Algorithmen und Datenstrukturen I

Bruder-Bäume

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 01.12.2019 19:06

Inhaltsverzeichnis

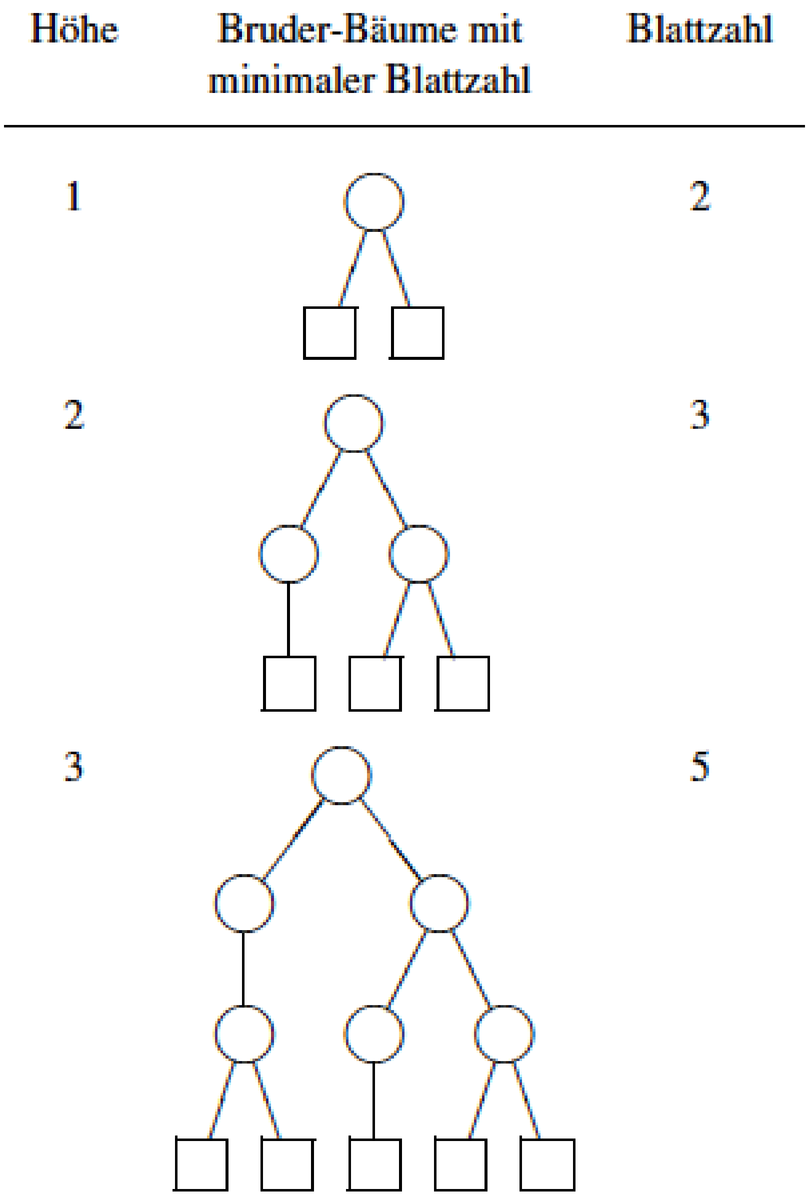
Definition

- ein binärer Baum heißt ein **Bruder-Baum**, wenn
  - jeder innere Knoten ein oder zwei Nachfolger hat,
  - jeder unäre Knoten einen binären Bruder hat und
  - alle Blätter dieselbe Tiefe haben
- **Brüder** sind Knoten die den gleichen Vorgänger haben
- aus der Definition folgt direkt
  - ist ein Knoten p einziger Nachfolger seines Vorgängers, so ist

