Algorithmen und Datenstrukturen I

Splay-Bäume

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 18.03.2018 18:16

Inhaltsverzeichnis

Idee

- ähnlich wie bei Listen: Strategien zur Selbstanordnung
- Ziel: möglichst ohne explizite Speicherung von Balancefaktoren o.ä. eine Strukturanpassung an unterschiedliche Zugriffshäufigkeiten
- z.B statt *move-to-front* in Listen, *move-to-root*
  - realisiert durch Rotationen

Splay-Bäume

- **Splay-Bäume** sind reine binäre Suchbäume
  - d.h. ohne Speicherung von Zusatzinformation

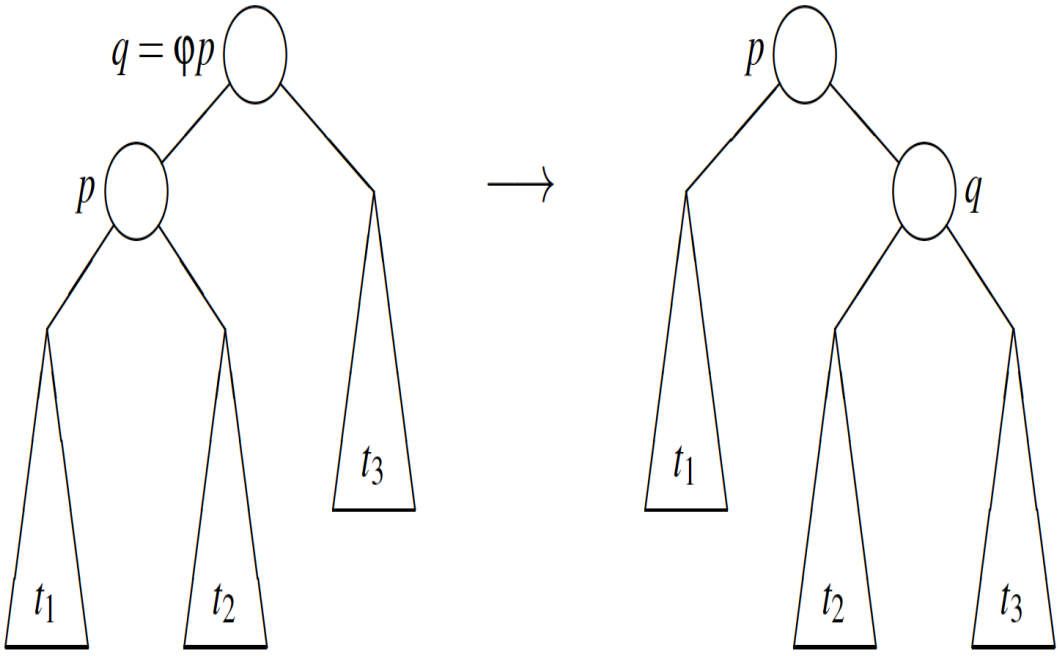
- ordnen sich selbst durch eine Variante der *Move-to-Root*-Strategie
- wichtigste Operation ist die **Splay-Operation**
  - verbreitert den Suchbaum so, dass jeder Schlüssel x auf den zugegriffen wurde zur Wurzel bewegt wird und
  - durch geschickte Zusammenfassung der Rotationen zu Paaren:
    * Länge der Pfade zu Schlüsseln auf dem Suchpfad zu x halbieren sich etwa

Die Splay-Operation

- sei t ein binärer Suchbaum und x ein Schlüssel
- dann ist das Ergebnis der Operation $Splay(t, x)$ der binäre Suchbaum, den man wie folgt erhält:

  Schritt 1: Suche nach x. Sei p der Knoten bei dem die erfolgreiche Suche endet, bzw. Vorgänger des Blattes bei dem die erfolglose Suche endet.
  Schritt 2: Wiederhole die Operationen **zig**, **zig-zig** und **zig-zag** beginnend bei p solange, bis sie nicht mehr ausführbar sind, weil p Wurzel geworden ist.

