DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart
Campus Horb

Algorithmen und Datenstrukturen

Teil 4:
Graphen (II): Bäume

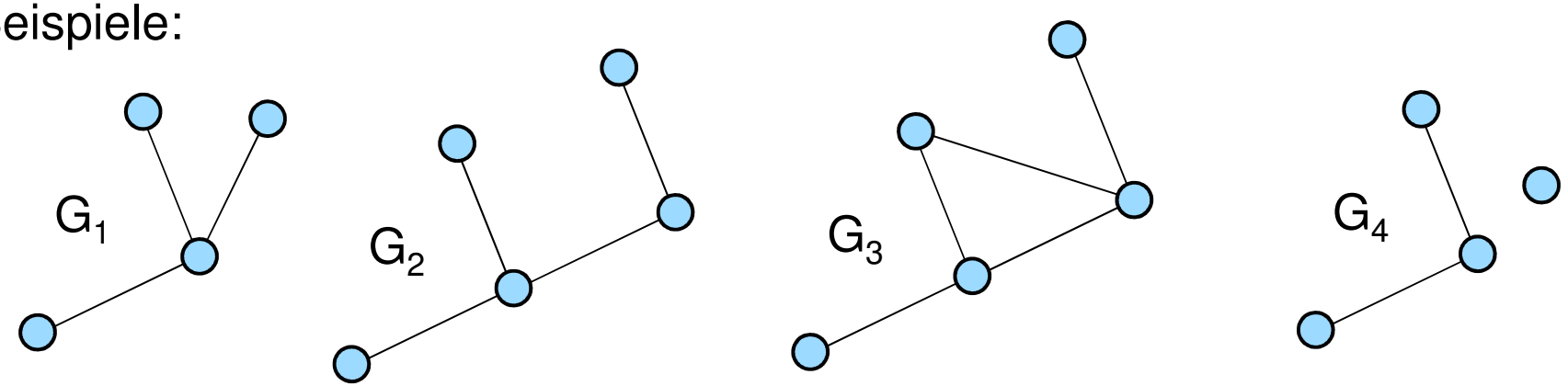
DHBW Stuttgart Campus Horb
Fakultät Technik
Studiengang Informatik
Dozent: Olaf Herden
Stand: 01/2019

Gliederung

- Definitionen und Eigenschaften
- Datenstrukturen für Bäume
- Anwendungen
- Algorithmen

Bäume

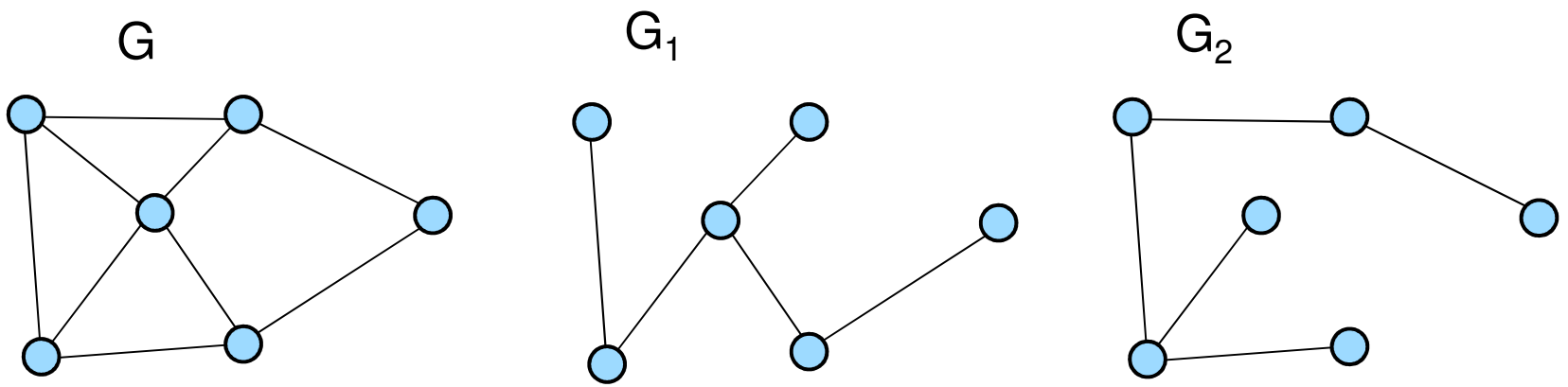
- Ungerichteter Graph ohne geschlossenen Weg heißt Wald
- Zusammenhangkomponenten eines Waldes heißen (ungerichtete) Bäume
- Anders ausgedrückt: (Ungerichteter) Baum ist zusammenhängender ungerichteter Graph, der keinen Zyklus enthält
- Beispiele:



  - G_1 und G_2 sind Bäume
  - G_3 ist kein Baum (enthält Zyklus)
  - G_4 ist Wald mit zwei Bäumen

Aufspannende Bäume

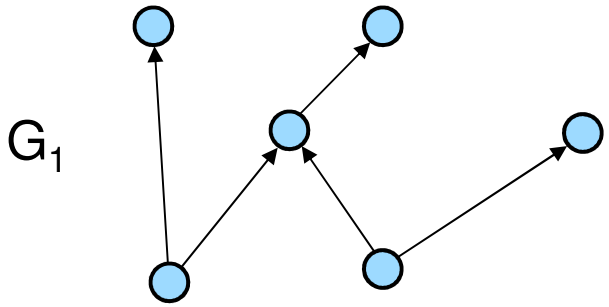
- Wenn G ein Graph ist, dann heißt G‘ aufspannender Baum von G, wenn
  - G‘ hat die gleiche Knotenmenge wie G
  - G‘ enthält Teilmenge der Kantenmenge von G
  - G‘ ist ein Baum
- Beispiel:

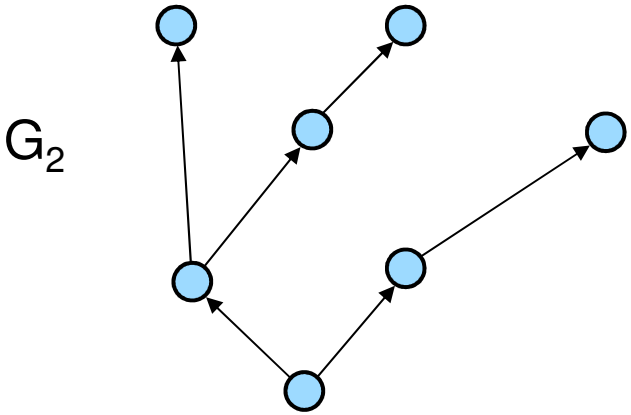


  - G_1 und G_2 sind aufspannende Bäume von G

Gerichtete Bäume und Wurzelbäume (I)

- Gerichteter Graph heißt gerichteter Baum, wenn der zugrundeliegende ungerichtete Graph Baum ist
- Sind v_1 und v_2 Knoten eines gerichteten Baumes und v_1 ist von v_2 aus erreichbar, dann heißt v_2 Vorgänger von v_1 und v_1 Nachfolger von v_2.
- Knoten v heißt Wurzel eines gerichteten Baumes, wenn von v aus jeder andere Knoten erreichbar
- Gerichteter Graph G heißt Wurzelbaum oder Baum, wenn G gerichteter Baum ist und genau eine Wurzel besitzt
- Beispiele:

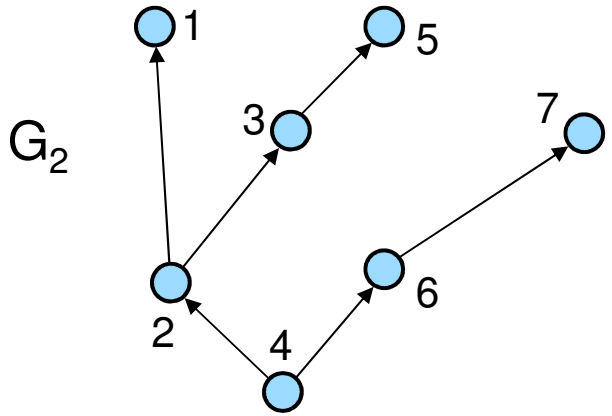




  - G_1 ist ein Baum, aber kein Wurzelbaum
  - G_2 ist ein Wurzelbaum

Gerichtete Bäume und Wurzelbäume (II)

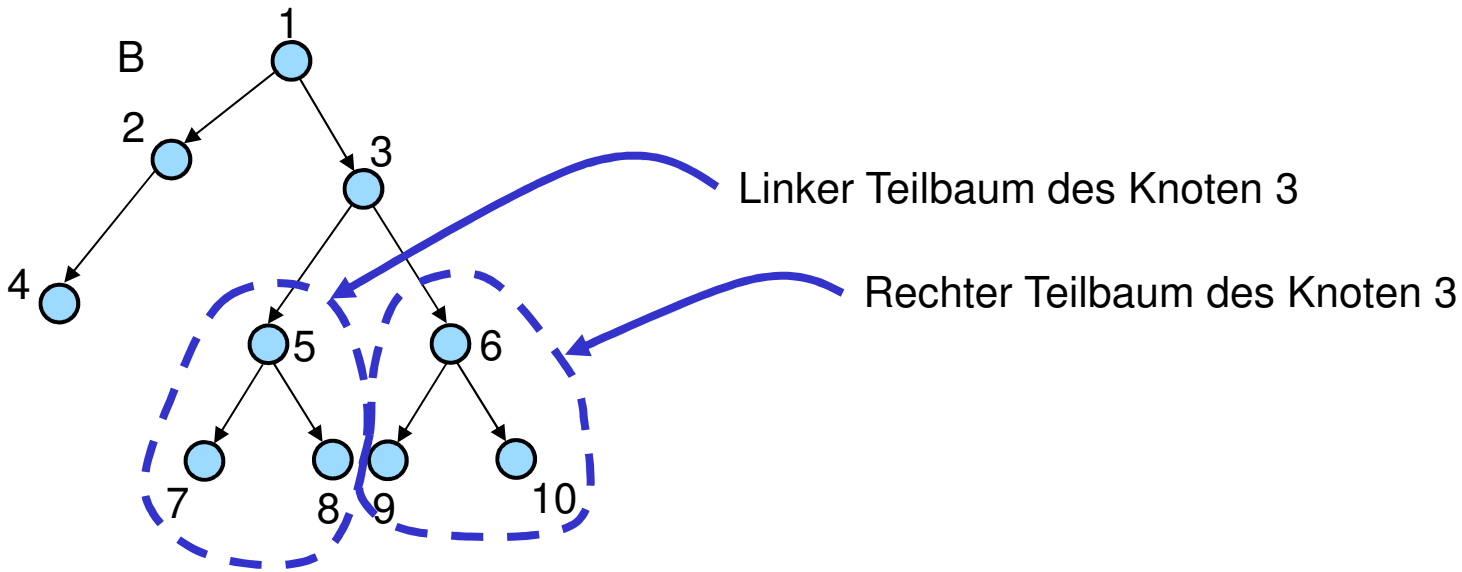
- Knoten eines gerichteten Baumes mit Ausgangsgrad 0 heißt Blatt.
- Andere Knoten des Baumes heißen innere Knoten.
- Niveau (oder Ebene) eines Knotens ist die Entfernung von der Wurzel in Anzahl Kanten + 1.
- Höhe eines Wurzelbaums ist das maximale Niveau eines Knotens.
- Beispiel:



