# Informatik I: Einführung in die Programmierung

## 11. Bäume

Albert-Ludwigs-Universität Freiburg

Bernhard Nebel

17. November 2017

# 1 Der Baum

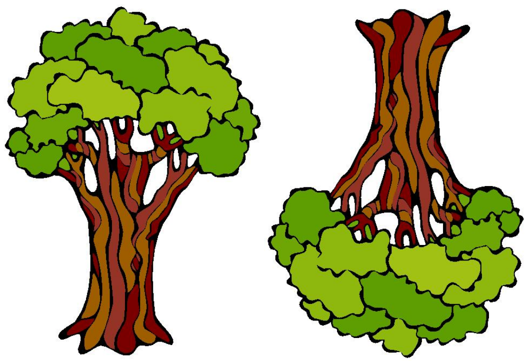
- Definition
- Terminologie
- Beispiele





## Bäume in der Informatik

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

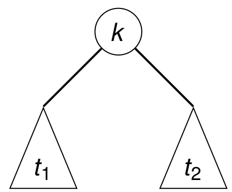






## Bäume in der Informatik - Definition

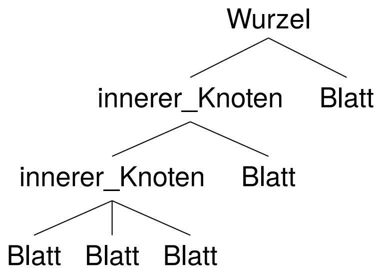
- Induktive Definition:
  - Ein einzelner Knoten $k$ ist ein Baum mit Wurzel $k$.
  - Sei $k$ ein Knoten und seien $t_1, \ldots, t_n$, $n \geq 0$ disjunkte Bäume. Dann ist auch die Struktur bestehend aus der Wurzel $k$ mit zugeordneten Teilbäumen $t_1, \ldots, t_n$ ein Baum.
  - Nichts sonst ist ein Baum.
  - Beispiel:




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

## Terminologie I

- Alle Knoten, denen keine Teilbäume zugeordnet sind, heißen Blätter.
- Knoten, die keine Blätter sind, heißen innere Knoten.




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





## Terminologie II

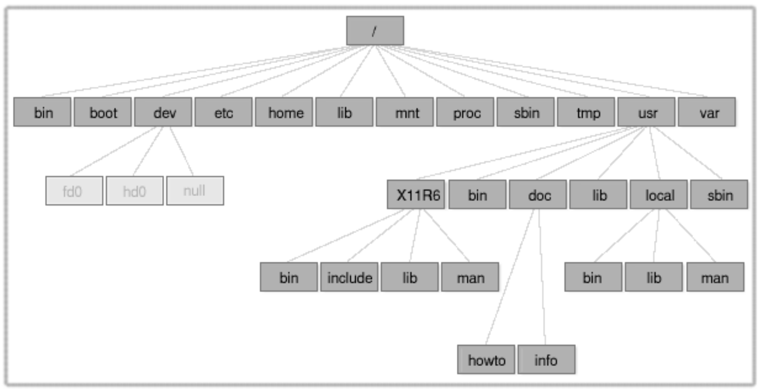
- Wenn $k_1$ ein Knoten und $k_2$ die Wurzel eines zugeordneten Teilbaums ist, dann sagt man:
  - $k_1$ ist Elternknoten von $k_2$,
  - $k_1$ sowie der Elternknoten von $k_1$ sowie dessen Elternknoten usw. sind Vorgänger von $k_2$.
  - $k_2$ ist Kind von $k_1$.
  - Alle Kinder von $k_1$, deren Kinder,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





## Beispiel: Verzeichnisbaum

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

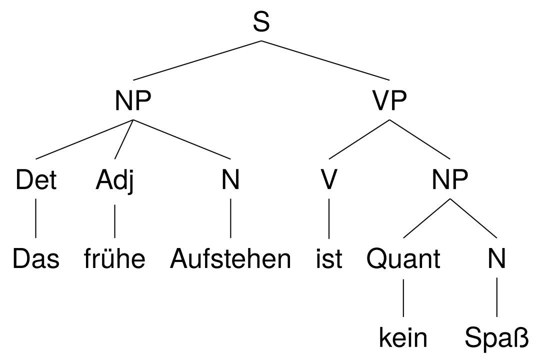







## Beispiel: Syntaxbaum

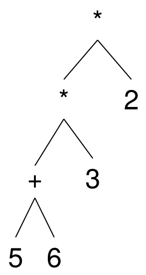
Wenn man die Struktur von Sprachen mit Hilfe formaler Grammatiken spezifiziert, dann kann man den Satzaufbau durch sogenannte Syntaxbäume beschreiben.



