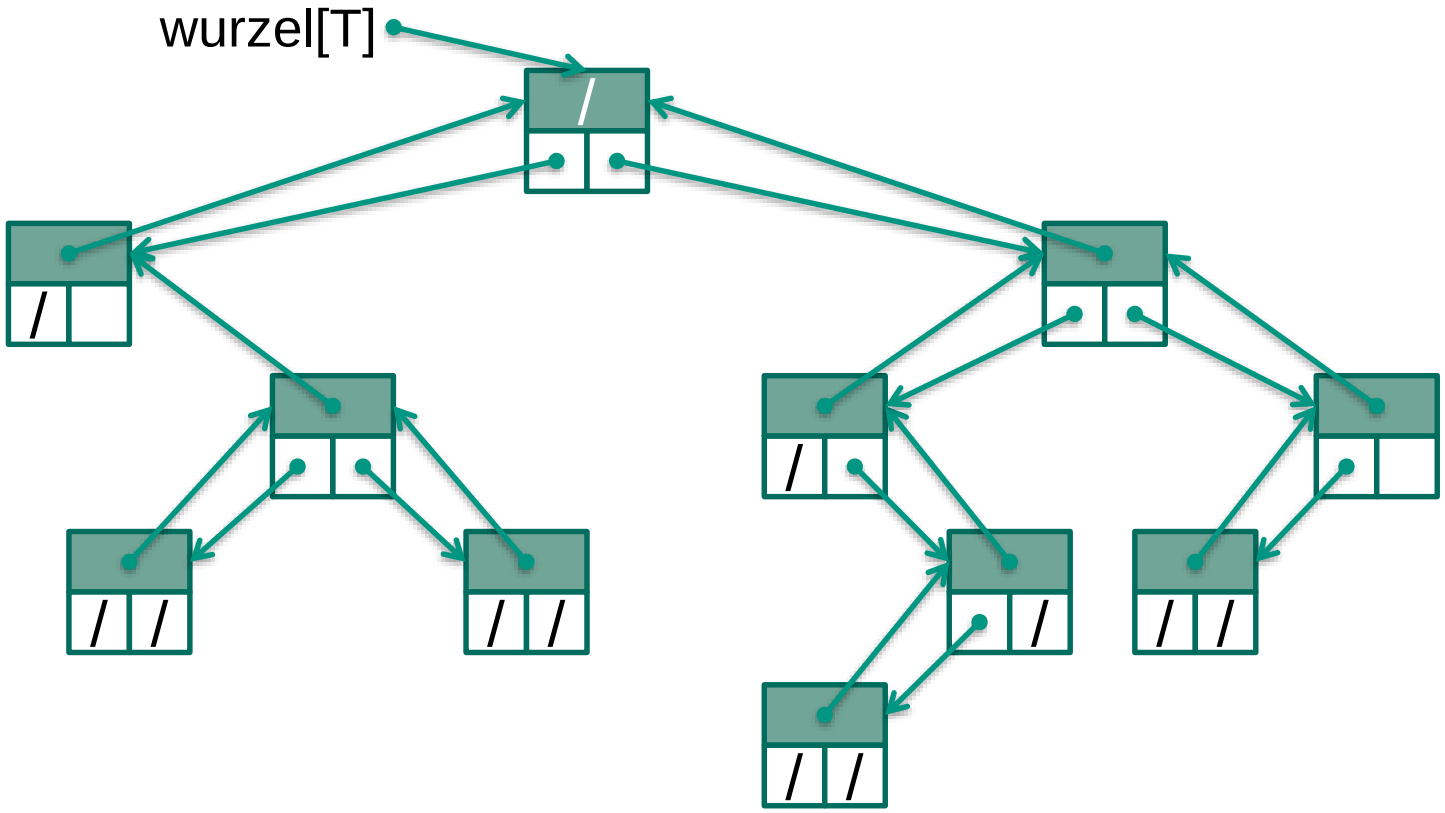
Pascal‘sches Dreieck



Blätter, Tiefe = 2