- Fall 1: p hat Vorgänger φp und φp ist Wurzel
- dann führe die Operation **zig** aus, d.h. eine Rotation nach links oder rechts, die p zur Wurzel macht

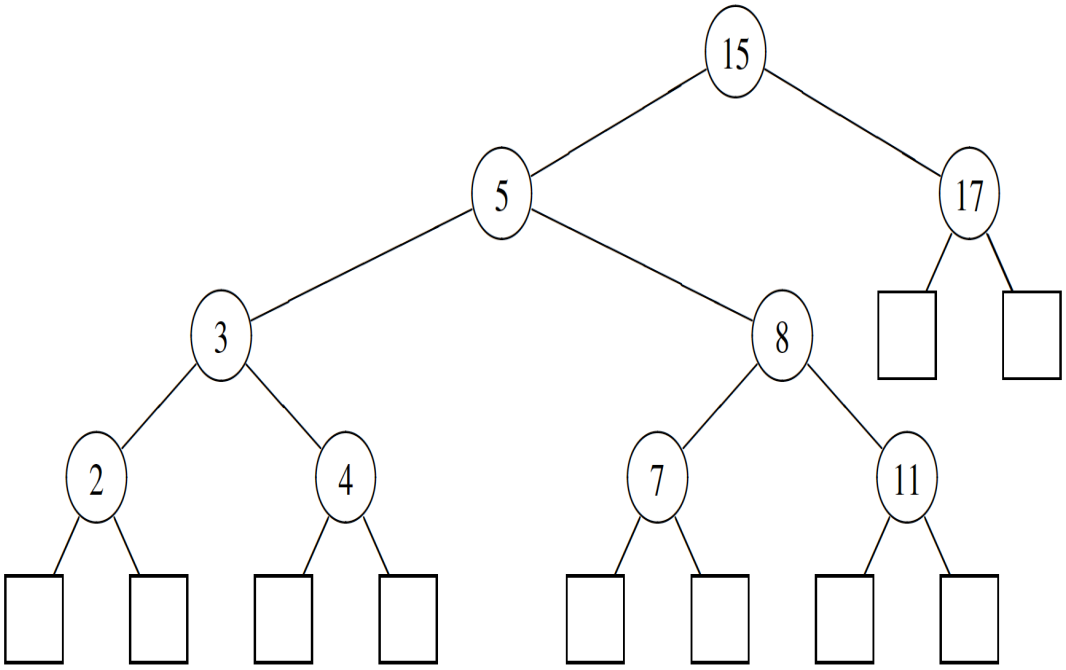


- Fall 2: p hat Vorgänger φp und Vorvorgänger $\varphi\varphi p$ und p und φp sind beides rechte oder beides linke Nachfolger