# Beispiel: Ausdrucksbaum



- 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:

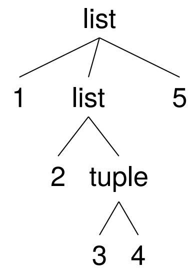





# Beispiel: Listen und Tupel als Bäume



- 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]`






# 2 Binärbäume



- Repräsentation
- Beispiel
- Baumeigenschaften
- Traversierung



# Der Binärbaum



- Der Binärbaum ist ein Spezialfall eines Baumes, bei dem jeder innere Knoten zwei Teilbäume besitzt.
- Für viele Anwendungsfälle angemessen.
- Funktionen über solchen Bäumen sind einfach definierbar.

# Binärbäume durch Listen repräsentieren

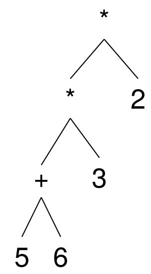


- Jeder Knoten wird durch eine Liste repräsentiert.
- Die Markierung ist das erste Element der Liste.
- Der linke Teilbaum ist das zweite Element. Bei Blattknoten steht hier None.
- Der rechte Teilbaum ist das dritte Element. Bei Blattknoten steht hier None.
- Beispiele:
  - Der Baum bestehend aus dem 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]]`



# 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 ] ]
```



# Tiefe von Knoten, Höhe und Größe von (Binär-)Bäumen

Der Baum
Binärbäume
Repräsentation
Beispiel
Baumeigenschaften
Traversierung
Suchbäume
Zusammenfassung

- Die Tiefe eines Knoten $k$ (Abstand zur Wurzel) ist
  - 0, falls $k$ die Wurzel ist,
  - $i+1$, wenn $i$ die Tiefe des Elternknotens ist.
- Die Höhe eines Baumes ist die maximale Tiefe über alle Blätter:
  - 0 für den Baum, bei dem die Wurzel ein Blatt ist,
  - $m+1$, wenn $m$ die maximale Tiefe aller der Wurzel zugeordneten Teilbäume ist.
- Die Größe eines Baumes ist die Anzahl seiner Knoten.



# Rekursive Definition von Höhe und Größe von Binärbäumen

Der Baum
Binärbäume
Repräsentation
Beispiel
Baumeigenschaften
Traversierung
Suchbäume
Zusammenfassung

$$height(tree) = \begin{cases} 0, & \text{if } tree \text{ has only root} \\ 1 + \max(\ height(lefttree), \\ \qquad height(righttree)), & \text{otherwise.} \end{cases}$$

$$size(tree) = \begin{cases} 1, & \text{if } tree \text{ has only root;} \\ 1 \ + size(lefttree) \\ \qquad + size(righttree)), & \text{otherwise.} \end{cases}$$

# Funktionen für Höhe und Größe



## Höhe und Größe von Binärbäumen

```
def height(tree):
    if (tree is None):
        return -1
    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



# Traversierung von Bäumen



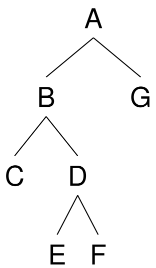
- Oft sollen alle Knoten eines Baumes besucht und bearbeitet werden.
- 3 Vorgehensweisen (Traversierungen) sind üblich:
  - Pre-Order (Hauptreihenfolge): Zuerst der Knoten selbst, dann der linke, danach der rechte Teilbaum
  - Post-Order (Nebenreihenfolge): Zuerst der linke, danach der rechte Teilbaum, 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



# Pre-Order Ausgabe eines Baums



- Gebe Baum *pre-order* aus



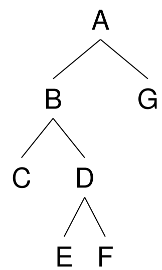

- Ausgabe: A B C D E F G



# Post-Order Ausgabe eines Baums



- Gebe Baum *post-order* aus

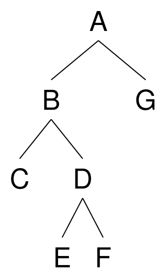



- Ausgabe: C E F D B G A

# In-Order Ausgabe eines Baums



- Gebe Baum *in-order* aus.




- Ausgabe: C B E D F A G



# Post-order Python-Programm



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*)



# 3 Suchbäume



- Definition
- Suche
- Aufbau



# Suchbäume



- *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*.

## Python-Suche im Suchbaum



### 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





## Aufbauen eines Suchbaums



- 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.





## Suchbaumaufbau in Python



### 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





# 4 Zusammenfassung

## Zusammenfassung

- 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!