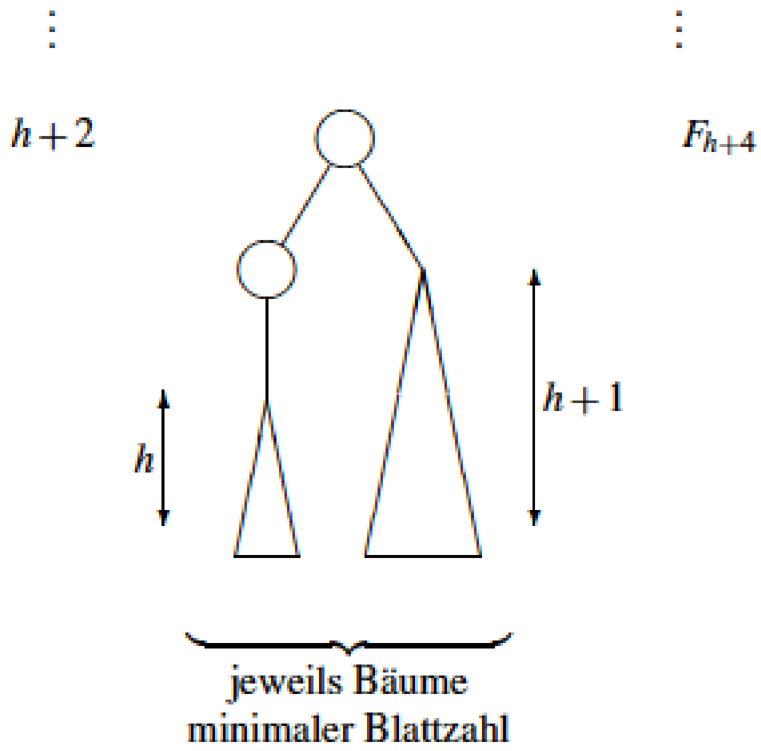
        * p ein Blatt oder
        * p ein binärer Knoten
    - von zwei Nachfolgern eines binären Knotens kann höchstens einer unär sein

- Anzahl der Blätter ist stets um 1 größer als die Anzahl der binären (inneren) Knoten

Höhe und Anzahl Blätter



- ein Bruderbaum mit Höhe h hat wenigstens F_{h+2} Blätter (F_i: i-te Fibonacci-Zahl)
- Bruderbaum mit N Blättern hat Höhe $h \leq log_2 N$



Schlüssel im Bruderbaum

- Blattsuchbaum
  - Wegweiser reichen in binären Knoten
- Suchbaum
  - Schlüssel nur in den binären Knoten
  - wir betrachten nur noch diese Variante
  - *1-2-Bruderbäume*
    - jeder innere Knoten hat mindestens 1, höchstens 2 Nachfolger

Mögliche Implementierung

```
class BrotherTree {
    struct Node {};
    struct UnaryNode : public Node {
        Node *succ = nullptr;
    };
    struct BinaryNode : public Node {
```

```
        const int key;
        Node *left = nullptr;
        Node *right = nullptr;
    };
    Node *root = new UnaryNode();
}
```

- für Einfügen und Löschen praktisch ist noch ein Zeiger auf den Vorgänger

Einfügen eines neuen Schlüssels x

- zunächst wieder Suche
- wenn Suche erfolglos an einem Blatt endet, sei p der Vorgänger dieses Blattes
- Fall 1: p hat nur einen Nachfolger
  - ersetzen des unären Knoten p durch einen binären Knoten mit dem Schlüssel x und zwei Blättern als Nachfolger
  - fertig, da Bruderbaum-Eigenschaft erhalten bleibt
- Fall 2: p ist binärer Knoten
  - ohne Einschränkung gilt: $x < p.key$
    - sonst vertausche x und $p.key$
  - x kann nicht mehr im Knoten p untergebracht werden
  - Idee: Um Platz für x zu schaffen, versuche x oder einen anderen Schlüssel beim Bruder oder beim Vorgänger unterzubringen
  - ist das erfolglos, verschiebt man das Einfügeproblem um ein Niveau nach oben bis zur Wurzel
  - wenn das Problem bei der Wurzel nicht gelöst werden kann, wird der Baum durch schaffen einer neuen Wurzel aufgestockt
    - ein Bruderbaum wächst also an der Wurzel!