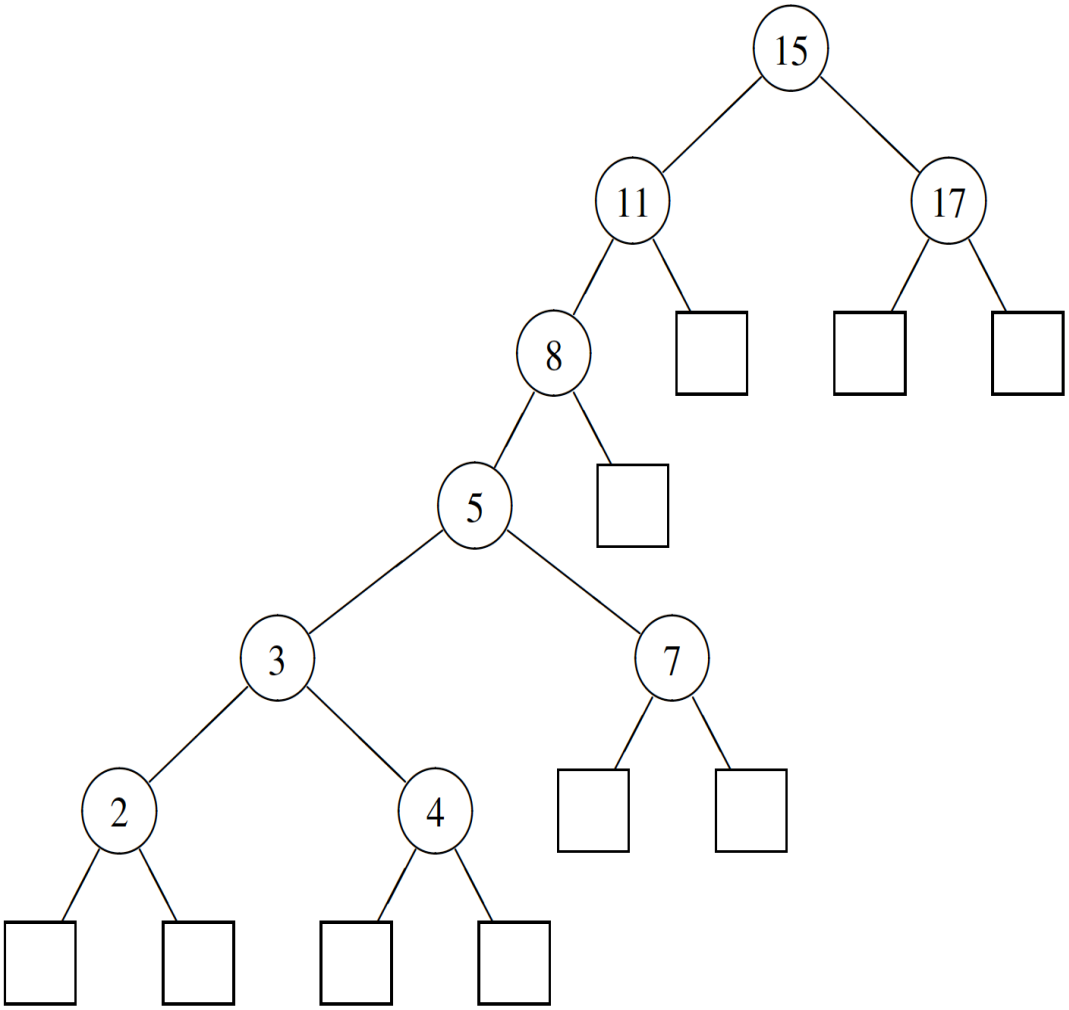
- dann führe die Operation **zig-zig** aus, d.h. zwei aufeinander folgende Rotationen in dieselbe Richtung, die p zwei Niveaus hinaufbewegen
- Fall 3: p hat Vorgänger φp und Vorvorgänger $\varphi\varphi p$ und einer der beiden Knoten p und φp ist rechter und der andere linker Nachfolger
- dann führe die Operation **zig-zag** aus, d.h. zwei aufeinander folgende Rotationen in entgegengesetzte Richtung, die p zwei Niveaus hinaufbewegen

