Algorithmen und Datenstrukturen I

Natürliche Bäume

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 18.11.2019 07:38

Inhaltsverzeichnis

Bäume — Begriffe

- Bäume sind verallgemeinerte Listenstrukturen
- ein Element heisst **Knoten**
- statt einem Nachfolger, eine endliche Anzahl von Nachfolgern
- einziger Knoten, der keinen Vorgänger hat, heisst **Wurzel**
- jeder andere Knoten hat **genau einen** Vorgänger

- eine Folge $p_0, ..., p_k$ für die gilt p_{i+1} ist Vorgänger von p_i mit $0 \leq i < k$ heisst **Pfad** der Länge k, der p_0 mit p_k verbindet
- jeder von der Wurzel verschiedene Knoten ist durch genau einen Pfad mit der Wurzel verbunden
- ist unter den Nachfolgern eine Ordnung definiert (erster Nachfolger, zweiter Nachfolger, …), so heisst der Baum **geordnet**
- die **Ordnung eines Baumes** ist die maximale Anzahl von Nachfolgern eines Knotens
- **Binärbäume** haben die Ordnung 2 und sind geordnet (linker und rechter Nachfolger)
- Knoten ohne Nachfolger heissen **Blätter**
- Knoten mit Nachfolgern heissen **innere Knoten**
- Anzahl der Nachfolger heisst **Rang** eines Knotens

Rekursive Definition von Bäumen

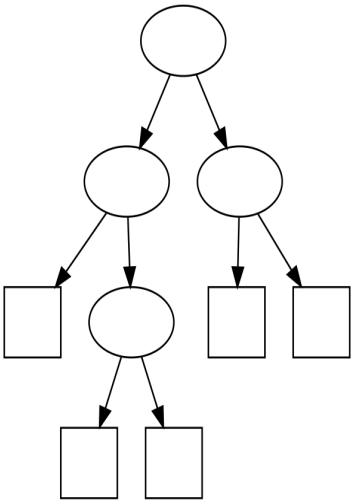
- Menge aller Bäume der Ordnung $d, d \geq 1$ lassen sich definieren durch:
- der aus einem einzelnen Knoten bestehende Baum ist ein Baum der Ordnung d
- sind $t_1, ..., t_d$ beliebige Bäume der Ordnung d, so erhält man einen neuen Baum der Ordnung d indem man die Wurzeln von $t_1, ..., t_d$ zu Nachfolgern einer neuen Wurzel w macht
- t_i für $0 \leq i \leq d$ heisst i-ter Teilbaum von w

Höhen und Tiefen

- die **Höhe eines Baumes** ist der maximale Abstand eines Blattes von der Wurzel
  - Höhe des Baumes aus einem Knoten ist 0
  - Höhe des Baumes ist die Höhe des höchsten Teilbaums der Wurzel + 1
- die **Tiefe eines Knotens** ist sein Abstand zur Wurzel, d.h. die Anzahl der Kanten des Pfades von der Wurzel zu diesem Knoten
- alle Knoten mit der gleichen Tiefe i, liegen auf dem **Niveau** i
- hat ein Baum auf jedem Niveau die maximale Anzahl möglicher Knoten und haben sämtliche Blätter die selbe Tiefe, heisst er **vollständig**

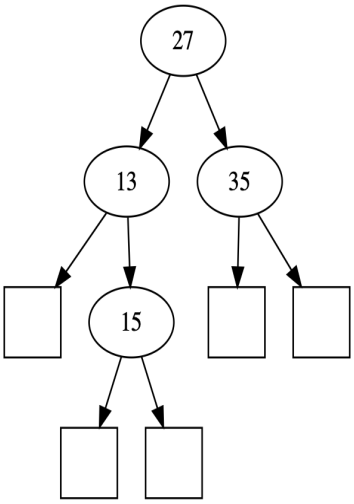
Natürliche Bäume

- Annahme: Alle Schlüssel sind ganzzahlig und paarweise verschieden
- **Suchbäume** speichern die Schlüssel nur in den inneren Knoten
- **Blattsuchbäume** speichern die Schlüssel nur in den Blättern



Suchbäume

- für jeden Knoten p gilt
  - alle Schlüssel im linken Teilbaum von p sind kleiner als der Schlüssel von p
  - alle Schlüssel im rechten Teilbaum von p sind größer als der Schlüssel von p
- die Blätter repräsentieren die Intervalle der in den inneren Knoten gespeicherten Schlüssel

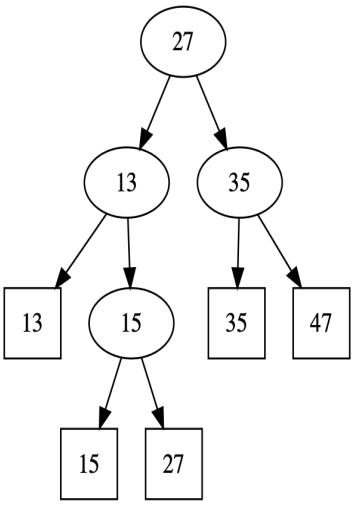


Suche im Suchbaum

- C++-Code als Aufgabe auf nächstem Aufgabenblatt
- Suche nach x im Baum p
  - Fall 1: p ist innerer Knoten
    - Fall 1.1: $p.key \equiv x \Rightarrow$ gefunden
    - Fall 1.2: $x < p.key \Rightarrow$ suche rekursiv im linken Teilbaum von p
    - Fall 1.3: $x > p.key \Rightarrow$ suche rekursiv im rechten Teilbaum von p
  - Fall 2: p ist Blatt $\Rightarrow$ nicht gefunden

