6. Bäume

Lernziele:

- Definition und Eigenschaften binärer Bäume kennen,
- Traversierungsalgorithmen für binäre Bäume implementieren können,
- die Bedeutung von Suchbäumen für die effiziente Implementierung verschiedenen abstrakter Datentypen erläutern können und
- Ausgleichstechniken für binäre Bäume verstehen und einsetzen können.

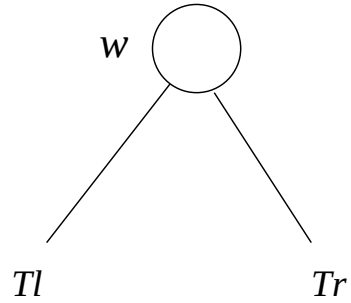
☞ Wir beschränken uns in diesem Kapitel auf Binärbäume.

☞ Datenstrukturen für die effiziente Suche im Hauptspeicher.

Binärbäume

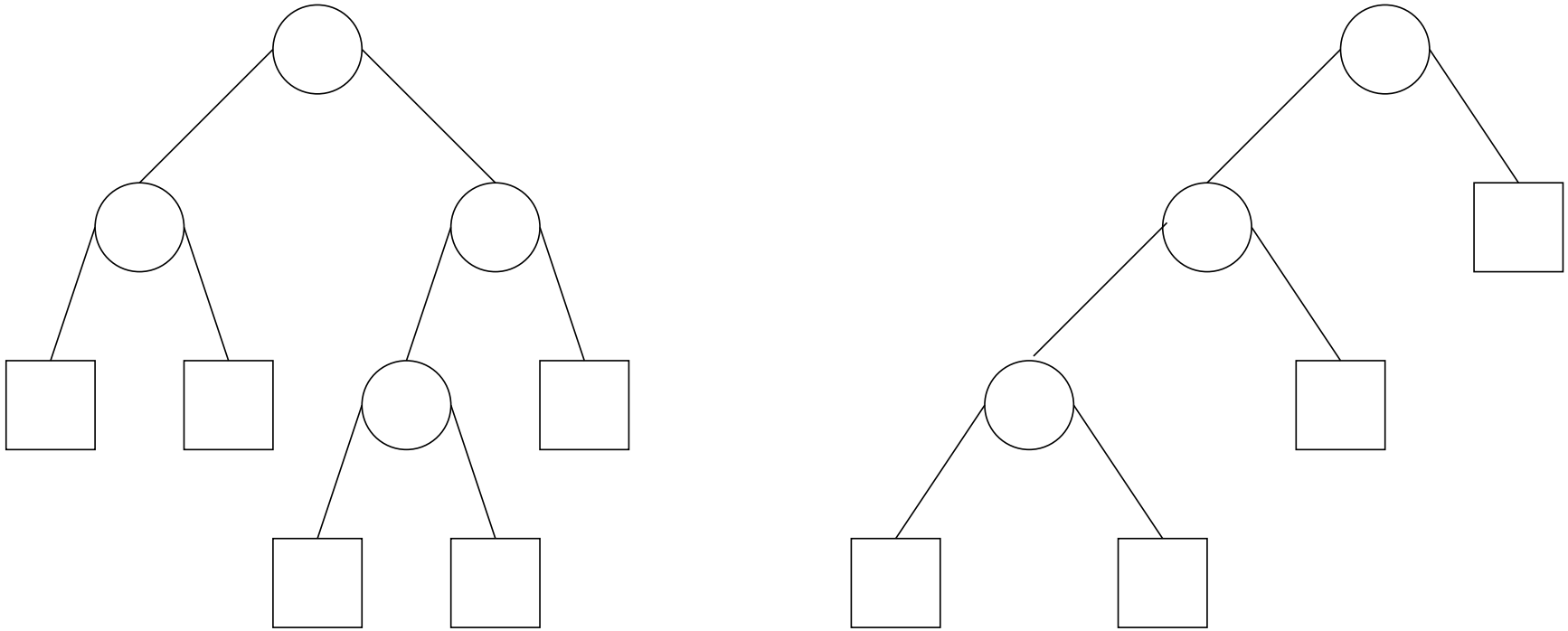
Definition 6.1. Ein *Binärbaum* ist wie folgt definiert:

- Der *leere Baum*, bezeichnet durch $\square$, ist ein Binärbaum.
- Wenn T_l und T_r Binärbäume sind, dann ist auch



ein Binärbaum. Der Knoten w ist die *Wurzel* des Binärbaums, T_l der linke und T_r der rechte Unterbaum von w.

Beispiel 6.1. Zwei binäre Bäume:



Knotenanzahl binärer Bäume

Definition 6.2. Die Knoten ◯ bezeichnet man auch als *innere Knoten*, die leeren Bäume □ als *externe Knoten*.

Lemma 6.1. Ein binärer Baum mit n inneren Knoten hat genau $n + 1$ externe Knoten.