  - Knoten 4 ist Wurzel, Knoten 1,5 und 7 sind Blätter, restliche Knoten innere Knoten
  - Niveau von Knoten 2 und 6 ist 2, Knoten 1, 3 und 7 haben Niveau 3 und der Knoten 5 Niveau 4
  - G_2 besitzt die Höhe 4
- Anmerkung: Häufig wird nicht zwischen Wurzelbaum und Baum unterschieden

Binärbäume

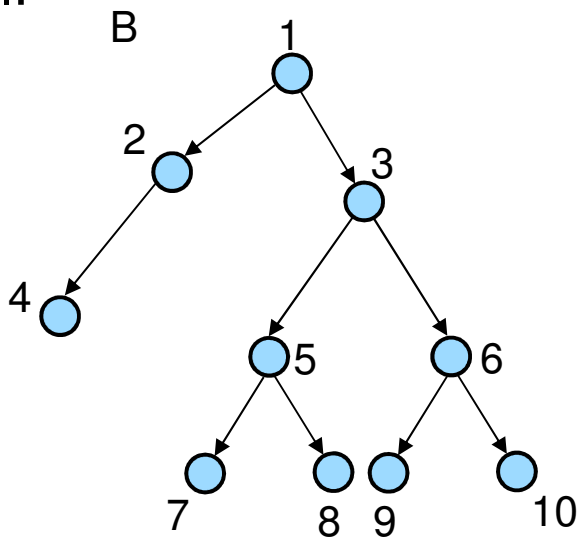
- Wurzelbaum heißt Binärbaum, wenn maximaler Ausgangsgrad eines Knotens 2 ist.
- Nachfolger eines Knoten v bilden Wurzeln des linken und rechten Teilbaums von v.
- Beispiel:



  - B ist Binärbaum

Namenskonvention

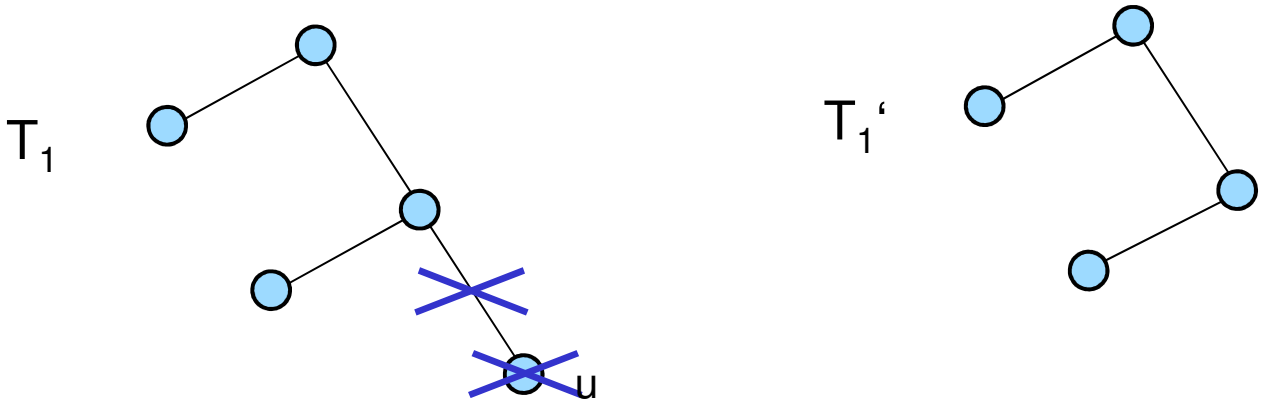
- Statt Nachfolger und Vorgänger wird bei Bäumen häufig von Eltern- und Kindknoten gesprochen
- Benachbarte Knoten heißen dann auch Geschwister
- Beispiel:



  - 3 ist Elternknoten (oder Vorgänger) von 5 und 6
  - 5 und 6 sind Kindknoten (oder Nachfolger) von 3
  - 5 und 6 sind Geschwisterknoten

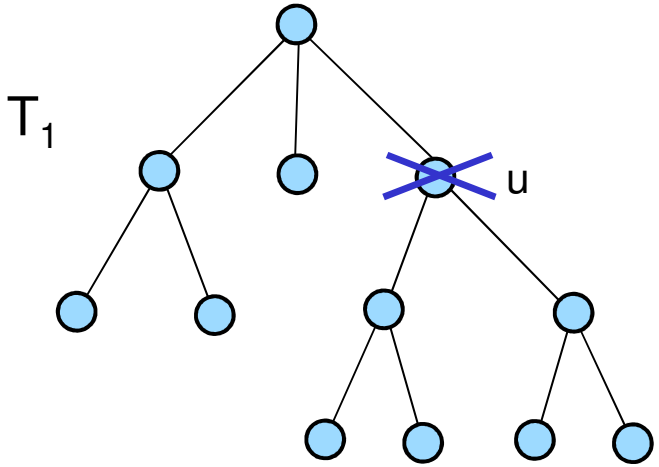
Eigenschaften (I)

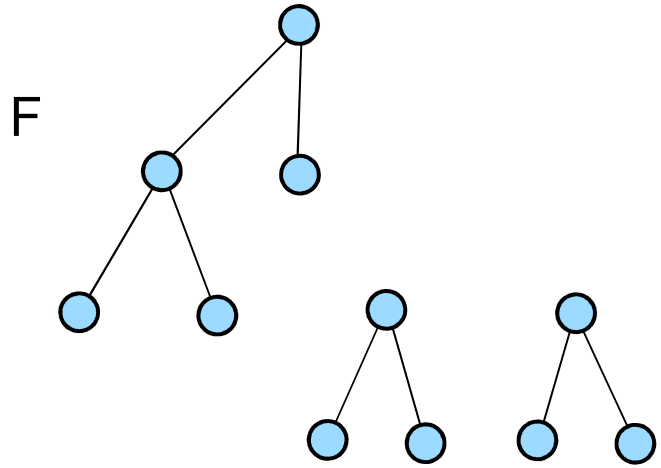
- Sei T=(V,E) Baum. Dann gilt:
  - T hat genau eine Kante weniger als Knoten (Kantenzahl = Knotenzahl – 1, |E| = |V| -1)
  - Wenn |V| ≥ 2 und u ∈ V Blatt, so bleibt T nach Entfernen von u und seinen ausgehenden Kanten ein Baum
  - Beispiel:



Eigenschaften (II)

- Wenn u ∈ V beliebiger Knoten, so entstehen nach Entfernen von u und seinen ausgehenden Kanten deg(u) Bäume
- Beispiel: Durch Entfernen von u aus T1 entsteht Wald F mit drei Bäumen, weil deg(u) = 3





Anzahl Binärbäume mit n Knoten

- Unterscheidung:
  - Knotennamen relevant (markierte Bäume)
  - Knotennamen irrelevant, nur Form zählt (Strukturell verschiedene Bäume)
- Es gilt:

Knoten	2	3	4	5	6	7	8
Markierte Bäume	1	3	16	125	1296	16807	262144
Knoten-namen irrelevant	2	5	14	42	132	429	1430

- Für markierte Bäume gilt (Satz von Cayley):
  Für n ≥ 2 Knoten gibt es genau n^{n-2} markierte Bäume.
- Für strukturell verschiedene Bäume gilt (Catalan-Zahlen):

$$b_n = \frac{1}{n+1}\binom{2n}{n} = \begin{cases} 1, & \text{falls } n = 0, \\ \sum_{k=0}^{n-1} b_k b_{n-1-k}, & \text{falls } n > 0. \end{cases}$$

Übersicht

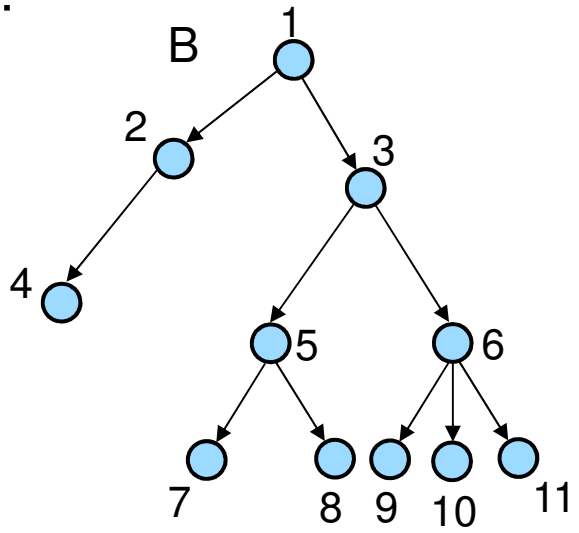
- Definitionen und Eigenschaften
- Datenstrukturen für Bäume
- Anwendungen
- Algorithmen

