

# Informatik I: Einführung in die Programmierung

## 11. Bäume

Albert-Ludwigs-Universität Freiburg



**UNI  
FREIBURG**

Bernhard Nebel

13. November 2015



- Definition
- Terminologie
- Beispiele

## Der Baum

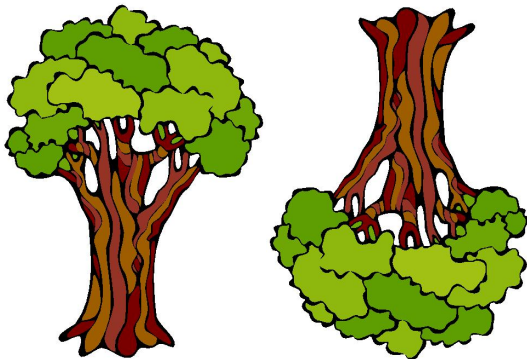
Definition  
Terminologie  
Beispiele

## Binärbäume

## Suchbäume

## Zusammenfassung

- Bäume sind in der Informatik allgegenwärtig.
- Gezeichnet werden sie meistens mit der Wurzel nach oben!



## Der Baum

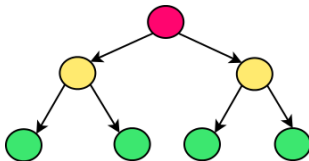
Definition  
Terminologie  
Beispiele

## Binärbäume

## Suchbäume

## Zusammenfassung

- Induktive Definition:
  - Ein **leerer Baum** ist ein Baum.
  - Sind  $t_1, \dots, t_n$  Bäume und ist  $m$  eine **beliebige Markierung**, so ist der **Knoten** bestehend aus  $m$  und den  $n \geq 0$  Teilbäumen  $t_1, \dots, t_n$  ein Baum.
  - Nichts sonst ist ein Baum.
  - Beispiel:



- **Beachte:** Bäume können auch anders definiert werden und können auch eine andere Gestalt haben (z.B. ungerichtet).

Der Baum

Definition

Terminologie

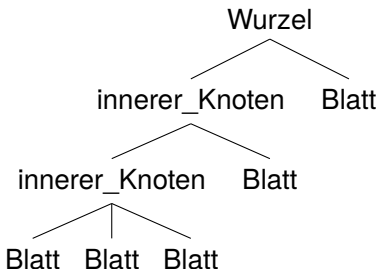
Beispiele

Binärbäume

Suchbäume

Zusammenfassung

- Der Knoten, der nicht Teilbaum eines anderen Baums ist, ist die **Wurzel**.
- Alle Knoten, die keine Teilbäume oder nur leere Teilbäume besitzen, heißen **Blätter**.
- Knoten, die keine Blätter sind, heißen **innere Knoten**.



- Die Wurzel kann also ein Blatt sein (keine weiteren Teilbäume) oder ein innerer Knoten.

Der Baum

Definition

Terminologie

Beispiele

Binärbäume

Suchbäume

Zusammenfassung

- Wenn  $k_1$  ein Knoten ist und  $k_2$  ein Teilbaum von  $k_1$  ist, dann sagt man:
  - $k_1$  ist **Elternknoten** von  $k_2$ ,
  - $k_1$  sowie der Elternknoten von  $k_1$  sowie dessen Elternknoten usw. sind **Vorgänger** von  $k_2$ .
  - $k_2$  ist **Kind** von  $k_1$ .
  - Alle Kinder von  $k_1$ , deren Kinder, usw. sind **Nachfolger** von  $k_1$ .

Der Baum

Definition

Terminologie

Beispiele

Binärbäume

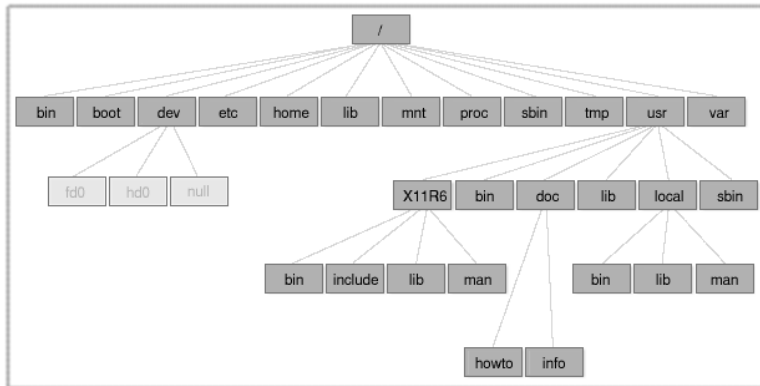
Suchbäume

Zusammenfassung

# Beispiel: Verzeichnisbaum



In Linux (und anderen Betriebssystemen) ist die Verzeichnisstruktur im Wesentlichen baumartig.