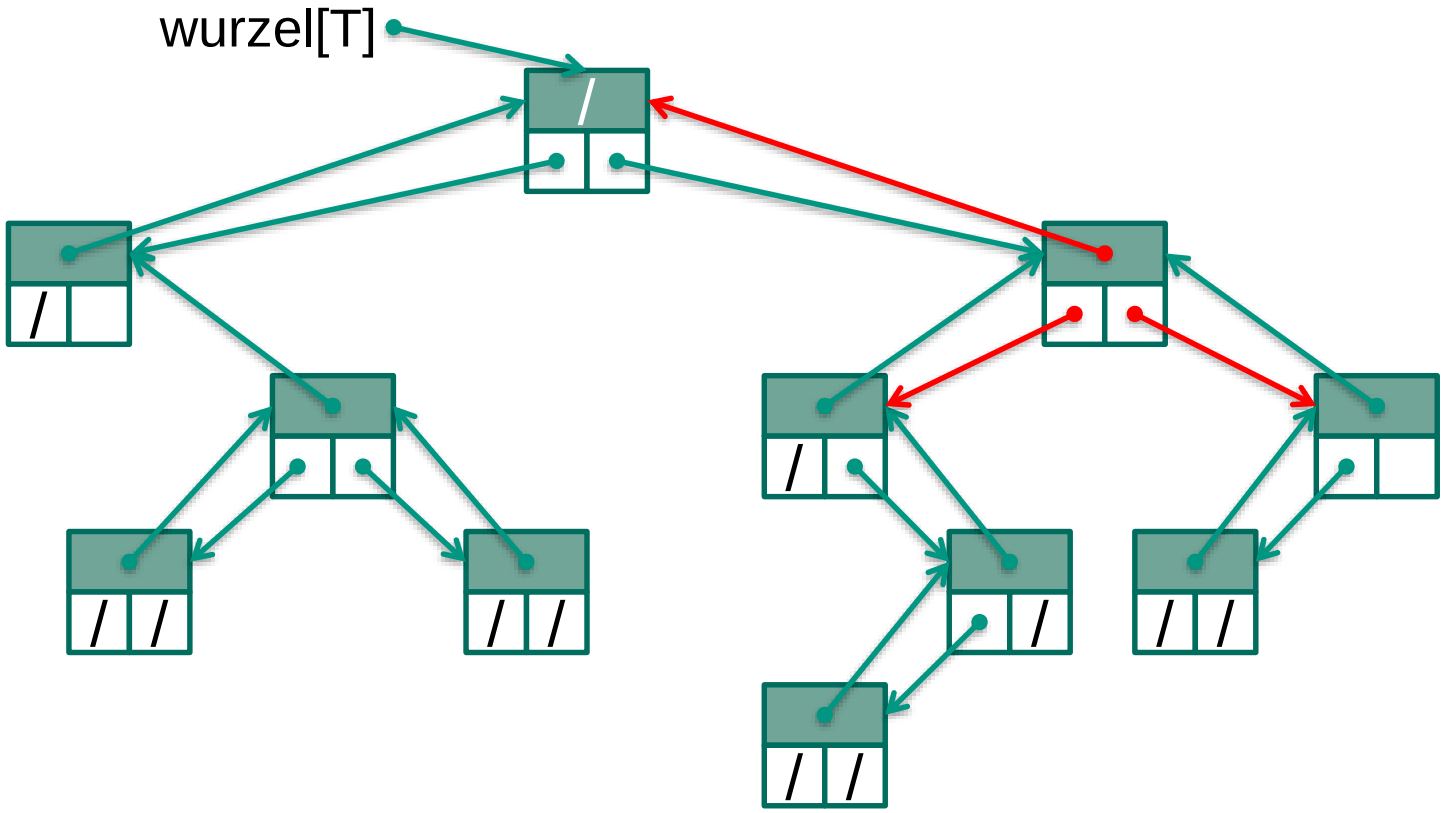
Darstellung eines Binärbaumes über eine Liste

