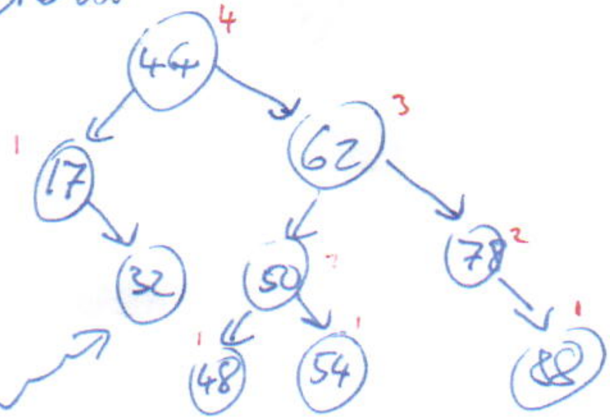
18.01.2012

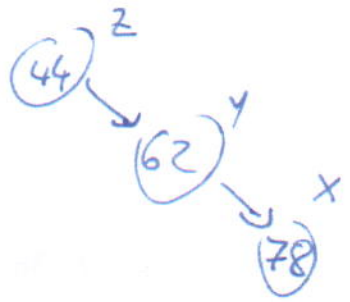
~~Einfügen:~~ ~~Entfernen:~~

Löschen:

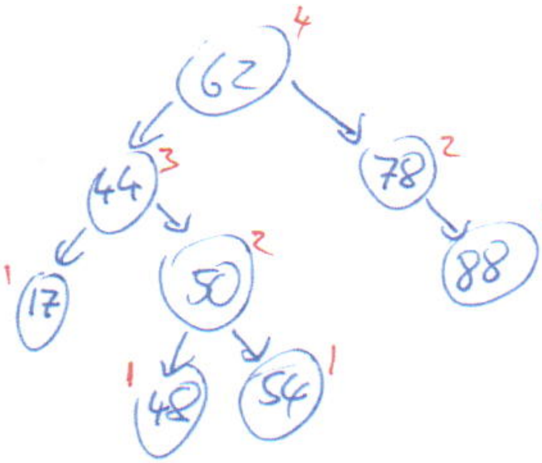


Löschen macht Baum unbalanciert!

Neu:



also



→ höhenbalanciert!

Ähnliche Idee:

~~Entfernen~~ Lösche v

Sei z der erste ~~unbalancierte~~ Knoten auf dem Weg von v zur Wurzel, der nach Entfernung von v unbalanciert wird.

Sei y das Kind von z mit größerer Höhe (offenbar kein Vorfahre von v).

Sei x ein Kind von y mit größter Höhe. (Ggf. Unentschieden, dann Wahl egal!)

Dann Restructure(x)

Problem: Höhe des Teilbaums von b kann sich reduzieren → Vorfahre von b kann unbalanciert werden.

Idee: Immer wieder Restructure auf dem Weg zur Wurzel, bis nicht mehr notwendig!

Satz 4.4

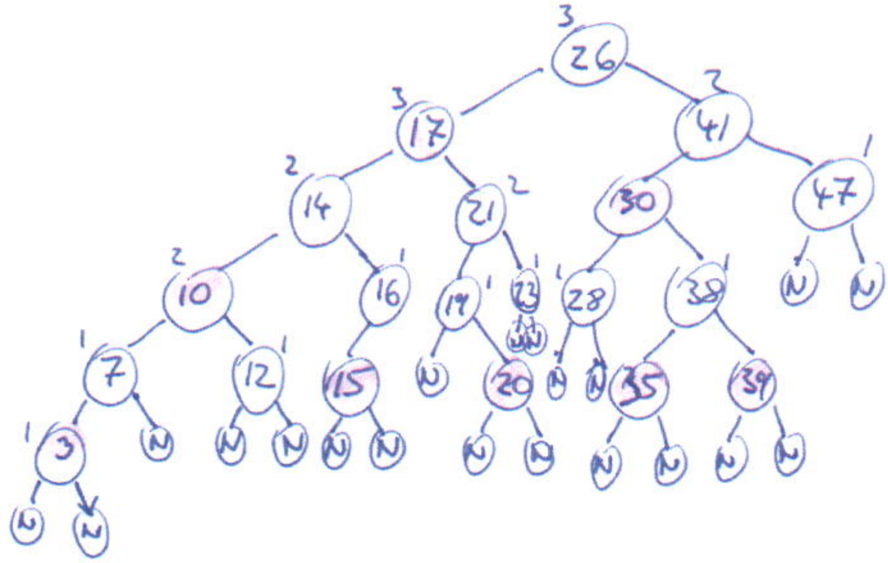
Löschen in einem AVL-Baum lässt sich (unter Bewahrung der Höhenbalanciertheit in $O(\log n)$ ausführen).