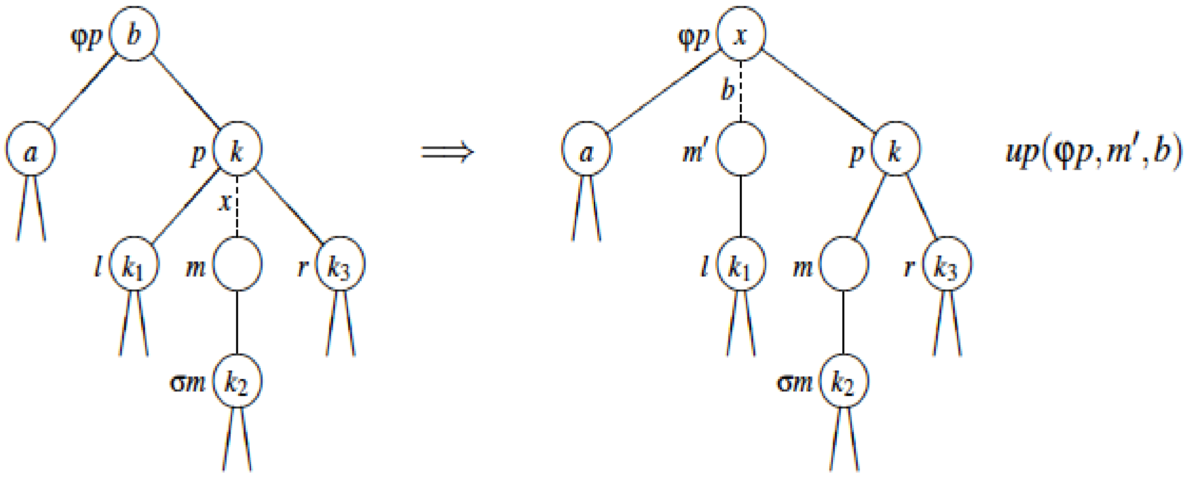
$up(p, m, x)$ — Invariante

- Prozedur *up* zum Platz suchen
- es gilt vor jedem Aufruf von $up(p, m, x)$ folgende Invariante:
  1. p hat zwei Nachfolger p_l und p_r, die beide Wurzeln von 1-2-Bruderbäumen sind

2. m ist entweder Blatt oder unärer Knoten dessen Nachfolger Wurzel eines 1-2-Bruderbaumes ist
3. Schlüssel in $p_l <$ x $<$ Schüssel in $m < p.key <$ Schlüssel in p_r