- Dabei besteht jeder Knoten aus dem Wert und drei Zeigern:
  → Vorgänger → linker Nachfolger → rechter Nachfolger



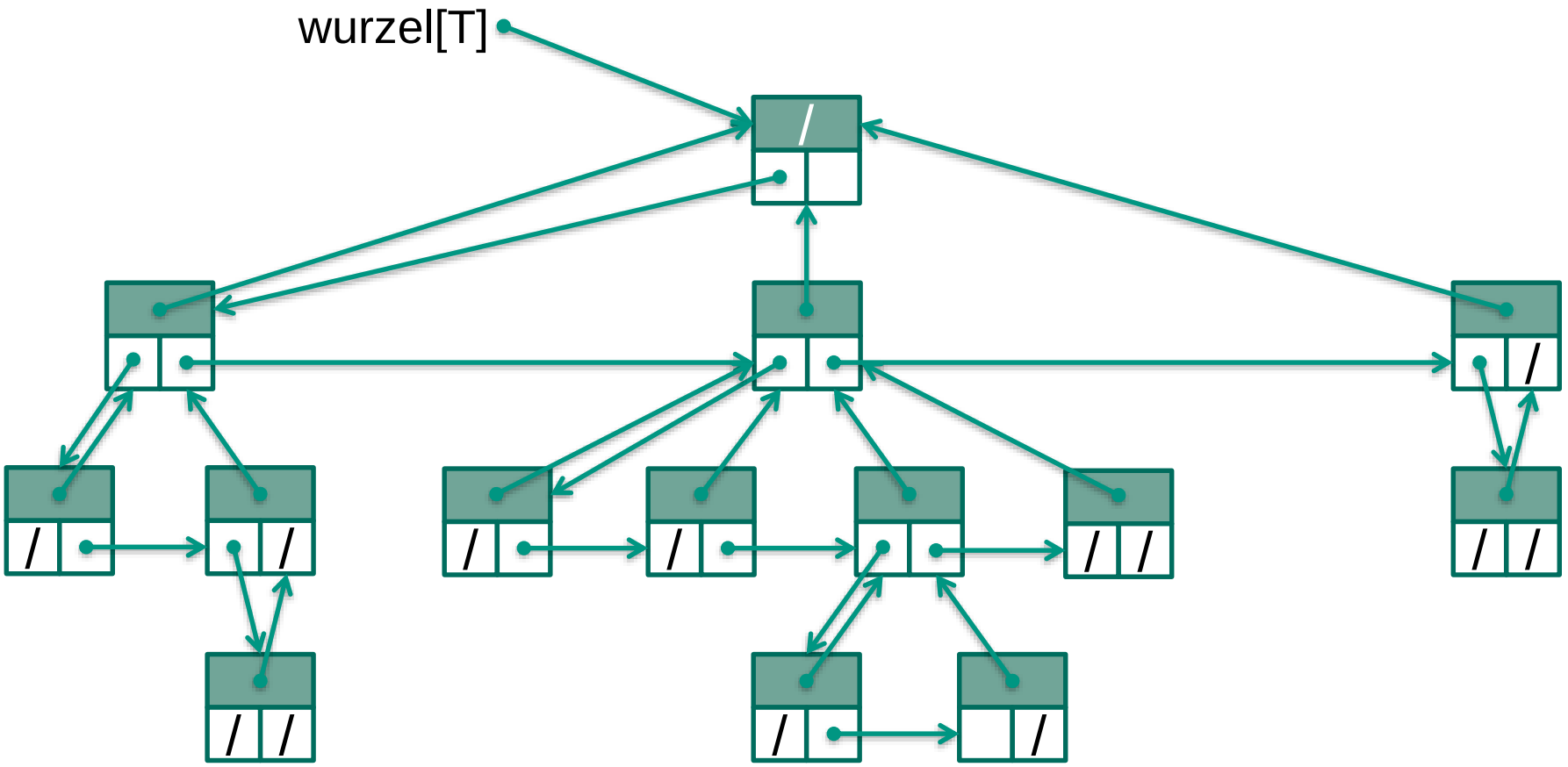
Darstellung eines Binärbaumes über eine Liste