Motivation

- Bäume sind spezielle Form von Graphen
- Für Graphen bekannte Datenstrukturen auch für Bäume verwendbar
- Aber: Spezialfall lässt Optimierungen zu
- Im Folgenden:
  - Effiziente Strukturen für Wurzelbäume
  - Spezielle Strukturen für Binärbäume
  - Beliebige Bäume als Binärbäume

Wurzelbaum als Feld (I)

- Wurzelbaum: Jeder Knoten (außer Wurzel) hat genau einen Vorgänger
- D.h. durch Angabe der Vorgänger ist Baum eindeutig bestimmt
- D.h. Feld der Vorgänger ist einfache, kompakte Datenstruktur
- Beispiel:



Index	1	2	3	4	5	6	7	8	9	10	11
Vorgänger	0	1	1	2	3	3	5	5	6	6	6

Wurzelbaum als Feld (II)

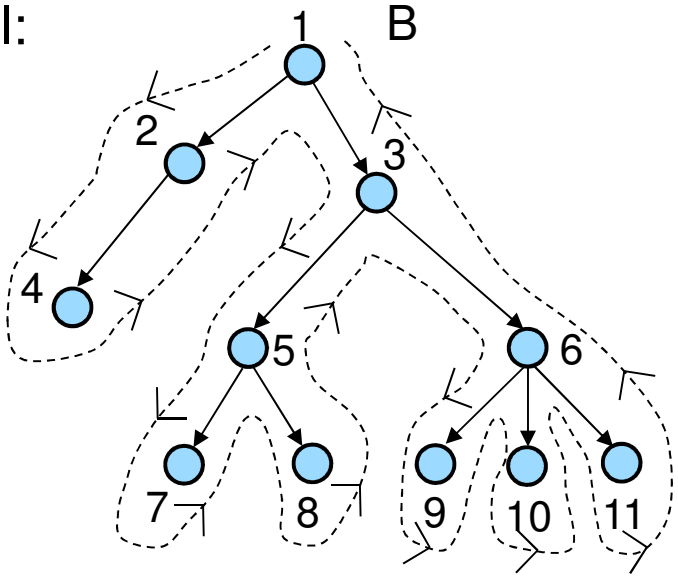
- Vorteil:
  - Abfrage nach Vorgänger (bzw. gesamter Weg zurück zur Wurzel) effizient
  - Beispiel:

Index	1	2	3	4	5	6	7	8	9	10	11
Vorgänger	0	1	1	2	3	3	5	5	6	6	6

- Nachteil:
  - Weg von Wurzel zu einem bestimmten Knoten ineffizient (ganzes Feld muss durchsucht werden)
  - Keine Ordnungsbäume darstellbar (später in VL mehr)

Wurzelbaum als Binärcode (I)

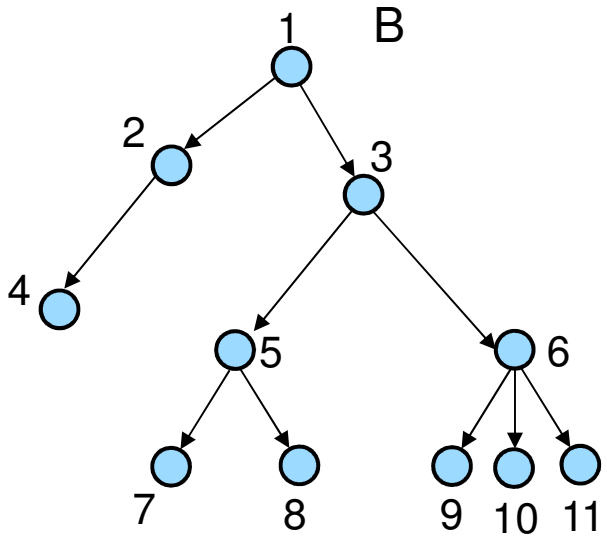
- Bijektion zwischen Wurzelbaum mit n Knoten und Binärcode der Länge 2•n - 2
- Von B zum Binärcode:
  - Starte an Wurzel
  - Umwandere Baum nach links unten (Gestrichelte Linie in Pfeilrichtung)
  - Bei Schritt zu tiefer gelegenem Knoten: Notiere 1
  - Bei Schritt zu höher gelegenem Knoten: Notiere 0
- Beispiel:



Binärcode: 11001110100110101000

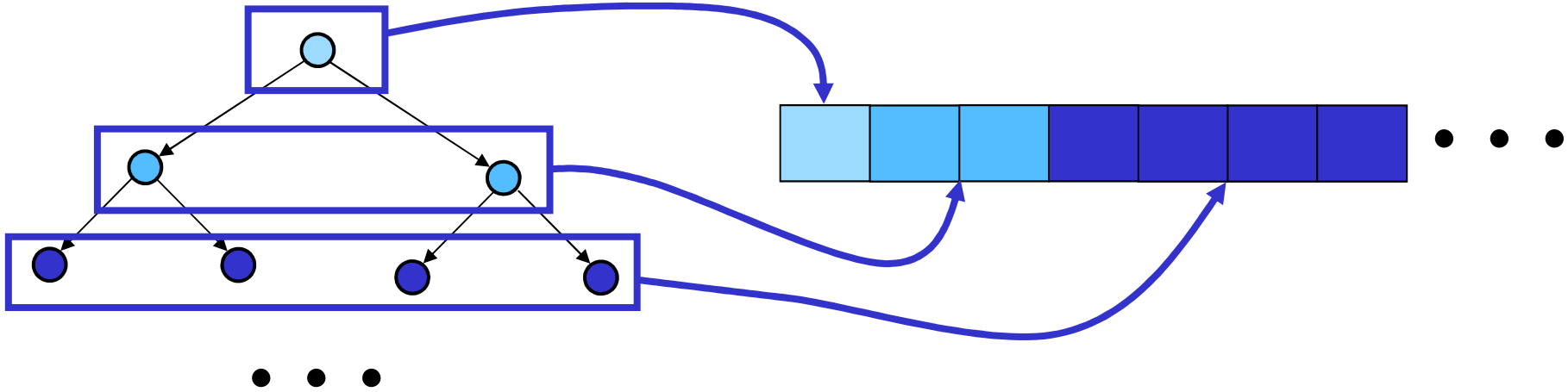
Wurzelbaum als Binärcode (II)

- Vom Binärcode zu B:
  - Starte an Wurzel
  - Gehe Binärcode von links nach rechts durch
  - Bei 1: Zeichne Knoten als Nachfolger von aktuellem Knoten
  - Bei 0: Gehe Schritt zu höher gelegenem Knoten
- Beispiel: Binärcode: 11001110100110101000



Binärbaum als Feld (I)

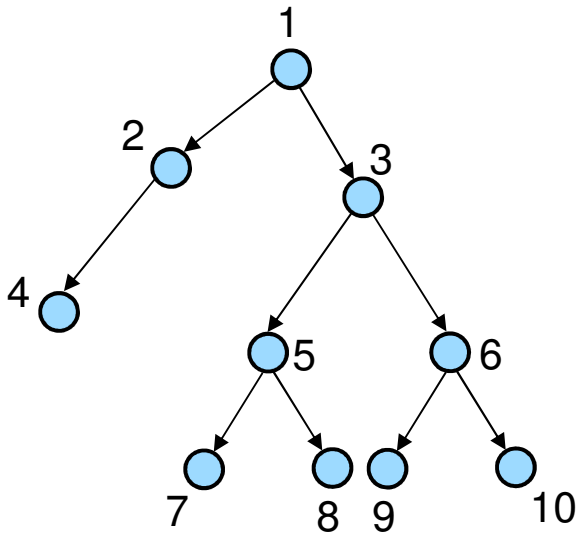
- Binärbäume lassen sich gut als Feld darstellen:
  - Baum der Höhe h hat max. 2^h-1 Knoten
  - i-te Element bei reihenweiser Traversierung wird an Position i gespeichert



- Vorteil: Leichte Navigation in beide Richtungen:
  - Vorgänger(Feld[i]) = Feld[i div 2]
  - Linker Nachfolger(Feld[i]) = Feld[2i]
  - Rechter Nachfolger(Feld[i]) = Feld[2i+1]

Binärbaum als Feld (II)

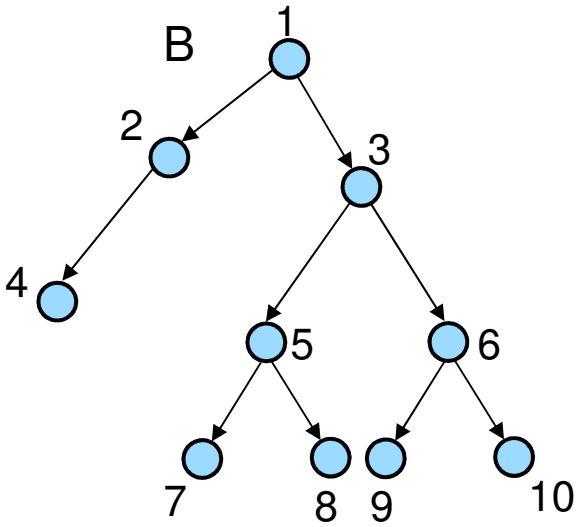
- Unvollständiger Binärbaum:
  - Nehme auch Feld der Länge 2^h-1
  - Trage für nicht vorhandene Knoten Sondersymbol (z.B. #) ein:
    - Feld[1] = Wurzel
    - Feld[2i] = linker Nachfolger von Feld[i], falls vorhanden, sonst #
    - Feld[2i+1] = rechter Nachfolger von Feld[i], falls vorhanden, sonst #
  - Beispiel:



Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Eintrag	1	2	3	4	#	5	6	#	#	#	#	7	8	9	10