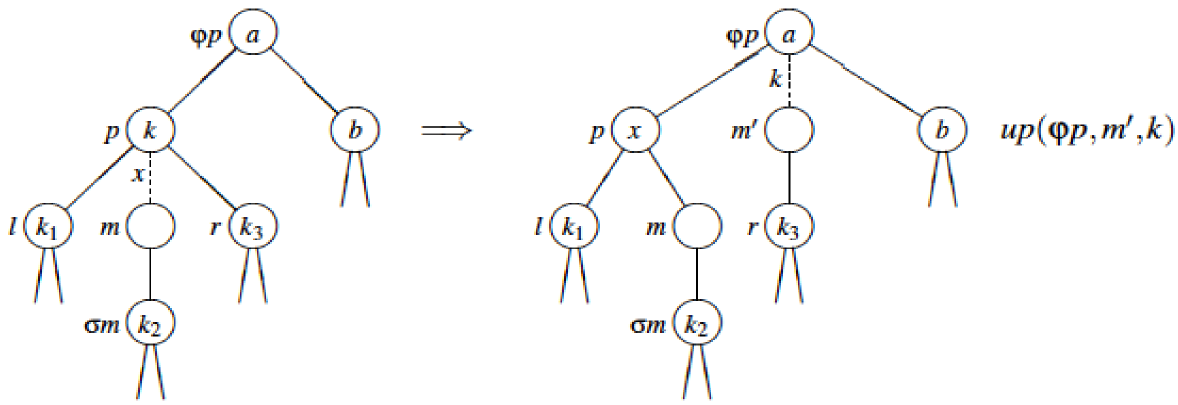
$up(p, m, x)$

- Fall 1: p hat linken (binären) Bruder mit zwei Nachfolgern

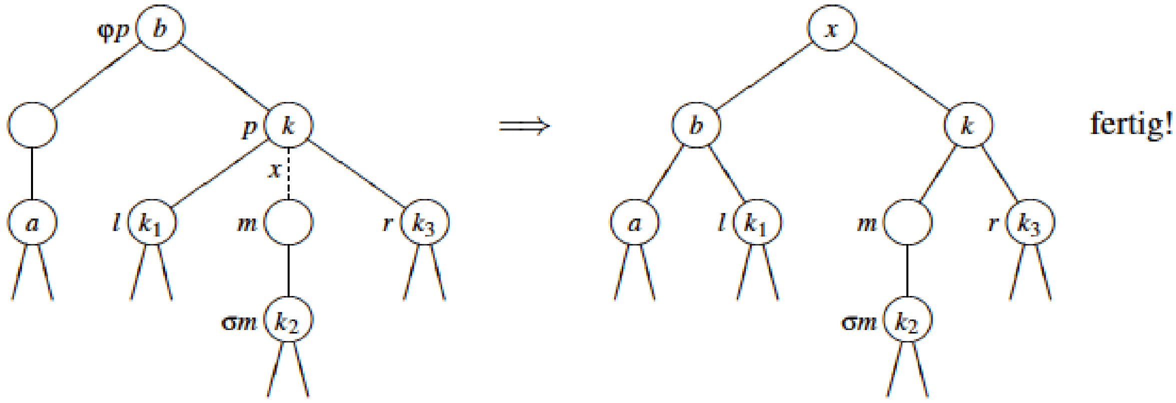


- beim ersten Aufruf von up sind l, m und r Blätter
- d.h. σm, k_1, k_2 und k_3 existieren nicht
- Analoges gilt für die folgenden Figuren

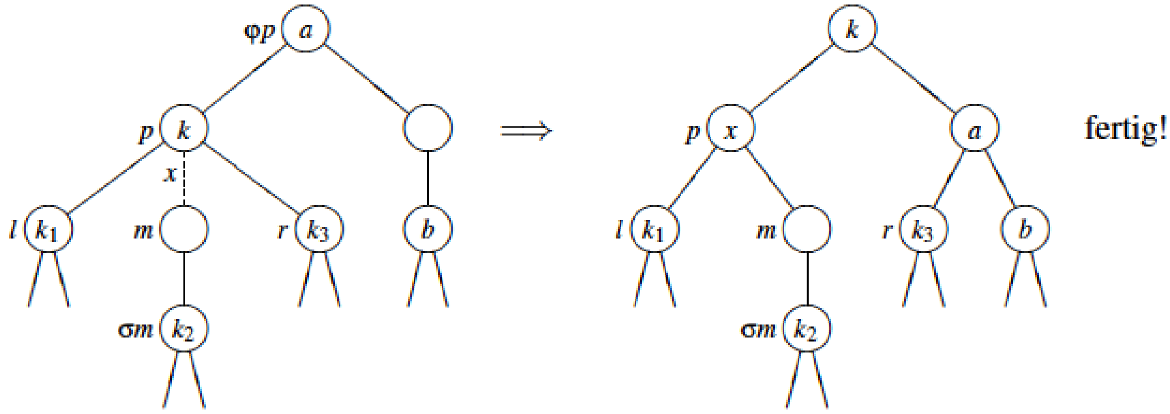
- Fall 2: p hat einen rechten (binären) Bruder mit zwei Nachfolgern



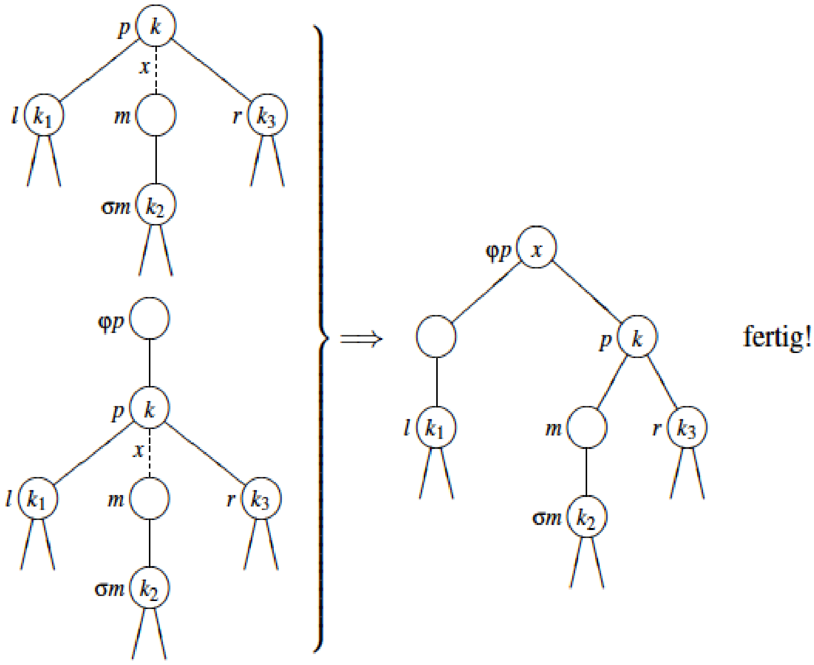
- Fall 3: p hat einen unären linken Bruder