- Dabei besteht jeder Knoten aus dem Wert und drei Zeigern:
  → Vorgänger → linker Nachfolger → rechter Nachfolger



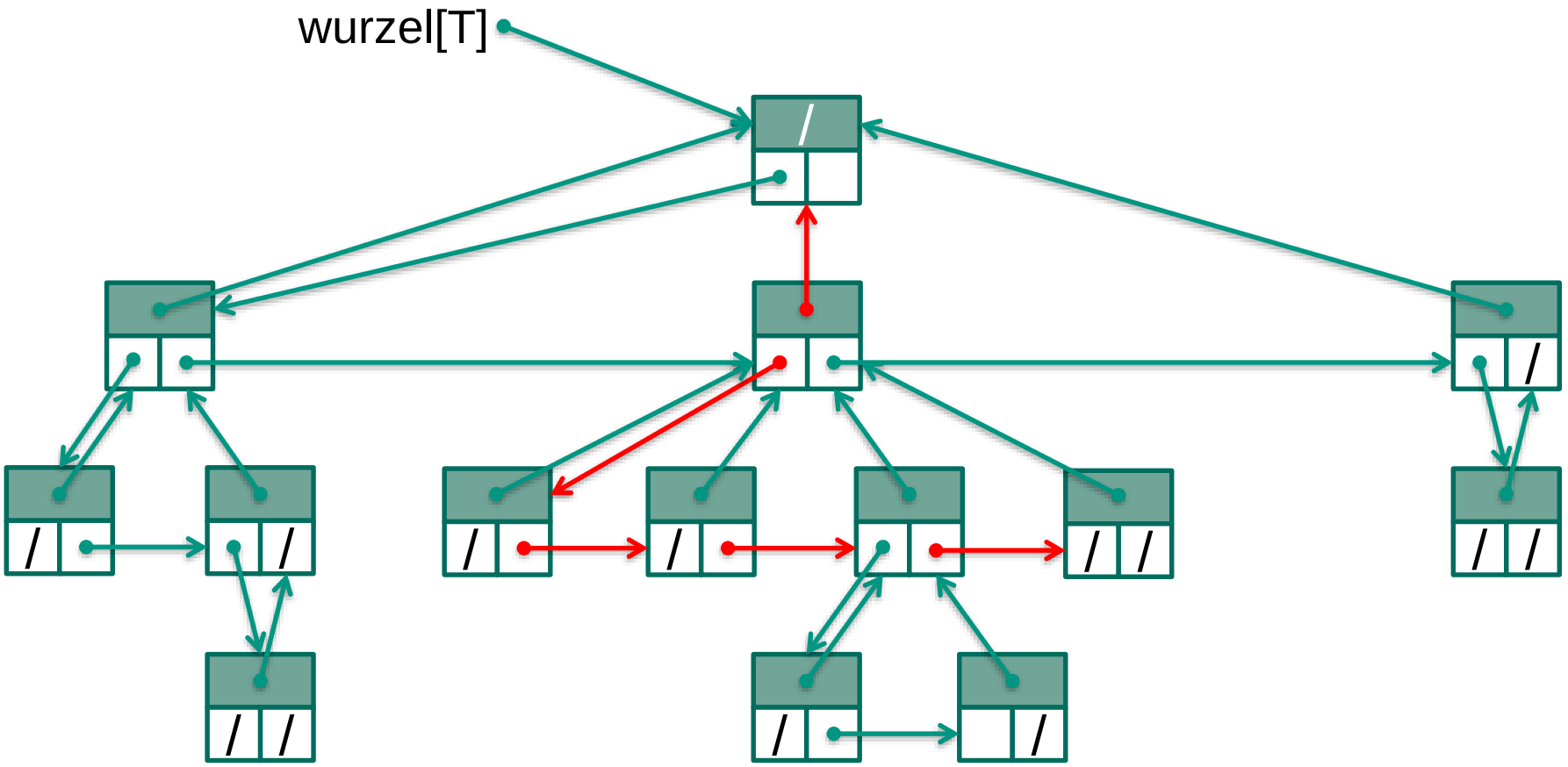
Darstellung eines allgemeinen Baumes über eine Liste