Adjazenzliste als Feld

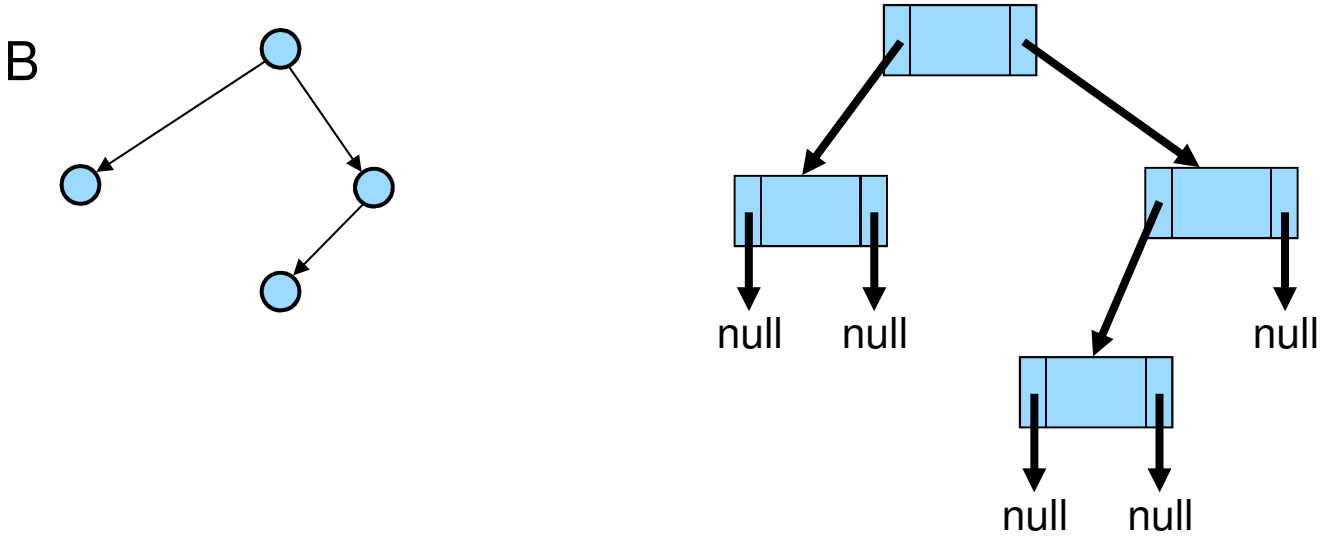
- Da bei Binärbäumen die Listen maximal die Länge 2 haben können, können diese gut als Feld realisiert werden
- Für Binärbäume sehr effiziente Struktur (Navigation in beide Richtungen schnell, ebenso Abfrage nach Wurzel bzw. Blatt)
- Bei jedem Knoten werden die eigentlichen Knoteninformationen (Name und Markierung), Vorgänger und die beiden Nachfolger gespeichert
- Beispiel:



Index	1	2	3	4	5	6	7	8	9	10
Vorgänger	0	1	1	2	3	3	5	5	6	6
Linker Nachfolger	2	4	5	0	7	9	0	0	0	0
Rechter Nachfolger	3	0	6	0	8	10	0	0	0	0

Binärbaum mit Verzeigerungen

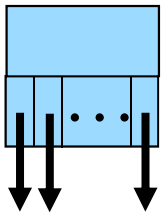
- Für Binärbäume:
  - Jeder Knoten besteht aus Schlüssel und Daten sowie zwei Zeigern, die auf den linken bzw. rechten Nachfolger verweisen
  - Beispiel:



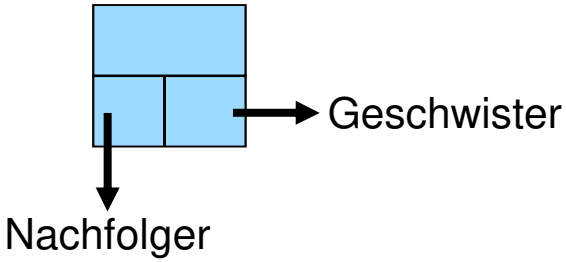
  - Variante/Erweiterung: Auch Rückverweise, wenn Navigation von Blättern zur Wurzel notwendig

Beliebige Bäume als Binärbäume

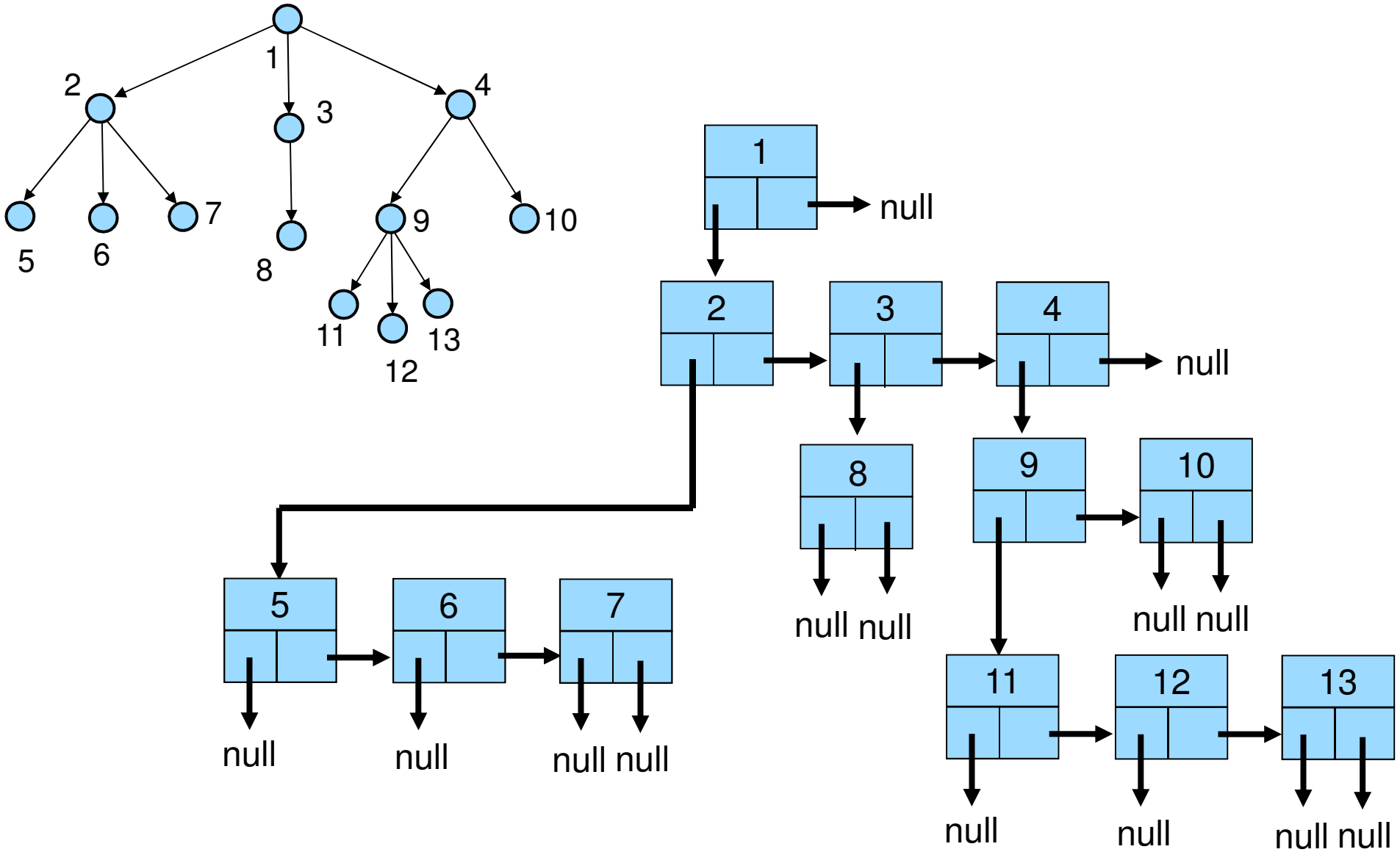
- Erste Idee:
  - Obergrenze an Nachfolgerknoten festlegen (k)
  - Jeder Knoten damit folgende Struktur:



  - Lösung unschön:
    - Feste Obergrenze
    - Speicherplatzverschwendung
- Daher:
  - Stelle Baum als Binärbaum dar
  - Knotenstruktur:



Beispiel

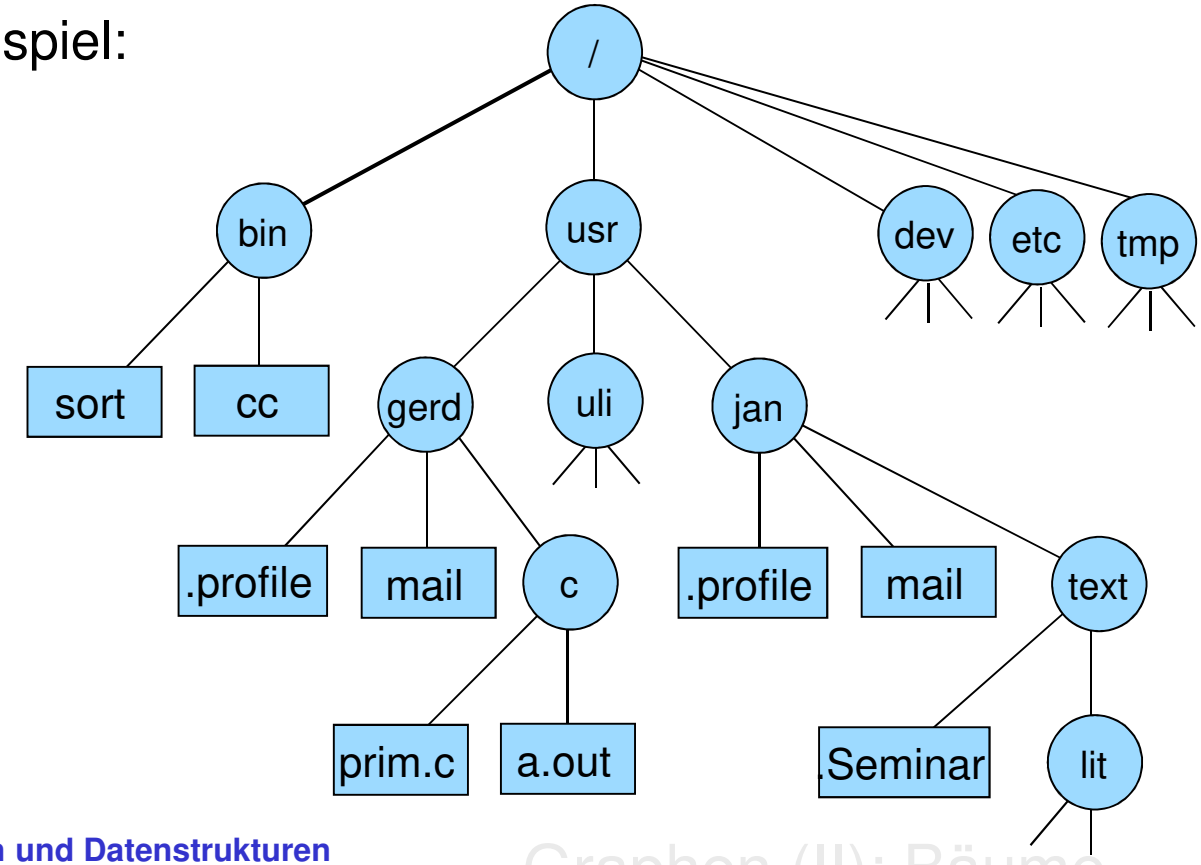

1
2
3
4
5
6
7
8
9
10
11
12
13
null
null null
null
null
null null
null null
null
null
null null

Übersicht

- Definitionen und Eigenschaften
- Datenstrukturen für Bäume
- Anwendungen
- Algorithmen

Dateisysteme

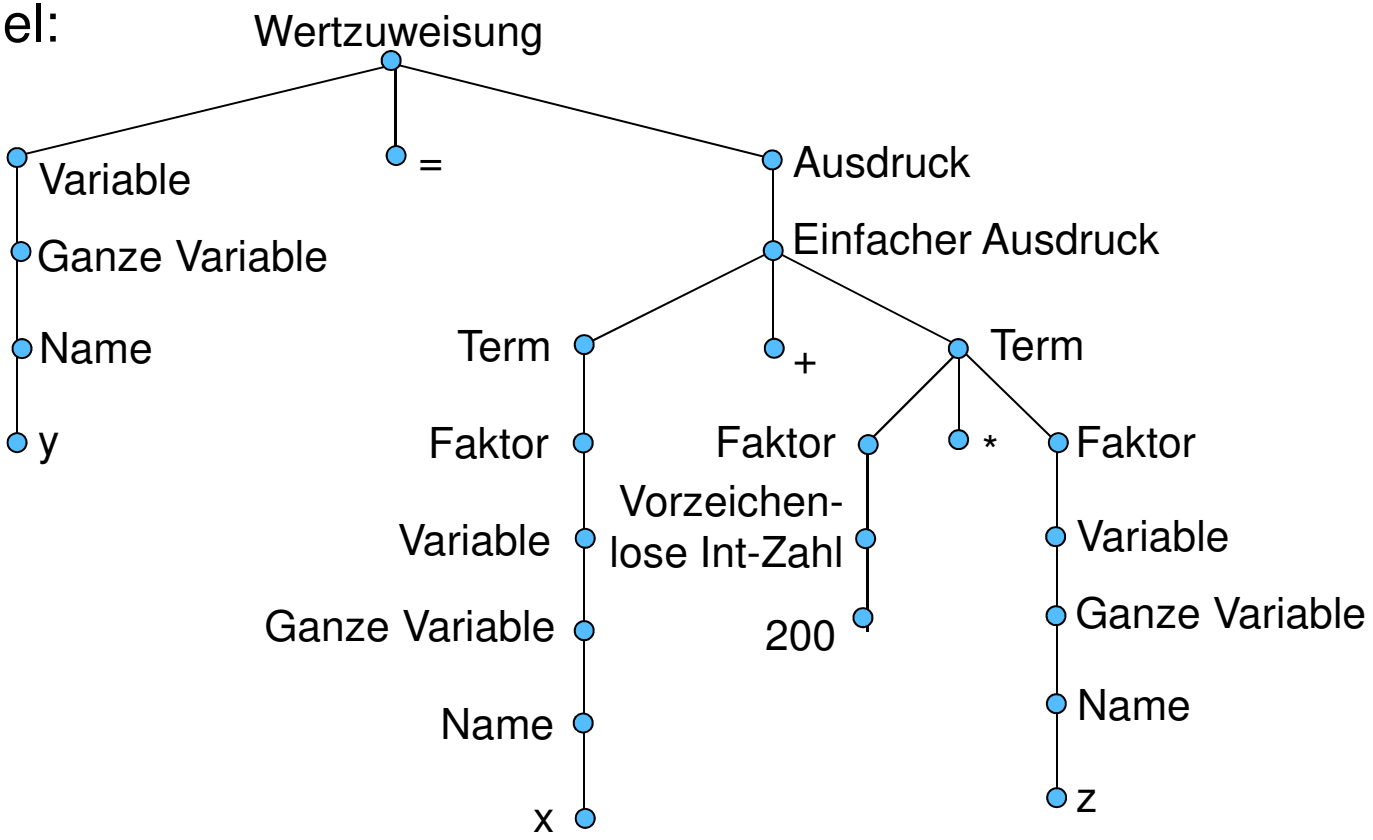
- Dateisysteme in Betriebssystemen sind hierarchisch organisiert:
  - Jeder Knoten entspricht einem Verzeichnis/einer Datei
  - Die in einem Verzeichnis enthaltenen Verzeichnisse/Dateien sind seine Nachfolger
  - Leere Verzeichnisse und Dateien sind Blätter des Baumes
- Beispiel:



[Tura04]

Compilerbau: Ableitungsbäume

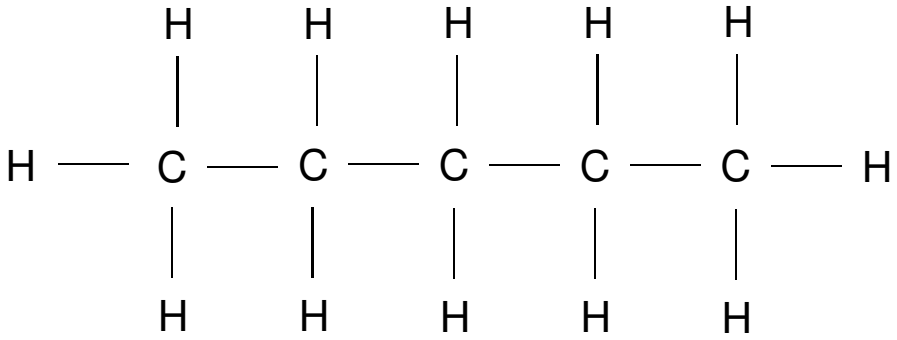
- Compiler-Teilschritt des Parsens: Ist Ausdruck (z.B. `y=x+200*z`) syntaktisch korrekt?
- Knoten sind Symbole, Kanten ihr Zusammenhang
- Innere Knoten sind Nichtterminalsymbole, Blätter sind Terminalsymbole
- Beispiel:



[Tura04]

Chemie: Darstellung Alkane

- Alkane (Kohlenstoff-Wasserstoff-Verbindungen) lassen sich als Baum darstellen
- Beispiel Pentan:

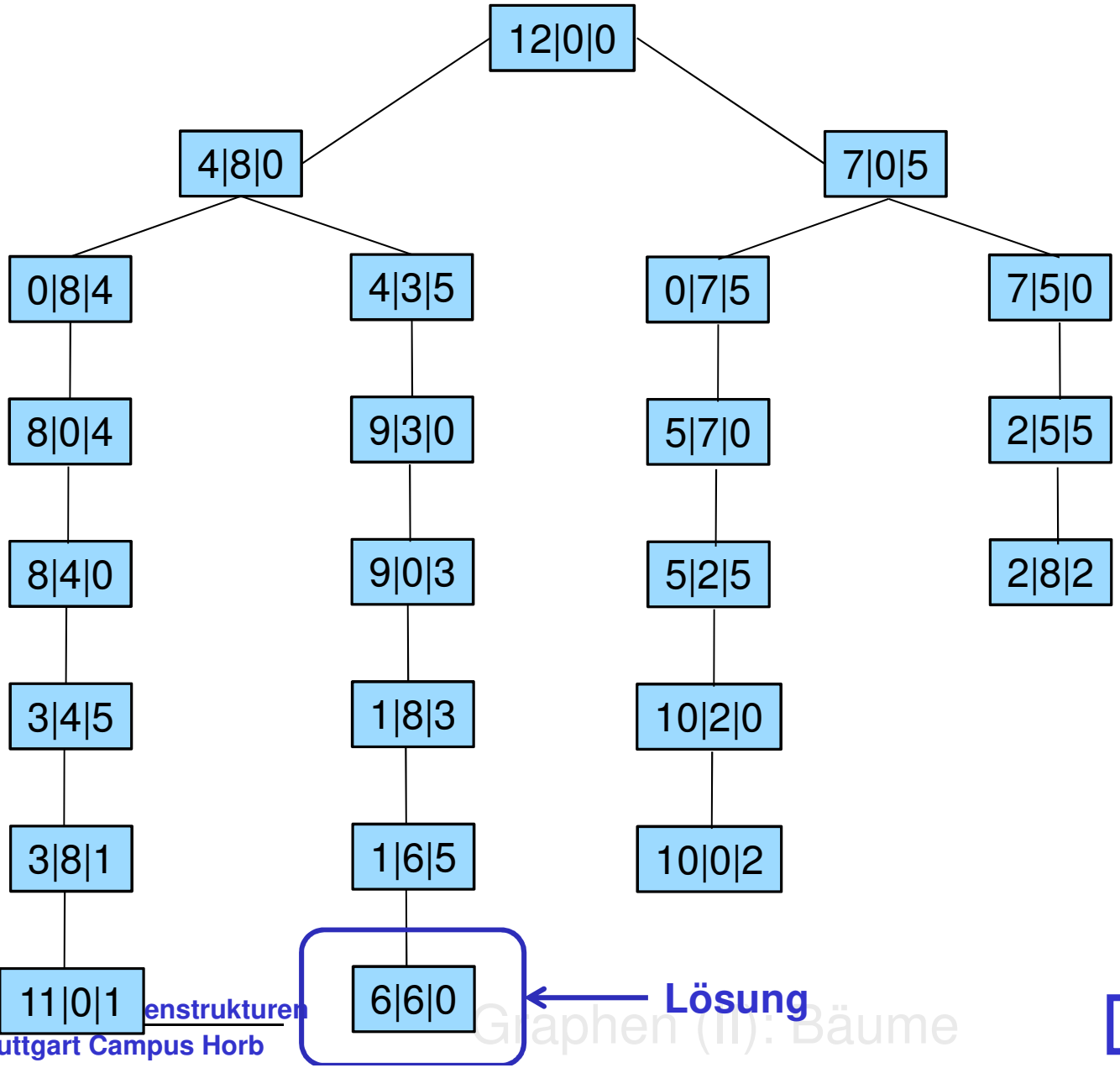


- Fragestellung: Wie viele verschiedene Pentane gibt es?
- Bzw.: Wie viele nichtisomorphe Bäume mit 5 Knoten von Grad 4 und 12 Knoten vom Grad 1 gibt es?
- Lösung: 3 (Beweis siehe Übung)

Entscheidungsbäume (I)

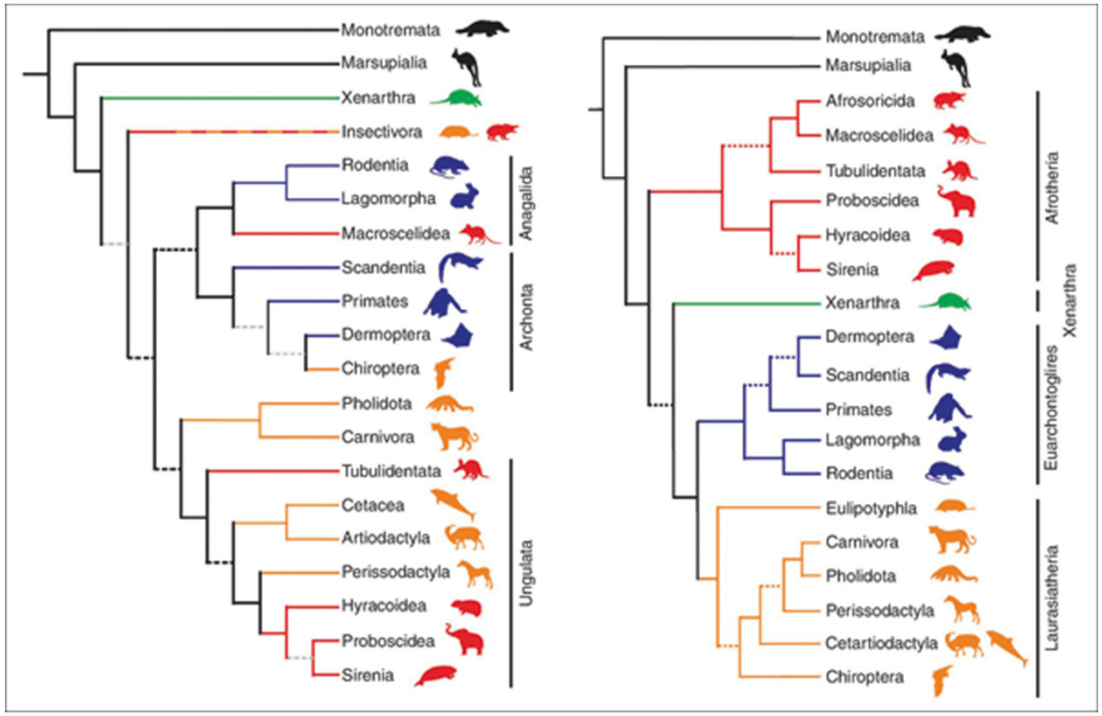
- Beispiel:
  - Behälter mit 12 l Wasser in zwei gleiche Teile aufteilen
  - Hilfsmittel: Zwei Behälter mit 8 bzw. 5 l Fassungsvermögen
  - Frage: Wie oft muss man mindestens umfüllen?
- Vorgehen:
  - Wasserverteilungen der drei Behälter bilden Knoten:
    - Von links nach rechts: 12l-, 8l- und 5l-Gefäß
  - Wurzelknoten: 12|0|0
  - Nachfolger:
    - Mögliche Verteilungen durch Umschütten
    - Nur Knoten, die bisher nicht Bestandteil des Baums

Entscheidungsbäume (II)



Phylogenetischer Baum

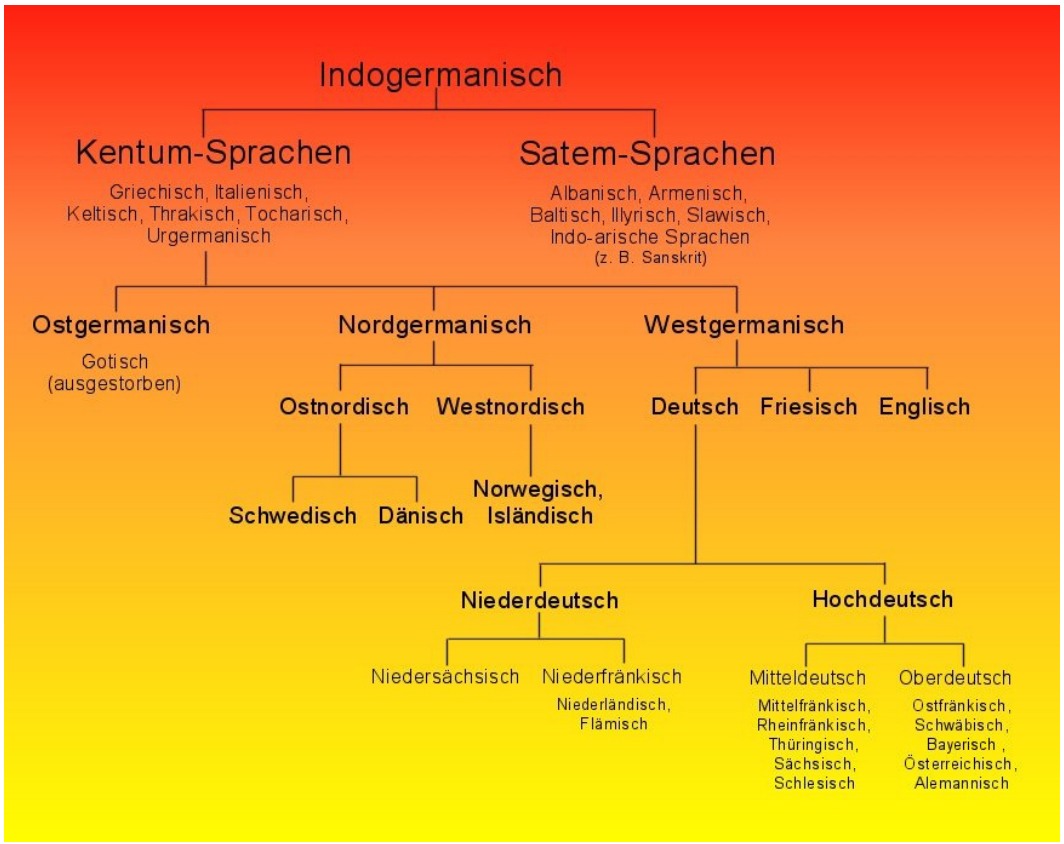
- Geben evolutionäre Entwicklung wieder
- Binäre Verzweigung
- Objekte an Blättern
- Innere Knoten mit oder ohne Label
- Kanten können beschriftet sein (z.B. zeitliche Distanz)
- Beispiel:



[http://www.si-journal.de/index2.php?artikel=jg20/heft2/sij202-s.html]

Entwicklung natürlicher Sprachen

- Binäre oder n-äre Verzweigung
- Innere Knoten Sprachgruppen/-familien
- Blätter aktuelle Sprachen
- Beispiel: Indogermanische Sprachen



http://www.udoklinger.de/Deutsch/Grammatik/Gesch.htm

Übersicht

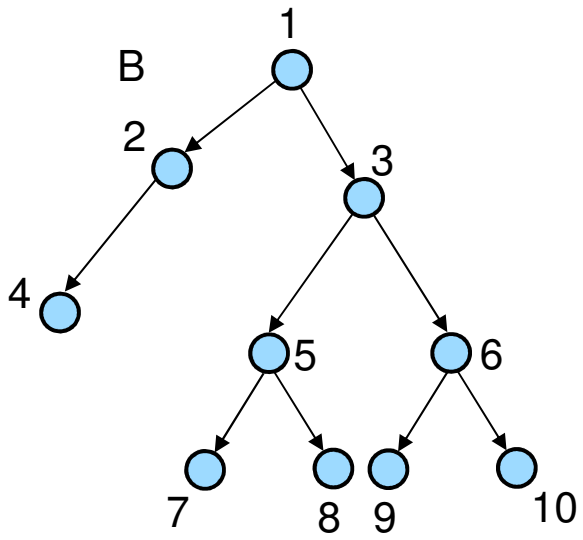
- Definitionen und Eigenschaften
- Datenstrukturen für Bäume
- Anwendungen
- Algorithmen

Traversierung Binärbaum (I)