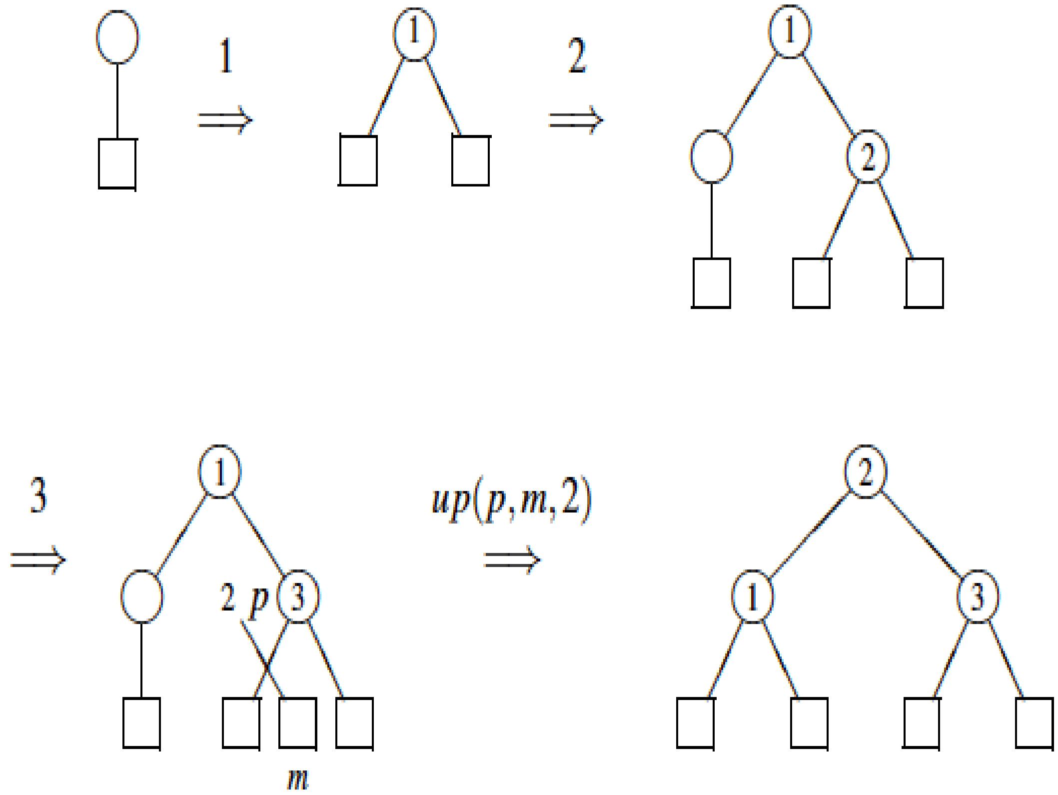
- Fall 4: p hat einen unären rechten Bruder