Beweis: Vollständige Induktion über die Anzahl der inneren Knoten

Induktionsanfang: $n = 0$

z.Z.: "Ein binärer Baum mit 0 inneren Knoten hat genau einen externen Knoten."

Es gibt nur einen binären Baum, der keinen inneren Knoten hat: der leere Baum. Der leere Baum hat genau einen externen Knoten. Also gilt die Aussage für $n = 0$.

Induktionsschritt: $n \to n + 1$

z.Z. "Wenn die Aussage für alle binären Bäume mit höchsten n inneren Knoten gilt, dann gilt sie auch für alle Bäume mit $n + 1$ inneren Knoten."

Induktionsvoraussetzung (Wenn-Teil): Für $0 \leq m \leq n$ gilt: Ein binärer Baum mit m inneren Knoten hat $m + 1$ externe Knoten.

Induktionsbehauptung (Dann-Teil): Ein beliebiger binärer Baum mit $n + 1$ inneren Knoten hat $n + 2$ externe Knoten.

Es sei T ein beliebiger binärer Baum mit $n + 1$ inneren Knoten. Es sei w die Wurzel von T, T_l sei der linke Unterbaum von w und T_r der rechte Unterbaum.

n_l sei die Anzahl der inneren Knoten von T_l, n_r sei die Anzahl der inneren Knoten von T_r.

Es gilt: $n_l + n_r = n$, denn T_l und T_r enthalten alle der $n + 1$ inneren Knoten von T bis auf die Wurzel.

Aus der Induktionsvoraussetzung folgt: T_l enthält $n_l + 1$ externe Knoten, T_r enthält $n_r + 1$ externe Knoten.

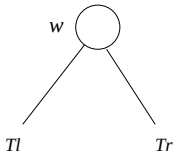
Alle externen Knoten von T sind in T_l oder T_r. Also hat T

$$(n_l + 1) + (n_r + 1) = (n_l + n_r) + 2 = n + 2$$

externe Knoten.

Höhe binärer Bäume

Definition 6.3. Die Höhe $h(T)$ eines binären Baums T ist wie folgt definiert:

- Für den leeren Baum gilt: $h(\square) = 0$
- Für einen nicht-leeren Baum mit $T =$ 

gilt:

$$h(T) = 1 + \max\{h(T_l), h(T_r)\}$$

Lemma 6.2.

(a) Ein binärer Baum der Höhe h hat höchstens 2^h externe und höchstens $2^h - 1$ innere Knoten.

(b) Für jeden Binärbaum T mit n inneren Knoten gilt

$$h(T) \geq \log_2(n + 1)$$

Beweis: (a) Vollständige Induktion über die Höhe h.

$h = 0$: Nur der leere Baum hat die Höhe 0 und er hat $2^0 = 1$ externe Knoten hat.

$h \to h + 1$: Es sei T ein binärer Baum mit Höhe $h + 1$, T_l sei der linke und T_r sei der rechte Unterbaum der Wurzel von T.

Die Höhe von T_l und T_r ist höchsten h. Nach Induktionsvoraussetzung enthalten T_l und T_r jeweils höchstens 2^h externe Knoten.

T enthält somit höchstens $2^h + 2^h = 2 \cdot 2^h = 2^{h+1}$ externe Knoten.

Die Formel für die Anzahl der internen Knoten folgt aus Lemma 6.1.

(b) Aus der Formel für die inneren Knoten folgt:

$$\begin{aligned} n \leq 2^{h(T)} - 1 \quad &\Leftrightarrow \quad n + 1 \leq 2^{h(T)} \\ &\Leftrightarrow \quad \log_2(n + 1) \leq h(T) \end{aligned}$$

Traversierungsalgorithmen

- Sehr häufig müssen alle Knoten eines Baumes aufgesucht und bearbeitet werden.
- Verfahren hierfür werden auch *Traversierungs-* oder *Durchlaufalgorithmen* genannt.
- Die folgenden Traversierungsalgorithmen für Binärbäume sind von besonderer Bedeutung.
- Diese Verfahren basieren direkt auf der rekursiven Definition eines Binärbaums.

Preorder-Traversierung

Algorithmus 6.1. [Preorder-Traversierung] Bei der *Preorder-Traversierung* wird ein Binärbaum T wie folgt durchlaufen:

- Falls T leer ist, tue nichts,
- ansonsten führe die folgenden Schritte in der genannten Reihenfolge aus:
  - Behandle die Wurzel w von T,
  - durchlaufe T_l in Preorder und
  - durchlaufe T_r in Preorder.

Inorder-Traversierung

Algorithmus 6.2. [Inorder-Traversierung] Bei der *Inorder-Traversierung* wird ein Binärbaum T wie folgt durchlaufen:

- Falls T leer ist, tue nichts,
- ansonsten führe die folgenden Schritte in der genannten Reihenfolge aus:
  - Durchlaufe den linken Unterbaum T_l in Inorder,
  - behandle die Wurzel w von T und
  - durchlaufe den rechten Unterbaum T_r in Inorder.