Der Baum

Definition  
Terminologie

Beispiele

Binärbäume

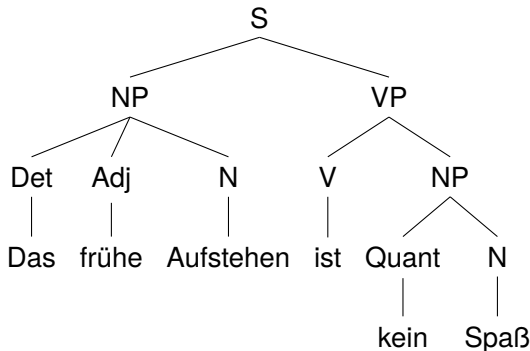
Suchbäume

Zusammenfassung

# Beispiel: Syntaxbaum



Wenn man die Struktur von Sprachen mit Hilfe formaler Grammatiken spezifiziert, dann kann man den Satzaufbau durch sogenannte Syntaxbäume beschreiben.



Der Baum

Definition  
Terminologie

Beispiele

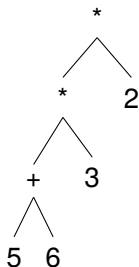
Binärbäume

Suchbäume

Zusammenfassung



- Bäume können arithmetische (und andere) Ausdrücke so darstellen, dass ihre Auswertung eindeutig (und einfach durchführbar) ist, ohne dass man Klammern nutzen muss.
- Beispiel:  $(5 + 6) * 3 * 2$
- Entspricht:  $((5 + 6) * 3) * 2$
- Operatoren als Markierung innerer Knoten, Zahlen als Markierung der Blätter:



Der Baum

Definition  
Terminologie

Beispiele

Binärbäume

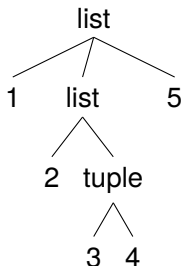
Suchbäume

Zusammenfassung

# Beispiel: Listen und Tupel als Bäume



- Man kann jede Liste und jedes Tupel als Baum verstehen, bei dem der Typ die Knotenmarkierung ist und die Elemente die Teilbäume sind.
- Beispiel:  $[1, [2, (3, 4)], 5]$



Der Baum

Definition  
Terminologie

Beispiele

Binärbäume

Suchbäume

Zusammenfassung

- Repräsentation
- Beispiel
- Baumeigenschaften
- Traversierung

Der Baum

**Binärbäume**

Repräsentation

Beispiel

Baumeigenschaften

Traversierung

Suchbäume

Zusammenfassung

- Der Binärbaum ist ein **Spezialfall eines Baumes**, bei dem jeder Knoten zwei Teilbäume besitzt.
- Blätter sind dann die Knoten, die **zwei leere Teilbäume** besitzen!
- Für viele Anwendungsfälle angemessen.
- Funktionen über solchen Bäumen sind einfach definierbar.

Der Baum

**Binärbäume**

Repräsentation

Beispiel

Baumeigenschaften

Traversierung

Suchbäume

Zusammenfassung

- Der **leere Baum** wird durch `None` repräsentiert.
- Jeder **Knoten** wird durch eine Liste repräsentiert.
- Die **Markierung** ist das erste Element der Liste.
- Der **linke Teilbaum** ist das zweite Element.
- Der **rechte Teilbaum** ist das dritte Element.
- Beispiele:
  - Der Baum bestehend aus dem einzigen Knoten mit der Markierung 8: `[8, None, None]`
  - Der Baum mit Wurzel '+', linkem Teilbaum mit Blatt 5, rechtem Teilbaum mit Blatt 6:  
`['+', [5, None, None], [6, None, None]]`

Der Baum

Binärbäume

Repräsentation

Beispiel

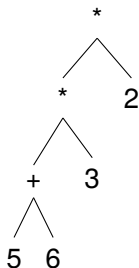
Baumeigenschaften

Traversierung

Suchbäume

Zusammenfassung

# Beispiel: Der Ausdrucksbaum



Der Baum

Binärbäume

Repräsentation

**Beispiel**

Baumeigenschaften

Traversierung

Suchbäume

Zusammenfassung

wird folgendermaßen als verschachtelte Liste dargestellt:

```
[ '*', [ '*', [ '+', [ 5, None, None ],  
                [ 6, None, None ] ],  
        [ 3, None, None ] ],  
[ 2, None, None ] ]
```