- Nun besteht jeder Knoten aus dem Wert und drei anderen Zeigern:
  → Vorgänger → linker Nachfolger → rechter „Bruder“



Darstellung eines allgemeinen Baumes über eine Liste

- Nun besteht jeder Knoten aus dem Wert und drei anderen Zeigern:
  → Vorgänger → linker Nachfolger → rechter „Bruder“



IT Inhalt

6. Objektorientierung

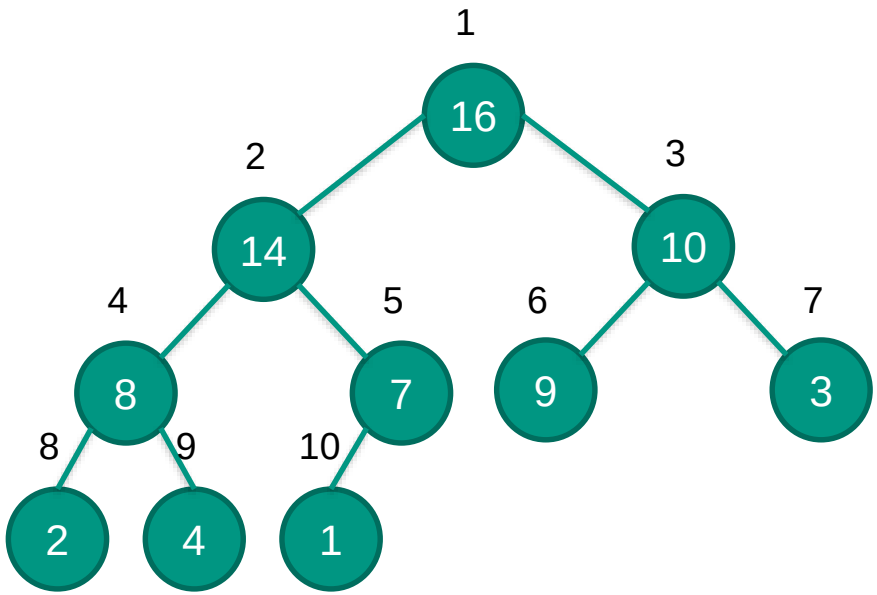
- Grundidee und Motivation
- Klassen
- Objekte
- Datenkapselung