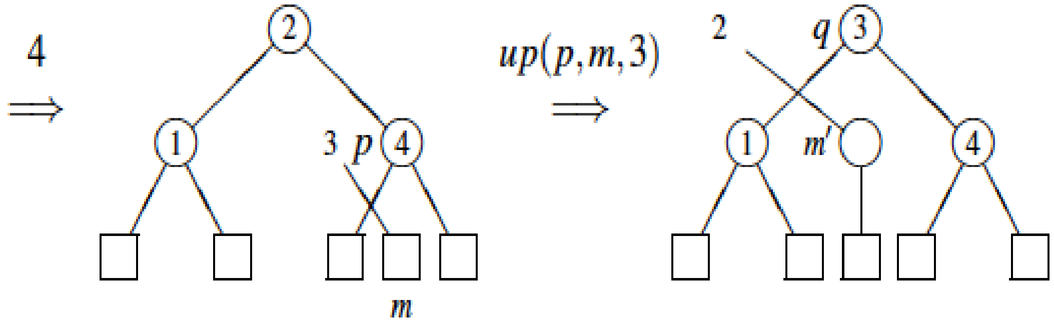


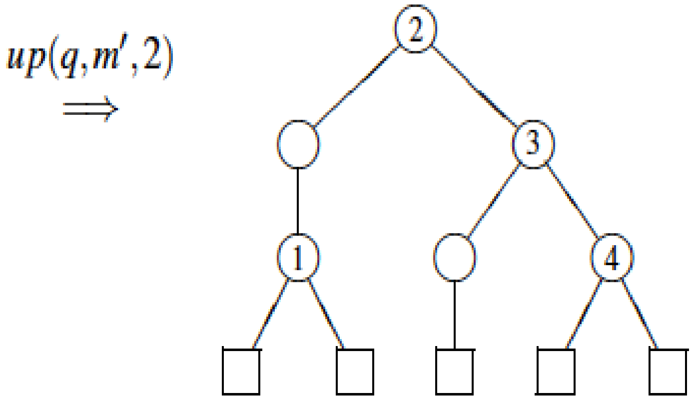
- Fall 5: p hat keinen Bruder (Wurzel oder Nachfolger von unärem Knoten)