- Die **Tiefe eines Knoten  $k$**  ist
  - 0, falls  $k$  die Wurzel ist,
  - $i + 1$ , wenn  $i$  die Tiefe des Elternknotens ist.
- Die **Höhe eines Baumes** ist:
  - 0 für den leeren Baum,
  - $m + 1$ , wenn  $m$  die maximale Tiefe über alle Knoten im Baum ist.
- Die **Größe eines Baumes** ist die Anzahl seiner Knoten.

Der Baum

Binärbäume

Repräsentation

Beispiel

Baumeigenschaften

Traversierung

Suchbäume

Zusammenfassung



Der Baum

Binärbäume

Repräsentation

Beispiel

Baumeigenschaften

Traversierung

Suchbäume

Zusammenfassung

$$\text{height}(\text{tree}) = \begin{cases} 0, & \text{if } \text{tree} \text{ is empty;} \\ 1 + \max(\text{height}(\text{lefttree}), \text{height}(\text{righttree})), & \text{otherwise.} \end{cases}$$

$$\text{size}(\text{tree}) = \begin{cases} 0, & \text{if } \text{tree} \text{ is empty;} \\ 1 + \text{size}(\text{lefttree}) + \text{size}(\text{righttree}), & \text{otherwise.} \end{cases}$$



## Höhe und Größe von Binärbäumen

```
def height(tree):
    if (tree is None):
        return 0
    else:
        return(max(height(tree[1]), height(tree[2])) + 1)

def size(tree):
    if (tree is None):
        return 0
    else:
        return(size(tree[1]) + size(tree[2]) + 1)

tree = [ '*', ['+', [6, None, None], [5, None, None] ],
        [1, None, None] ]
```

## size-Visualisierung

Der Baum

Binärbäume

Repräsentation

Beispiel

Baumeigenschaften

Traversierung

Suchbäume

Zusammenfassung

- Oft sollen alle Knoten eines Baumes besucht und bearbeitet werden.
- 3 Vorgehensweisen (Traversierungen) sind üblich:
  - **Pre-Order** (Hauptreihenfolge): Zuerst der Knoten selbst, dann der linke, danach der rechte Teilbaum
  - **Post-Order** (Nebenreihenfolge): Zuerst der linke, danach der rechte Teilbaum, zum Schluss der Knoten selbst
  - **In-Order** (symmetrische Reihenfolge): Zuerst der linke Teilbaum, dann der Knoten selbst, danach der rechte Teilbaum
  - Manchmal betrachtet man auch **Reverse In-Order** (anti-symmetrische Reihenfolge): Rechter Teilbaum, Knoten, dann linker Teilbaum
  - Auch das Besuchen nach Tiefenlevel von links nach rechts (**level-order**) ist denkbar

Der Baum

Binärbäume

Repräsentation

Beispiel

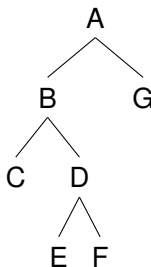
Baumeigenschaften

Traversierung

Suchbäume

Zusammenfassung

- Gebe Baum *pre-order* aus



- Ausgabe: A B C D E F G

Der Baum

Binärbäume

Repräsentation

Beispiel

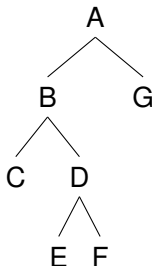
Baumeigenschaften

Traversierung

Suchbäume

Zusammenfassung

- Gebe Baum *post-order* aus



- Ausgabe: C E F D B G A

Der Baum

Binärbäume

Repräsentation

Beispiel

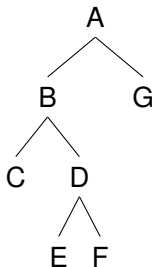
Baumeigenschaften

Traversierung

Suchbäume

Zusammenfassung

- Gebe Baum *in-order* aus.