7. Datenstrukturen

- Array
- Liste
- Stack
- Queue
- Hash-Tabelle
- Graph
- Baum
- Haufen

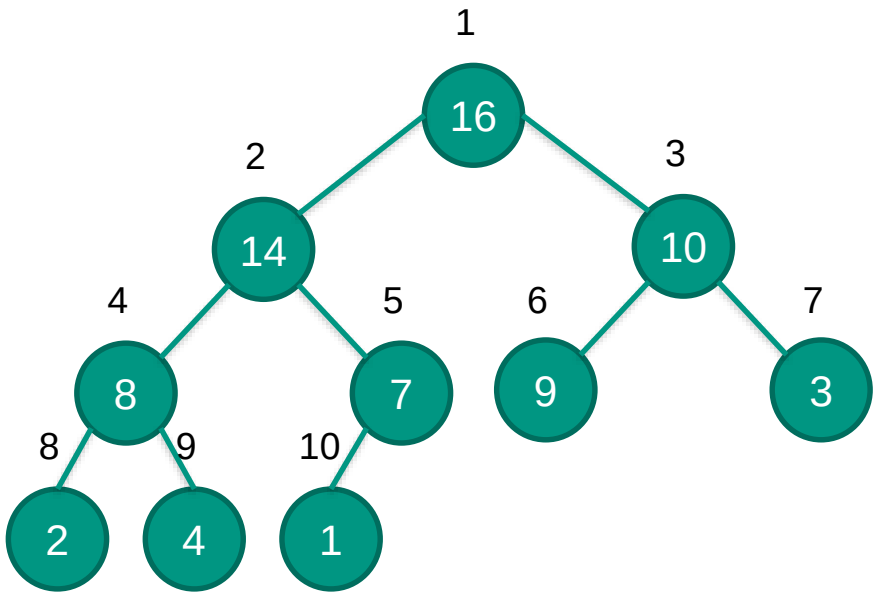
Heap

- Heaps werden meistens über Bäume aufgebaut.
  - Über die Menge der Schlüssel wird eine totale Ordnung festgelegt, über welche die Reihenfolge der eingefügten Elemente bestimmt wird.
  - Beispielsweise kann die Menge der ganzen Zahlen zusammen mit der Kleiner-Relation (<) als Schlüsselmenge fungieren.
- Verwendung finden Heaps vor allem dort, wo schnell ein Element mit höchster Priorität aus dem Heap zu entnehmen ist.



Heap

- Je nach Reihenfolge wird ein Min-Heap oder ein Max-Heap verwendet.
- Eigenschaft Max-Heap:
  - Für jeden Knoten (außer Wurzel) gilt:
  - der Wert eines Knotens i ist kleiner dem Wert des Vaterknotens V(i)
- Eigenschaft Min-Heap:
  - Vice versa



- Bäume
- Heap

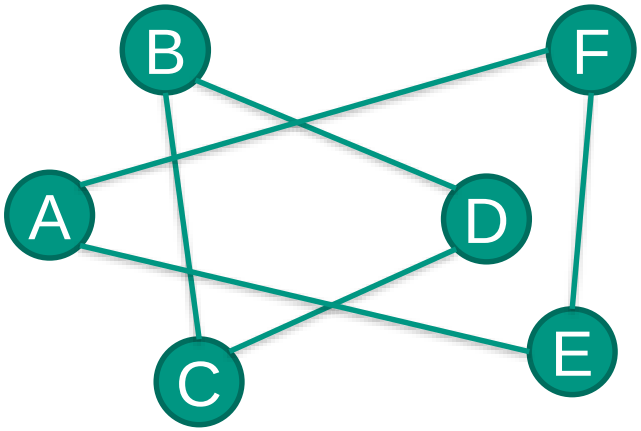
Backup

Graphentheorie: Begriffe

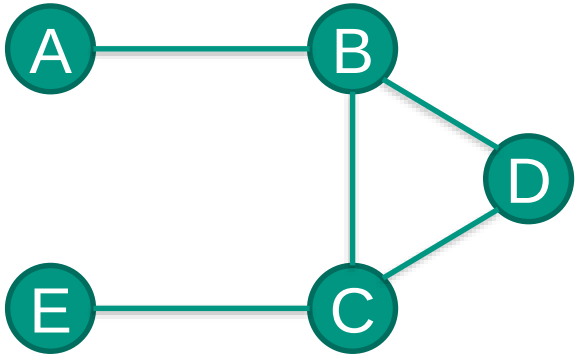
- **Globale Charakterisierung von Graphen:**
  - Graphen, deren Mengen V und E endlich sind:
    → endliche Graphen (insbesondere für technische Anwendungen)
  - Ist die Menge der Kanten E leer
    → so handelt es sich um einen entarteten Graphen
    → dieser besteht nur aus isolierten Knoten
  - Wenn zu je zwei verschiedenen Knoten höchstens eine Kante existiert (keine Mehrfachkanten)
    → so handelt es sich um einen einfachen Graphen