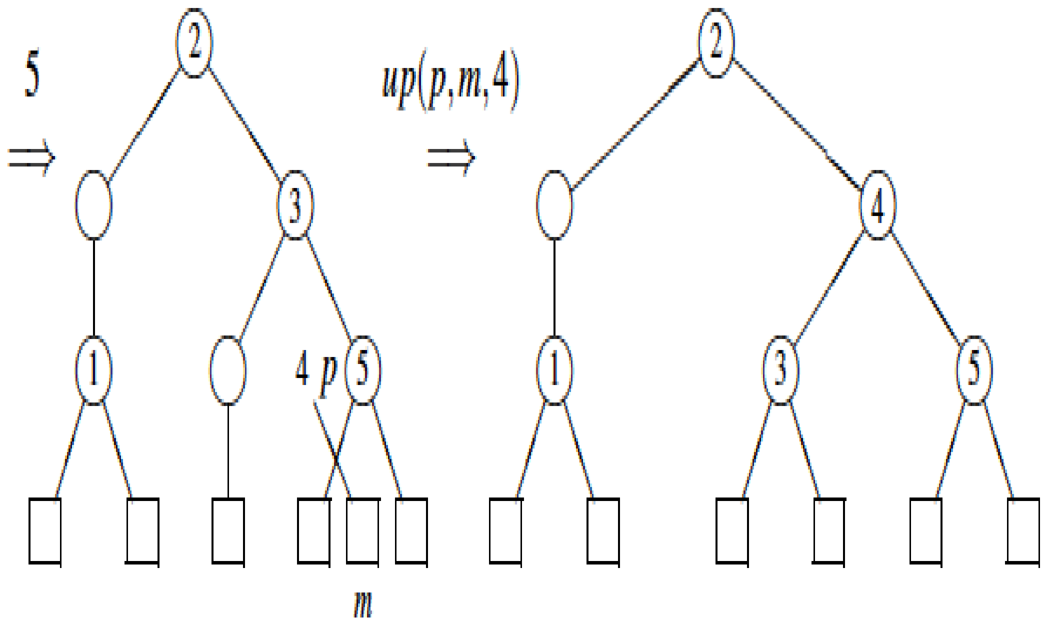


Beispiel: Einfügen 1, 2, 3, 4, 5 in leeren Baum



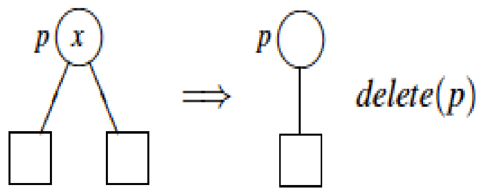

4
2
1
3 p
4
m
up(p,m,3)
2
q
3
1
m′
4


up(q,m′,2)
2
3
1
4

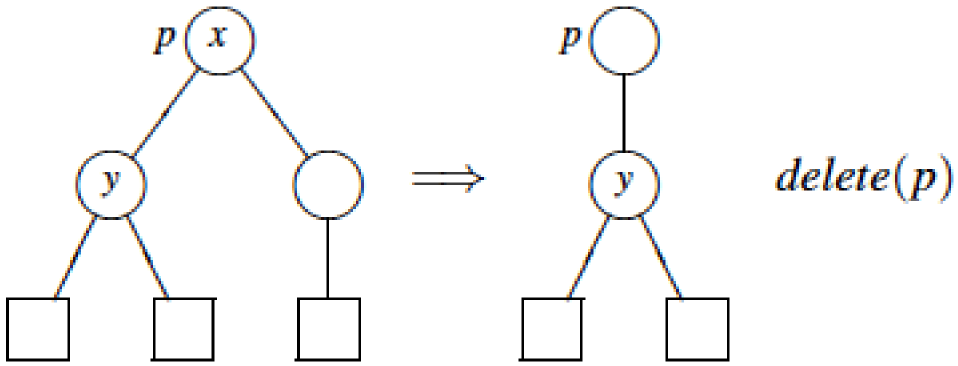

5
2
3
1
4 p
5
m
up(p,m,4)
2
4
1
3
5

Entfernen eines Schlüssels

- zunächst wieder Suche nach Schlüssel
- wenn nicht gefunden $\Rightarrow$ fertig
- sonst unter Umständen das Entfernen von p mit $p.key = x$ zurückführen auf Entfernen des symmetrischen Nachfolgers (Vorgängers)
- daraus ergeben sich ohne Einschränkung die folgenden Fälle:
- Fall 1: die Nachfolger von p sind Blätter



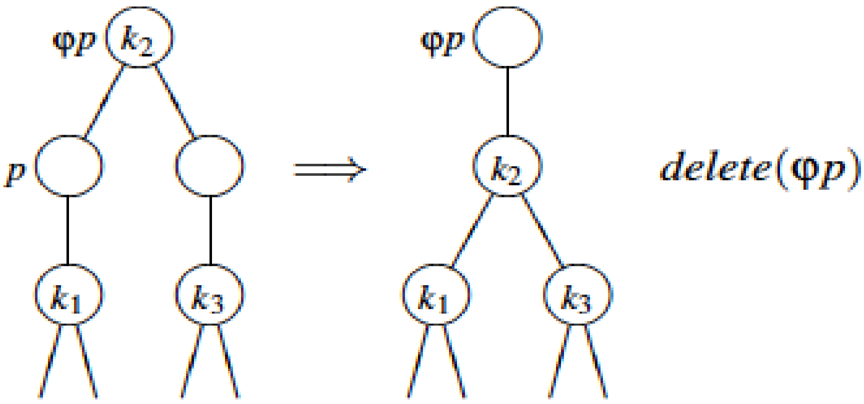
- p wird zu unärem Knoten, d.h. Schlüssel x wird entfernt
- anschließend Aufruf von $delete(p)$ um die eventuell verletzte 1-2-Bruderbaum-Eigenschaft wieder herzustellen
- Fall 2: p hat einen unären Nachfolger