- Ausgabe: C B E D F A G

Der Baum

Binärbäume

Repräsentation

Beispiel

Baumeigenschaften

Traversierung

Suchbäume

Zusammenfassung



## Post-Order Printing

```
def postorder(node):
    if node is not None:
        postorder(node[1])
        postorder(node[2])
        print(node[0])
tree = [ '*', ['+', [6, None, None], [5, None, None] ],
        [1, None, None] ]
postorder(tree)
```

## Visualisierung

Hinweis: Im Falle von arithmetischen Ausdrücken spricht man bei der *post-order* Ausgabe eines arithmetischen Baumes auch von **umgekehrt polnischer** oder **Postfix**-Notation (HP-Taschenrechner, Programmiersprache *Forth*)

Der Baum

Binärbäume

Repräsentation

Beispiel

Baumeigenschaften

Traversierung

Suchbäume

Zusammenfassung

# 3 Suchbäume

- Definition
- Suche
- Aufbau

Der Baum

Binärbäume

**Suchbäume**

Definition

Suche

Aufbau

Zusammenfassung



- *Suchbäume* realisieren Wörterbücher und dienen dazu, Items schnell wieder zu finden.
- Ein **Suchbaum** ist ein binärer Baum, der die **Suchbaumeigenschaften** erfüllt:
  - Alle Markierungen im linken Teilbaum sind *kleiner* als die aktuelle Knotenmarkierung, alle Markierungen im rechten Teilbaum sind *größer*.
- **Suchen nach einem Item  $m$** : Vergleiche mit Markierung im aktuellem Knoten,
  - wenn gleich, stoppe und gebe True zurück,
  - wenn  $m$  kleiner ist, gehe in den linken Teilbaum,
  - wenn  $m$  größer ist, in den rechten.
- Suchzeit ist proportional zur **Höhe des Baums**! Meist *logarithmisch in der Größe des Baums*.

Der Baum

Binärbäume

Suchbäume

Definition

Suche

Aufbau

Zusammenfassung



## Search in search tree

```
def search(tree, item):
    if tree is None:
        return False
    elif tree[0] == item:
        return True
    elif tree[0] > item:
        return search(tree[1], item)
    else:
        return search(tree[2], item)
```

```
# kleinere Werte im linken, größere im rechten Teilbaum
nums = [10, [5, [1, None, None], None],
        [15, [12, None, None], [20, None, None]]]
print(search(nums, 12))
```

## Visualisierung

Der Baum

Binärbäume

Suchbäume

Definition

Suche

Aufbau

Zusammenfassung



- Aufruf `insert(tree, item)` für das Einsortieren von `item` in `tree`
- Ist `tree` leer, wird der Blattknoten `[item, None, None]` zurückgegeben.
- Wenn die Markierung `tree[0]` größer als `item` ist, wird `item` in den linken Teilbaum eingesetzt (das erhält die Suchbaumeigenschaft!).
- Falls der linke Teilbaum leer ist, müssen wir hier eine **Zuweisung** an `tree[1]` durchführen! Können wir aber auch sonst machen, wenn immer der aktuelle Teilbaum zurückgegeben wird.
- Für den Fall `tree[0]` kleiner als `item` entsprechend.
- Für `tree[0] == item` müssen wir nichts machen.

Der Baum

Binärbäume

Suchbäume

Definition

Suche

Aufbau

Zusammenfassung

## Creating a search tree

```
def insert(tree, item):
    if tree is None:
        return [item, None, None]
    if tree[0] > item:
        tree[1] = insert(tree[1], item)
    elif tree[0] < item:
        tree[2] = insert(tree[2], item)
    return tree

numlist = [10, 15, 20, 12, 5, 1]
tree = None
for key in numlist:
    tree = insert(tree, key)
```

## Visualisierung

Der Baum

Binärbäume

Suchbäume

Definition

Suche

**Aufbau**

Zusammenfassung

# 4 Zusammenfassung



**UNI  
FREIBURG**

Der Baum

Binärbäume

Suchbäume

**Zusammen-  
fassung**



- Der **Baum** ist eine Struktur, die in der Informatik allgegenwärtig ist.
- **Binärbäume** sind Bäume, bei denen jeder Knoten genau zwei Teilbäume besitzt.
- Operationen über (Binär-)Bäumen lassen sich einfach als **rekursive Funktionen** implementieren.
- Es gibt drei Hauptarten der **Traversierung** von Binärbäumen.
- **Suchbäume** sind Binärbäume, die die Suchbaumeigenschaft besitzen, d.h. in linken Teilbaum sind nur kleinere, im rechten nur größere Markierungen.
- Das **Suchen** und **Einfügen** kann durch einfache rekursive Funktionen realisiert werden. **Sortierte Ausgabe** ist auch sehr einfach!

Der Baum

Binärbäume

Suchbäume

Zusammenfassung