- Traversierung: Systematisches Abarbeiten aller Knoten
- Drei mögliche Varianten (Einstieg jeweils über Wurzel)
  - Inorder-Durchlauf:
    - Gehe in linken Teilbaum des aktuellen Knoten, führe Inorder rekursiv aus
    - Bearbeite aktuellen Knoten
    - Gehe in rechten Teilbaum des aktuellen Knoten , führe Inorder rekursiv aus
  - Präorder-Durchlauf:
    - Bearbeite aktuellen Knoten
    - Gehe in linken Teilbaum des aktuellen Knoten, führe Präorder rekursiv aus
    - Gehe in rechten Teilbaum des aktuellen Knoten , führe Präorder rekursiv aus
  - Postorder-Durchlauf:
    - Gehe in linken Teilbaum des aktuellen Knoten, führe Postorder rekursiv aus
    - Gehe in rechten Teilbaum des aktuellen Knoten , führe Postorder rekursiv aus
    - Bearbeite aktuellen Knoten

Traversierung Binärbaum (II)

- Beispiel:



  - Inorder-Durchlauf: 4 2 1 7 5 8 3 9 6 10
  - Präorder-Durchlauf: 1 2 4 3 5 7 8 6 9 10
  - Postorder-Durchlauf: 4 2 7 8 5 9 10 6 3 1

Erstellung Phylogenetischer Baum (I)

- Gegeben: Gene von sechs Lebewesen

```
A   ATCGTGGTACTG

                B   CCGGAGAACTAG

          C   AACGTGCTACTG

    D   ATGGTGAAAGTG

                E   CCGGAAAACTTG

F   TGGCCCTGTATC
```

- Beispiel entnommen aus [https://www.youtube.com/watch?v=09eD4A_HxVQ]

Erstellung Phylogenetischer Baum (II)

- Vorverarbeitung:
  - Schritt 1: Ähnlichkeit feststellen: Ausrichtung (Alignment)

```
A   ATCGTGGTACTG
B   CCGGAGAACTAG
C   AACGTGCTACTG
D   ATGGTGAAAGTG
E   CCGGAAAACTTG
F   TGGCCCTGTATC
```

  - Schritt 2:
    - Sequenzen vergleichen
    - Metrik: Anzahl Unterschiede
    - Beispiel: A und B vergleichen

```
A   ATCGTGGTACTG
B   CCGGAGAACTAG
```

    - Differenz (A,B) = 9

- Quelle: Beispiel entnommen aus [https://www.youtube.com/watch?v=09eD4A_HxVQ]

Erstellung Phylogenetischer Baum (III)

- Schritt 3: Erstellen Differenzmatrix

	A	B	C	D	E	F
A		9	2	4	9	10
B			9	6	2	10
C				5	9	10
D					6	10
E						10
F						

Erstellung Phylogenetischer Baum (IV)

- Algorithmus:
  - Finde zwei ähnlichste Einträge E_1 und E_2
  - Bilde Baum aus neuem Knoten und Nachfolgern E_1 und E_2
  - Berechne neue Distanzmatrix mit $\{E_1, E_2\}$ als ein Eintrag
  - Dabei: $\forall x \in E \; mit \; x \neq E_1 und \; x \neq E_2$:

    $$Diff(x, \{E_1, E_2\}) := \frac{Diff(x,E_1)+Diff(x,E_2)}{2}$$

  - Fahre fort, bis Baum erreicht

Erstellung Phylogenetischer Baum (V)

- Erstellen Sie den phylogenetischen Baum!

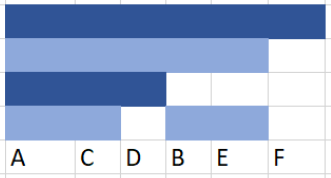
	a	b	c	d	e	f
a	x	9,0	2,0	4,0	9,0	10,0
b	x	x	9,0	6,0	2,0	10,0
c	x	x	x	5,0	9,0	10,0
d	x	x	x	x	6,0	10,0
e	x	x	x	x	x	10,0
f	x	x	x	x	x	x

	a/c	b	d	e	f
a/c	x	9,0	4,5	9,0	10,0
b	x	x	6,0	2,0	10,0
d	x	x	x	6,0	10,0
e	x	x	x	x	10,0
f	x	x	x	x	x

	a/c	b/e	d	f
a/c	x	9,0	4,5	10,0
b/e	x	x	6,0	10,0
d	x	x	x	10,0
f	x	x	x	x

	a/c/d	b/e	f
a/c/d	x	6,75	10,0
b/e	x	x	10,0
f	x	x	x

	a/b/c/d/e	f
a/b/c/d/e	x	8,375
f	x	x



"C:\Users\dh10mbo\OneDrive - Durr Group\Documents\Uni\S2\Algos\PhylogenetischerBaum.xlsx"

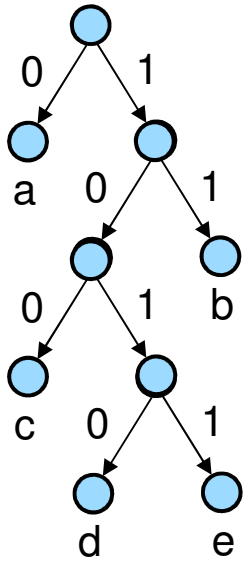
Datenkompression (I)

- Verlustfreie Kompression:
  - Reduzierung der Datenmenge unter Beibehaltung des Informationsgehaltes
  - Ursprüngliche Daten müssen sich aus komprimierten Daten wieder rekonstruieren lassen
- Gegenteil: Verlustbehaftete Kompression

- Idee: Einzelne Zeichen nicht durch Codes konstanter Länge darstellen (wie z.B. im ASCII-Code oder Unicode), sondern die Häufigkeit der Zeichen berücksichtigen
- Häufiger vorkommende Zeichen erhalten dann einen kürzeren Code als die weniger häufig vorkommenden
- Also: Zeichen nach ihrer Häufigkeit sortieren und dann Binärwerte zuordnen, erstes Zeichen erhält 0, zweites 1, drittes 00, viertes 01, ...
- Problem: Rekonstruktion unmöglich (Was bedeutet „00“?)
- Lösung bietet sog. Präfix-Code: Die Darstellung eines Zeichens ist niemals der Anfang der Darstellung eines anderen Zeichens

Datenkompression (II)

- Darstellung Präfix-Code als Binärbaum:
  - Zeichen Blätter, Codierung Weg von Wurzel zum Blatt
  - Kante zu linkem Nachfolger wird mit 0 markiert, Kante zu rechtem Nachfolger mit 1
- Beispiel: Zeichen {a,b,c,d,e}



Zeichen	Code
a	0
b	11
c	100
d	1010
e	1011

  - Code(„bade“) = 11010101011
  - Decode(01010101011101O) = „addbd“

Datenkompression (III)

- Für gegebene Zeichenmenge Präfix-Code nicht eindeutig
- Bewertung: Was ist „guter“ Code?
  - Erste Idee: Maximale Länge eines Codeworts
  - Besser: Mittlere Codewortlänge $l = \sum_{i=1}^{n} p_i \cdot l_i$
  - l_i ist die Länge des i-ten Codeworts, p_i die Häufigkeit seines Auftretens
- Gesucht: Algorithmus, der Präfix-Code mit minimaler mittlerer Wortlänge systematisch konstruiert
- Huffman-Algorithmus:
  - Erzeugt Huffman-Code
  - Aufwand O(n·logn)
  - Vorgeschlagen von David A. Huffman (1952)
- Anmerkung: (Noch) besserer Komprimierungsgrad wird erreicht, wenn nicht einzelne Zeichen, sondern Paare, Tripel oder größere Gruppen von Zeichen der Ausgangsmenge zusammen codiert werden

Huffman-Algorithmus: Prinzip

- Voraussetzungen:
  - Gegeben ist ein Text mit n unterschiedlichen Zeichen
  - Zeichen a_1, a_2, ..., a_n bilden das Alphabet A des Textes
  - Relative Häufigkeit des Auftretens des Zeichens a_i wird mit h_i bezeichnet
  - Dabei gilt: $h_1 + h_2 + \ldots + h_n = 1$
- Vorgehen:
  - Erzeuge für jedes Zeichen einen Wurzelbaum, der aus nur einem Knoten besteht, markiere diesen mit dem Zeichen und der relativen Häufigkeit
  - Die beiden Wurzelbäume mit den geringsten relativen Häufigkeiten werden zu einem neuen Baum vereint, wobei die Wurzel des neuen Baums die summierte relative Häufigkeit der beiden zusammengefassten Bäume bekommt
  - Fahre mit diesem Schritt solange fort, bis nur noch ein Baum existiert
  - Beschrifte jeweils die linke Kante mit 0 und die rechte mit 1
  - Präfix-Code kann nun direkt abgelesen werden
  - Anm. zu Schritt 2: Gibt es mehrere Auswahlmöglichkeiten für die Bäume mit geringster relativer Häufigkeit, können beliebige dieser Bäume genommen werden

Huffman-Algorithmus: Aufgabe (I)

- Gegeben sei der folgende Text, für den ein Präfix-Code mit minimaler mittlerer Codewortlänge konstruiert werden soll:

  `AGNES HAT ANGST.`

- Daraus ergibt sich:

Zeichen	A	E	G	H	N	S	T	<Leer>	.
Absolute Häufigkeit	3	1	2	1	2	2	2	2	1
Relative Häufigkeit	$\frac{3}{16}$	$\frac{1}{16}$	$\frac{2}{16}$	$\frac{1}{16}$	$\frac{2}{16}$	$\frac{2}{16}$	$\frac{2}{16}$	$\frac{2}{16}$	$\frac{1}{16}$

- Führen Sie den Huffman-Algorithmus aus!

Huffman-Algorithmus: Aufgabe (VIII)

- Codieren:
  - Codiere(AGNES HAT ANGST) = 01010011001101110010111010000001010110100111000101
- Decodieren:
  - Decodiere(0101101100110001111000011001110001 01) = ???