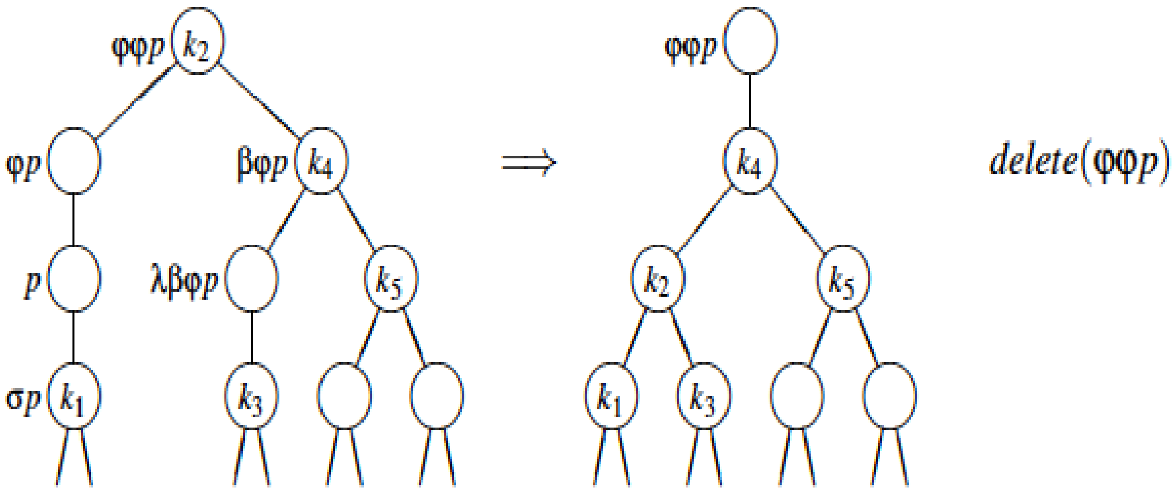
- Fall mit 2 binären Nachfolgern nicht möglich, da sonst symmetrische Nachfolger (Vorgänger) entfernt wird

$delete(p)$

- Fall 1: p hat binären Bruder $\Rightarrow$ fertig
- Fall 2: p hat unären Bruder

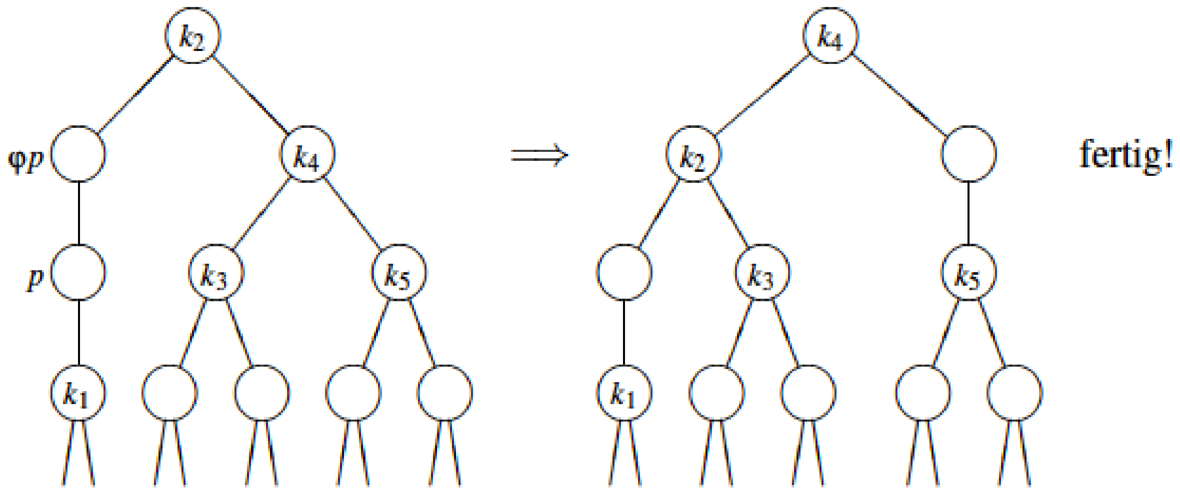


- p ist rechter Nachfolger: analog
- Fall 3: p hat keinen Bruder
- Fall 3.1: p ist Wurzel
  - entferne p und mache den einzigen Nachfolger von p zur neuen Wurzel
- Fall 3.2: p ist einziger Nachfolger seines unären Vorgängers φp
  - nach Invariante muss φp einen binären Bruder $\beta\varphi p$ haben
  - Fallunterscheidung je nachdem $\beta\varphi p$ drei oder vier Enkel hat
- Fall 3.2.1: der linke Nachfolger von $\beta\varphi p$ hat nur einen Nachfolger
  - Annahme φp ist linker Nachfolger von $\varphi\varphi p$
  - andere Fälle symmetrisch



- Fall 3.2.2: beide Nachfolger von $\beta\varphi p$ haben zwei Nachfolger
  - Annahme φp ist linker Nachfolger von $\varphi\varphi p$

- anderer Fall symmetrisch



Literaturhinweis

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