

2/3/4 Bäume Selbstaushleichendes Einfügen (1)

„you've got to stand for something or you'll fall for anything“

transformieren Höhe++
Neue Kapazität
füllen integrieren

2/4/2003 218

2/3/4 Bäume Selbstaushleichendes Einfügen (2)

„you've got to stand for something or you'll fall for anything“

Höhe++

2/4/2003 219

2/3/4 Bäume Selbstaushleichendes Einfügen (3)

- Optimistische Strategie**
 Starte bei der Wurzel;
 Traversiere den Baum ordnungsorientiert bis Blattebene;
 Falls Blattknoten voll (d. h. 4er Knoten), löse ihn auf,
 mit Backtracking im Falle eines 4er Vorgängerknotens
 Füge neues Element im Blattknoten ein
- Pessimistische Strategie**
 Starte bei der Wurzel;
 Traversiere den Baum ordnungsorientiert bis Blattebene,
 löse dabei all 4er Knoten präventiv auf;
 Füge neues Element im Blattknoten ein

2/4/2003 220

2/3/4 Bäume

Selbstaussgleichendes Einfügen (4)

○ Fallanalyse zur Auflösung von Viererknoten

- **Kein Vorgängerknoten**

2/4/2003 221

2/3/4 Bäume

Selbstaussgleichendes Einfügen (5)

- **2er Knoten als Vorgänger**

2/4/2003 222

2/3/4 Bäume

Selbstaussgleichendes Einfügen (6)

- **3er Knoten als Vorgänger**

symmetrisch analog

2/4/2003 223

2/3/4 Bäume

Selbstaussgleichendes Einfügen (7)

- 4er Knoten als Vorgänger

symmetrisch analog

Rückführung auf 2er Fall

Rückführung auf 2er Fall

2/4/2003 224

2/3/4 Bäume

Darstellung als Rot/Schwarz-Bäume (1)

Binär-

2er Knoten

3er Knoten

4er Knoten

rotR(b, a)
oder
rotL(a, b)

„Interne“
Kanten
rot

2/4/2003 225

2/3/4 Bäume

Darstellung als Rot/Schwarz Bäume (2)

- Fallanalyse des selbstaussgleichenden Einfügens in Rot/Schwarzdarstellung
- Kein Vorgängerknoten

2/4/2003 226

2/3/4 Bäume

Darstellung als Rot/Schwarz Bäume (3)

○ 2er Knoten als Vorgänger

The diagram illustrates a transformation of a 2/3/4 tree into a red-black tree. On the left, a 2/3/4 tree has root 'a' (blue) with children 'b' (blue) and 'c' (blue). Node 'b' has children 'a' (green) and 'c' (green). Node 'c' has children 'b' (blue) and 'd' (blue). On the right, the resulting red-black tree has root 'd' (green) with children 'b' (green) and 'c' (green). Node 'b' has children 'a' (green) and 'c' (green). Node 'c' has children 'b' (blue) and 'd' (blue). Red arrows indicate the new parent-child relationships.

2/4/2003 227

2/3/4 Bäume

Darstellung als Rot/Schwarz Bäume (4)

○ 3er Knoten als Vorgänger, Fall „Aussen“

The diagram shows a 2/3/4 tree with root 'd' (green) and children 'b' (green) and 'e' (green). Node 'b' has children 'a' (green) and 'c' (green). Node 'e' has children 'd' (blue) and 'b' (blue). Node 'b' (under 'e') has children 'a' (green) and 'c' (green). A rotation $rotR(e, d)$ is indicated, leading to a tree with root 'd' (green) and children 'b' (green) and 'e' (green). Node 'b' has children 'a' (green) and 'c' (green). Node 'e' has children 'd' (blue) and 'b' (blue). Node 'b' (under 'e') has children 'a' (green) and 'c' (green). The result is labeled '???'.

2/4/2003 228

2/3/4 Bäume

Darstellung als Rot/Schwarz Bäume (5)

○ 3er Knoten als Vorgänger, Fall „Mitte“

The diagram shows a 2/3/4 tree with root 'e' (green) and children 'a' (green) and 'c' (green). Node 'a' has children 'b' (blue) and 'd' (blue). Node 'c' has children 'a' (green) and 'd' (green). Node 'a' (under 'c') has child 'b' (blue). A rotation $rotL(a, c)$ leads to a tree with root 'e' (green) and children 'c' (green) and 'a' (green). Node 'c' has children 'a' (green) and 'd' (green). Node 'a' has child 'b' (blue). A second rotation $rotR(e, c)$ leads to a tree with root 'c' (green) and children 'a' (green) and 'e' (green). Node 'a' has children 'b' (blue) and 'd' (blue). Node 'e' has children 'a' (green) and 'd' (green). Node 'a' (under 'e') has child 'b' (blue).

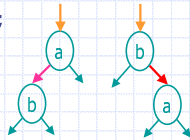
2/4/2003 229

2/3/4 Bäume

Programm Ausgleichendes Einfügen (4)

```

PROCEDURE Rotate (VAR a, b: Node): Node;
BEGIN
  IF b = a.left THEN (*rotate right*)
    a.left := b.right; b.right := a
  ELSE (*rotate left*) a.right := b.left; b.left := a
  END;
  b.red := a.red; a.red := TRUE;
  RETURN b
END Rotate;
  
```



2/4/2003

233

2/3/4 Bäume

Suchen

Algorithmus und Effizienz

- Analog „Binary Search“ bei Daten mit Direktzugriff

Suche unter n Elementen	2/3/4 Darstellung	R/S Darstellung
Algorithmus	Mehrweg Baumsuche	Gewöhnliche Binärbaumsuche
Pfadlänge minimal	$4 \log_3(n)$	$2 \log_2(n)$
Pfadlänge maximal	$2 \log_3(n)$	$2 * 2 \log_2(n)$

2/4/2003

234