Zusammenhängende Graphen

- durch Folgen von Kanten und Knoten kann man von jedem beliebigen Knoten des Graphen zu jedem anderen Knoten des Graphen gelangen
  → der Graph ist zusammenhängend
- Ist der Graph nicht zusammenhängend
  → so besteht er aus mindestens zwei Teilgraphen
- Beispiel:



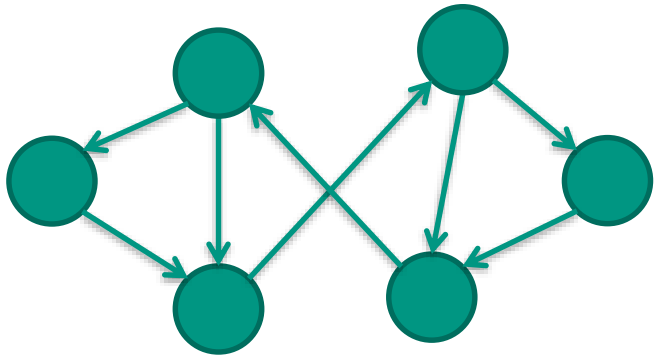
nicht zusammenhängend



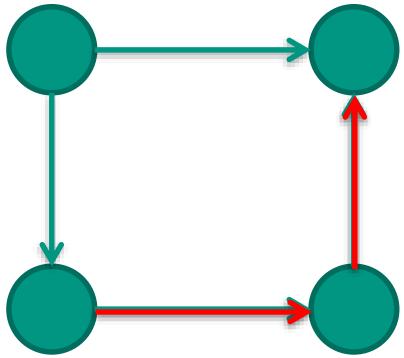
zusammenhängend

Streng zusammenhängende Graphen

- **Gerichtete Graphen** bezeichnet man als **zusammenhängend**, wenn der zugehörige **ungerichtete Graph zusammenhängend** ist bzw. je zwei Knoten sich durch eine Kantenfolge verbinden lassen.
- Findet man zusätzlich von **jedem** beliebigen Knoten des Graphen zu jedem anderen Knoten eine **gerichtete Folge von Kanten** (Weg unter Einbezug der Richtungen!) → so ist der **Graph streng zusammenhängend**.



streng zusammenhängend



nicht streng zusammenhängend

Lokale Eigenschaften von Graphen:

- **Schlinge oder Schleife**
  - Verbindet eine Kante einen Knoten mit sich selbst, so wird diese als **Schleife** oder **Schlinge** bezeichnet
  - $\Phi(e) = (g,g)$
- **Mehrfachkanten**
  - Existieren **mehrere Kanten** zwischen zwei Knoten g und h, so heißen diese **parallel** bzw. **Mehrfachkanten**
    - $\Phi(e) = \Phi(f) = \{g,h\}$ bzw. (g,h) $\quad e \neq f$
  - Bei **gerichteten Graphen** nennt man Kanten **antiparallel**, falls sie zwei Knoten in entgegengesetzter Richtung verbinden
    - $\Phi(e) = (g,h)$, $\Phi(f) = (h,g)$

Aufspannender Baum

- Definition: Aufspannender Baum
  - Es gilt: bei jedem zusammenhängenden Graphen kann man alle Zyklen durch gezieltes Entfernen von Kanten auflösen
    (→ Kruskal Algorithmus: Minimal Aufspannender Baum, MAB)
    ohne dass der Graph in zwei Teilgraphen zerfällt
  - Auf diese Weise erhält man zu jedem beliebigen Graphen einen zugehörigen aufspannenden Baum