Beispiel

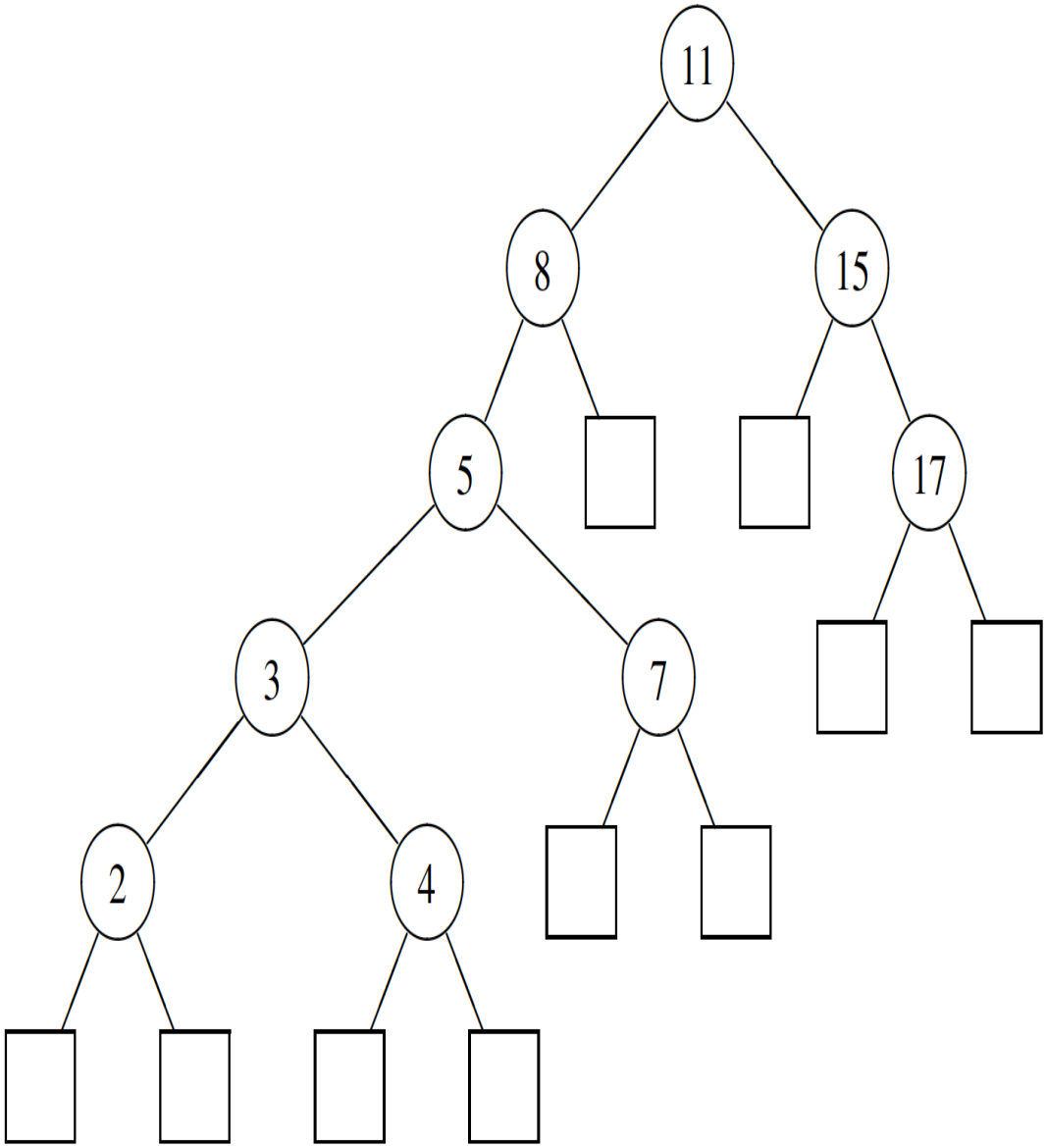
- $Splay(t, 11)$ bei folgendem Baum



- zunächst **zig-zig**



- dann **zig**



Einfügen

- zum Einfügen von x in t führe zunächst $Splay(t, x)$ aus
- falls x zum Schlüssel der Wurzel wird, kam x bereits in t vor
- falls x nicht in t vorkommt, entsteht ein Baum mit dem symmetrischen Vorgänger oder Nachfolger y von x an der Wurzel
- dann schaffe neue Wurzel mit x als Schlüssel und y als linkem bzw. rechtem Nachfolger

Entfernen

- führe zunächst wieder $Splay(t, x)$ aus
- falls x nicht Schlüssel der Wurzel ist, kam x nicht in t vor
- andernfalls ist x Schlüssel der Wurzel und Wurzel hat linken Teilbaum t_l und rechten Teilbaum t_r
  - führe dann $Splay(t_l, +\infty)$ aus, wobei $+\infty$ ein Schlüssel ist, der größer als alle Schlüssel in t_l ist
  - dabei entsteht ein Baum t_l' mit dem größten Schlüssel von t_l an der Wurzel und einem leeren rechten Teilbaum
  - ersetze diesen leeren Teilbaum durch t_r

Literaturhinweis

Die Graphen in diesem Kapitel sind aus dem Buch *Algorithmen und Datenstrukturen* von Ottmann & Widmayer.