Postorder-Traversierung

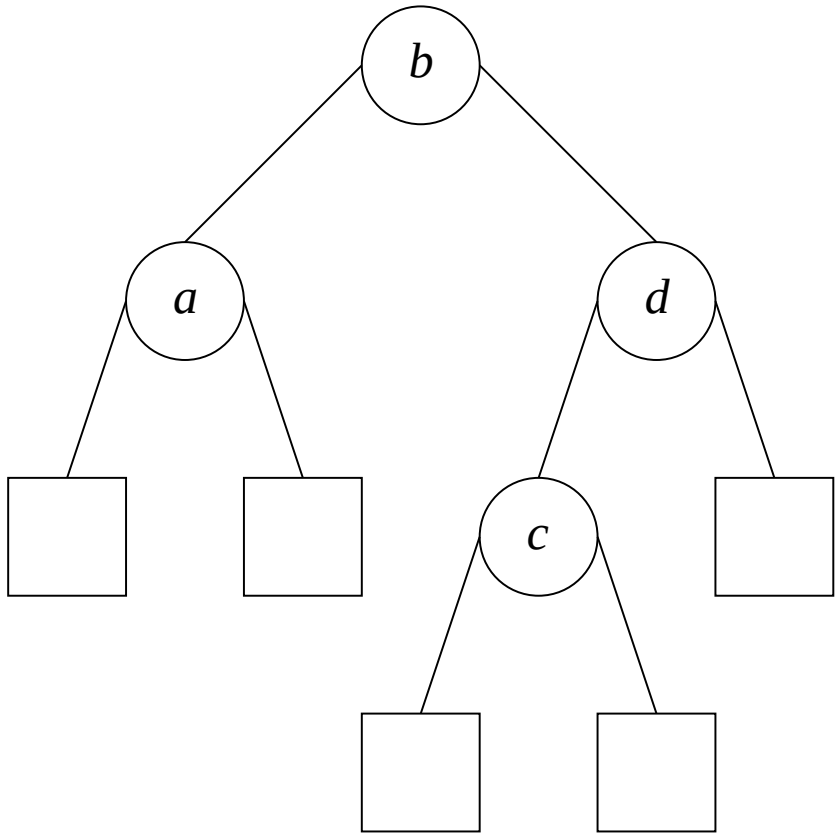
Algorithmus 6.3. [Postorder-Traversierung] Bei der *Postorder-Traversierung* wird ein Binärbaum T wie folgt durchlaufen:

- Falls T leer ist, tue nichts,
- ansonsten führe die folgenden Schritte in der genannten Reihenfolge aus:
  - Durchlaufe den linken Unterbaum T_l in Postorder,
  - durchlaufe den rechten Unterbaum T_r in Postorder und
  - behandle die Wurzel w von T.

Beispiel Traversierungsalgorithmen

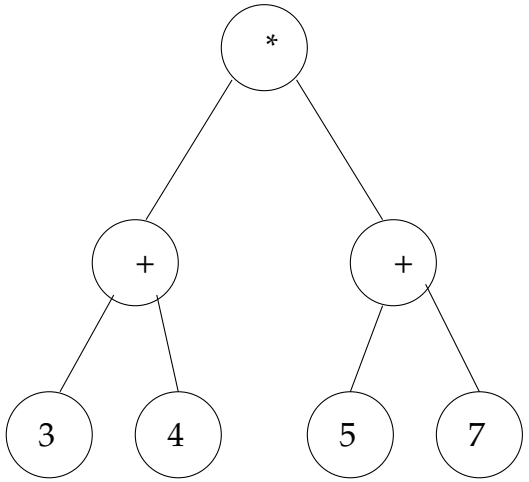
Beispiel 6.2.

- Preorder: *b a d c*
- Inorder: *a b c d*
- Postorder: *a c d b*



Repräsentation arithmetischer Ausdrücke

Beispiel 6.3. Repräsentation des arithmetischen Ausdrucks $(3+4)*(5+7)$ als Binärbaum:



Inorder (nicht eindeutig): $3 + 4 * 5 + 7$

Postorder (UPN, eindeutig): $3\,4 + 5\,7 + *$

Bemerkungen zu den Traversierungsalgorithmen

- Preorder bzw. Postorder liefern eine injektive Abbildung von binären Strukturbäumen für arithmetische Ausdrücke in Strings.
- Ergebnis von Preorder bzw. Postorder ist dann die sog. klammerfreie Präfix- bzw. Postfix-Notation (Umgekehrte Polnische Notation, UPN).
- Inorder dagegen ist nicht injektiv. Die Infixnotation wird erst durch Klammern eindeutig.