Zusammenfassung (I)

- Definitionen:
  - Wald, Baum, Wurzelbaum
  - Wurzel, Blätter, Innere Knoten
  - Niveau, Höhe

- Datenstrukturen für Bäume:
  - Wurzelbaum als Feld
  - Binärbäume mit zwei Verweisen
  - Binärbaum: Adjazenzliste als Feld
  - Binärbaum mit Verzeigerungen
  - Beliebige Bäume als Binärbäume

Zusammenfassung (II)

- Anwendungen:
  - Dateisysteme
  - Compiler
  - Alkane
  - Entscheidungsbäume
  - Präfix-Code

- Algorithmen:
  - Traversierung:
    - Präorder, Inorder, Postorder
  - Konstruktion Phylogenetischer Baum
  - Kompression:
    - Huffman-Algorithmus: Auffinden optimaler Präfix-Code

Übungen (I)

- **Aufgabe 1**

  Finden Sie im folgenden Rätsel fünf Begriffe aus diesem Vorlesungsabschnitt!

```
N R A T L L J P I L S V W D Z H R Z O R
T E I O E S S L K X C T M R X B E J Y S
Z Y T Z L E D O C N A M F F U H D T U E
A Q R O P P L J O O J I M J N V R T T A
M U H U N I L W U B G W R B L X O E Y Y
W O R H T K X I K K D O I A N Z T P Z U
A F E C V Y D P E R M F A E D A S N D W
H Q L H F L I N B W U S Q Z C C O B O G
G B I Y R P M E I W K A B E W T P T Q F
C U D V B L A T T K Y P D Z M F Z S T I
E W E N L Q H W E C A Q G C U H T M F T
M I X Z C V P R B Y J I N R I M K J J E
A M H W S I D H U J S V R L L Y N N N A
U P B I E S X H Y N H B H U X F A I K U
P E S O E W E N J I J T N K N I C I D N
N Y T P X J B G O X F U B D L L K L U U
M M I L S D K J D T D S N J K U T L W M
T U H J D P Z B T Y Y S U I O P S Q I L
F H A K H P X S T I O L Y U F N T U O X
P C F I E C F T Z Q Y Q H G A B I K Y I
```

Übungen (II)

- **Aufgabe 2**

  Für den Zeichenvorrat {v, w, x, y, z} liegen die folgenden Präfix-Codes und die angegebenen Häufigkeiten des Auftretens vor:

Zeichen	Code 1	Code 2	Häufigkeit
v	0	00	35%
w	11	01	20%
x	100	10	30%
y	1010	110	10%
z	1011	111	5%

a) Geben Sie die mittlere Codewortlänge der beiden Codierungen an. Was besagen die Werte?

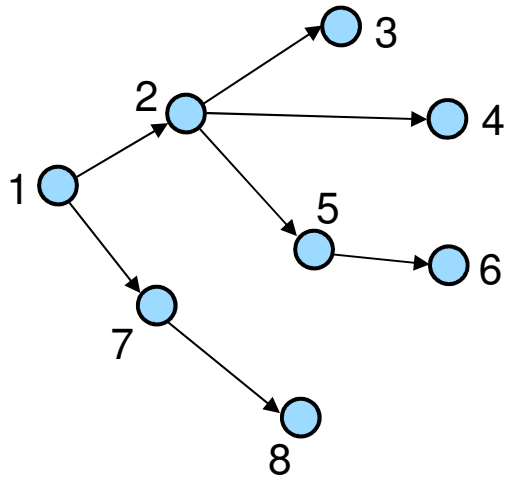
b) Zeichnen Sie Binärbäume der beiden Codes!

c) Durch welche Idee lassen sich Codes mit höherem Komprimierungsgrad erreichen?

Übungen (III)

- **Aufgabe 3**

  Gegeben sei der folgende Baum B.

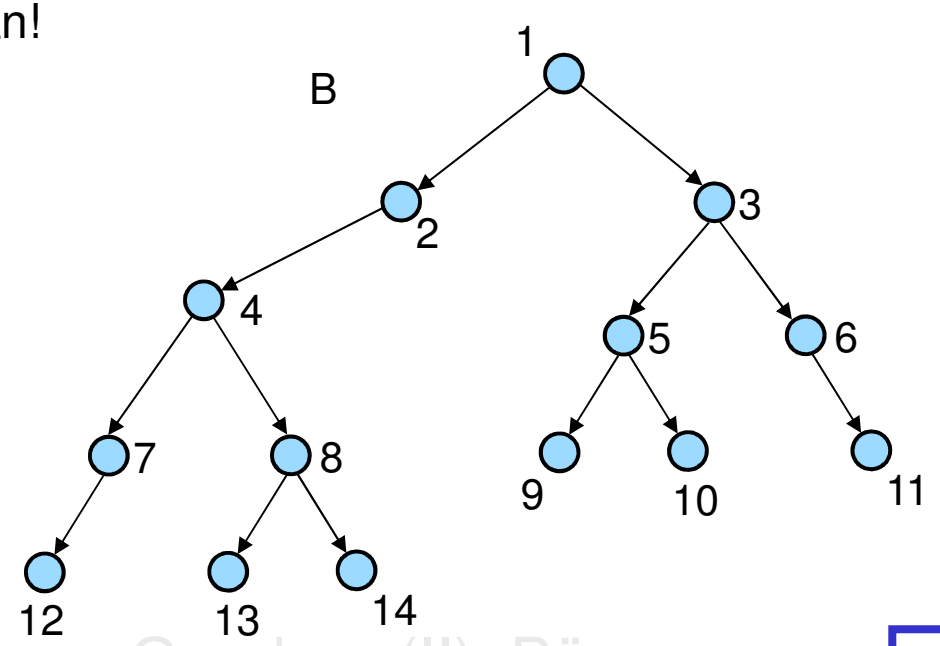


a) Welcher Knoten bildet die Wurzel?
b) Wie viele Blätter hat B?
c) Welches Niveau hat Knoten 4?
d) Ist B ein Binärbaum? Warum (nicht)?
e) Welche Höhe hat B?

Übungen (IV)

- **Aufgabe 4**
  a) Welche Aussagen lassen sich über die Adjazenzmatrix eines Wurzelbaums treffen?
  b) Welche Aussagen lassen sich über die Adjazenzmatrix eines Baums treffen?
  c) Welche Aussagen lassen sich über die Adjazenzmatrix eines Binärbaums treffen?
- **Aufgabe 5**

  Geben Sie für den folgenden Binärbaum B die Knotenfolge von Prä-, In- und Postorder-Durchlauf an!



Übungen (V)

- **Aufgabe 6**

  Bestimmen Sie durch Anwendung des Huffman-Algorithmus einen Präfix-Code für die Zeichen a, b, c, d, e und f mit den Häufigkeiten 0.12, 0.32, 0.04, 0.2, 0.16 und 0.16.

  Bestimmen Sie die mittlere Codewortlänge!

- **Aufgabe 7**

  a) Geben Sie die drei unterschiedlichen Pentane an!

  b) Wie viele unterschiedliche Hexane gibt es? Geben Sie diese an!

Übungen (VI)

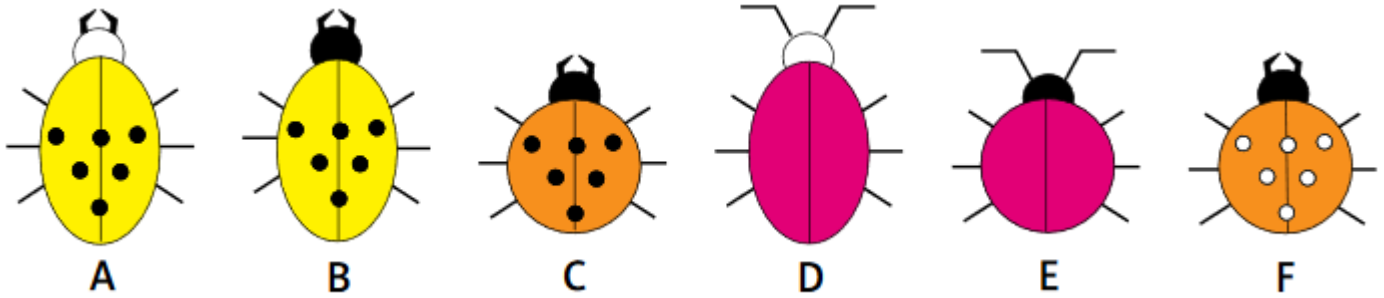
- **Aufgabe 8**

	Wahr	Falsch
In einem Binärbaum hat jeder Knoten zwei ausgehende Kanten.		
Bei der Ausgabe von Dateien aus aktuellem Verzeichnis und allen Unterverzeichnissen kann das Postorder-Verfahren angewendet werden.		
Für einen gegebenen Zeichenvorrat ist der Präfix-Code immer eindeutig.		
Bei einem Postorder-Durchlauf in einem Binärbaum wird die Wurzel stets zuletzt besucht.		
Bäume können keine isolierten Knoten haben.		
Ein Baum ist immer zyklenfrei.		
Schlingen können in Bäumen vorkommen.		
Der Huffman-Algorithmus berechnet einen Präfix-Code mit minimaler Codewortlänge.		

Übungen (VII)

- **Aufgabe 9**

  Gegeben seien die folgenden sechs Käfer:



Erstellen Sie einen Phylogenetischen Baum!

Gehen Sie folgendermaßen vor:

- Erstellen Sie eine Tabelle mit den Merkmalen „Körperform“, „Farbe“, „Kopffarbe“, „Fühler vorhanden“, „Kiefer sichtbar“, „Punkte auf Rücken“, „Punktfarbe“
- Erstellen Sie basierend hierauf eine Differenzmatrix
- Erstellen Sie den Baum

Übungen (VIII)

- **Aufgabe 10**

  a) Wie viele markierte und wie viele strukturell unterschiedliche Binärbäume mit 4 Knoten gibt es?

  b) Zeichnen Sie die strukturell unterschiedlichen Bäume aus Aufgabe a)!

  c) Erläutern Sie die Formel für die Anzahl strukturell unterschiedlicher Binärbäume!