Blattsuchbaum

- Blätter speichern die Schlüssel
- innere Knoten speichern Wegweiser
- Suche
  - folgt "Wegweiser"
  - endet immer an Blatt



Implementierung von Bäumen

- Arrays mit Adressrechnung für Nachfolger, z.B. wie bei Heapsort
- Zeiger

```
struct Node {
  int key;
  Node *left;
  Note *right;
}
```

- Baum ist dann ein Zeiger auf den Wurzelknoten

  ```
  class Tree {
    Node *root;
  }
  ```

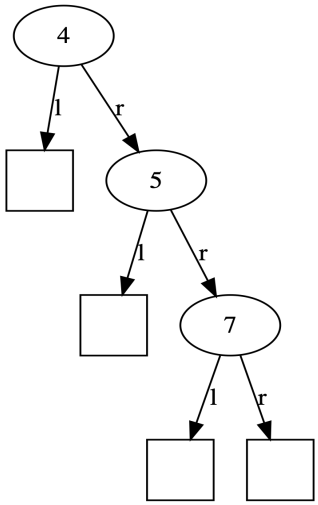
- üblicherweise in Suchbäumen Blätter als `nullptr`

Einfügen in Suchbäumen

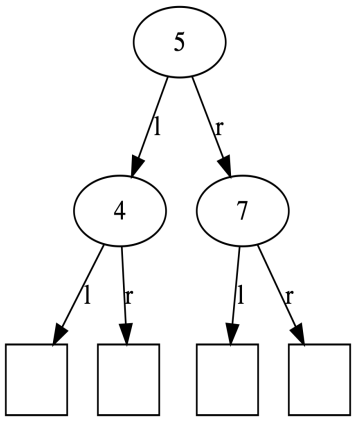
- zunächst wie Suche
- Schlüssel gefunden ⇒ fertig
- bei Blatt angekommen:
  - neuen inneren Knoten erstellen mit zwei Blättern als Nachfolger
  - Blatt durch diesen ersetzen
- der Baum, der so aus einem leeren Baum entsteht, heisst *natürlicher* Baum

Problem beim Einfügen

- der Baum hängt von der Reihenfolge des Einfügens ab
- Reihenfolge 4, 5, 7



- Reihenfolge 5, 4, 7



Entfernen eines Schlüssels aus einem Suchbaum

- zunächst wie Suche
- Schlüssel nicht gefunden ⇒ fertig
- Schlüssel gefunden bei Knoten p der nur Blätter als Nachfolger hat ⇒ p ersetzen durch Blatt
- Schlüssel gefunden bei Knoten p der einen inneren Knoten q und ein Blatt als Nachfolger hat ⇒ p ersetzen durch q
- Schlüssel gefunden bei Knoten p der zwei innere Knoten als Nachfolger hat
  - bestimme den Knoten q im rechten Teilbaum, der den **kleinsten** Schlüssel hat (heisst **symmetrischer Nachfolger** von p)
    * q hat als linken Nachfolger ein Blatt
  - entferne q und ersetze den Schlüssel von p mit dem Schlüssel von q
- statt dem symmetrischen Nachfolger kann auch der **symmetrische Vorgänger** gweählt werden
  - Knoten mit **größtem** Schlüssel in linkem Teilbaum

Durchlaufordnungen von Binärbäumen

- Inspizieren aller Knoten um z.B. bestimmte Eigenschaften
  - von Knoten,
  - der in den Knoten gespeicherten Schlüsseln oder
  - der Struktur des Baumes zu ermitteln

- die drei wichtigstens Reihenfolgen:
  - **Hauptreihenfolge** (engl. *preorder*)
  - **Nebenreihenfolge** (engl *postorder*)
  - **symmetrische Reihenfolge** (engl. *inorder*)
- *pre*, *post* und *in* sagt aus, wann die Wurzel besucht wird
- Hauptreihenfolge (preorder)
  - besuche die Wurzel p
  - durchlaufe den linken Teilbaum von p in Hauptreihenfolge
  - durchlaufe den rechten Teilbaum von p in Hauptreihenfolge
- Nebenreihenfolge (postorder)
  - durchlaufe den linken Teilbaum von p in Nebenreihenfolge
  - durchlaufe den rechten Teilbaum von p in Nebenreihenfolge
  - besuche die Wurzel p
- symmetrische Reihenfolge (inorder)
  - durchlaufe den linken Teilbaum von p in sym. Reihenfolge
  - besuche die Wurzel p
  - durchlaufe den rechten Teilbaum von p in sym. Reihenfolge