4.6 Andere Baumstrukturen

4.6.1 Rot-Schwarz-Bäume

Eigenschaften

0. Binärer Suchbaum, alle Blätter sind „NIL"



Kein AVL-Baum!

1. Jeder Knoten ist entweder rot oder schwarz
2. Die Wurzel ist schwarz.
3. Jedes Blatt (NIL) ist schwarz.
4. Wenn ein Knoten rot ist, dann sind seine beiden Kinder schwarz
5. Für jeden Knoten enthalten alle Pfade, die an einem Knoten starten und in einem Blatt des Teilbaumes dieses Knotens enden, die gleiche Anzahl schwarzer Knoten.

Man kann folgende Dinge beweisen:

Satz 4.12

Ein Rot-Schwarz-Baum mit n inneren Knoten hat höchstens die Höhe $2 \log(n+1)$.

Beweis: Nicht hier!

Satz 4.13

Löschen und Einfügen (samt notwendiger Reparaturoperationen zum Erhalt der Eigenschaften 0.–5.) lassen sich in einem Rot-Schwarz-Baum in $O(\log n)$ durchführen, wobei n die Zahl der inneren Knoten ist

Historisch:

Bayer (1972) – Grundidee

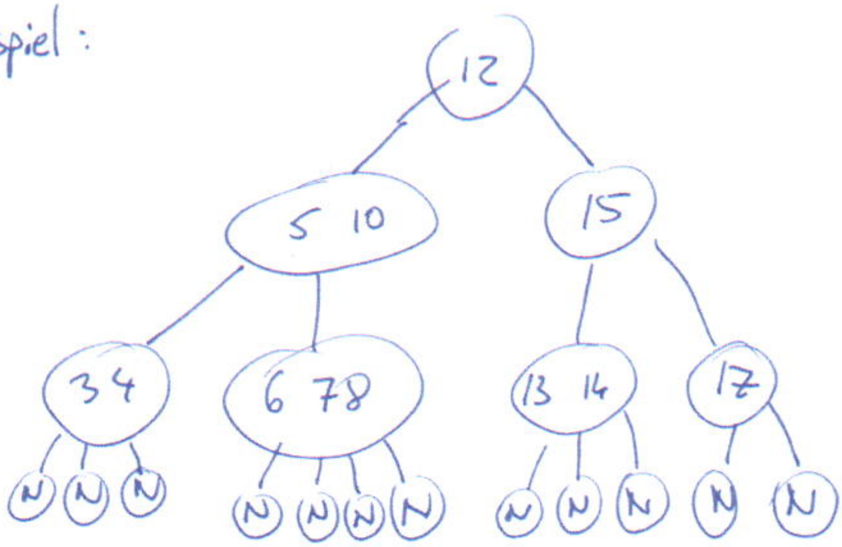
Guibas / Sedgwick (1978) – Farben + Eigenschaften

4.6.2 (2,4)-Bäume

Grundideen: (I) Erlaube mehr als ein Objekt pro Knoten (z.B. zwei oder drei)

(II) Erlaube mehr als zwei Kinder (z.B. bis zu vier)

Beispiel:



Eigenschaften:

(i) Jeder Knoten hat höchstens 3 Objekte,

(ii) Jeder Knoten hat höchstens 4 Kinder

Sowie

(iii) Jeder innere Knoten hat mindestens zwei Kinder

(iv) Alle Blätter haben dieselbe Tiefe

Satz 4.14

Die Höhe eines (2,4)-Baumes für n Objekte ist $\Theta(\log n)$.

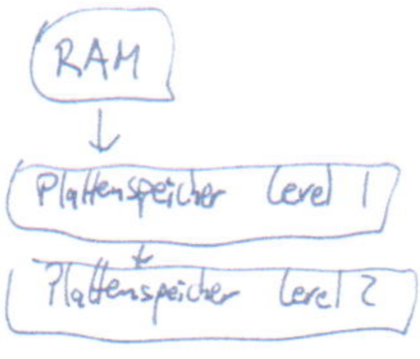
Satz 4.15

Einfügen und Löschen in einen (2,4)-Baum für n Objekte (samt Reperatur) ist in $O(\log n)$ möglich.

Beweise: Nicht hier!

46.3 B-Bäume

Idee: Speicherhierarchien! ("Cache")



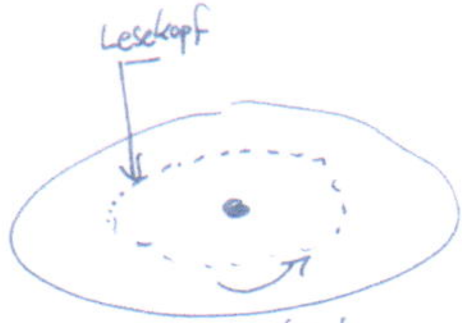
Geschwindigkeit des Zugriffs:

RAM : schnell

Platte : Fixkosten (richtige Stelle auf Platte finden),

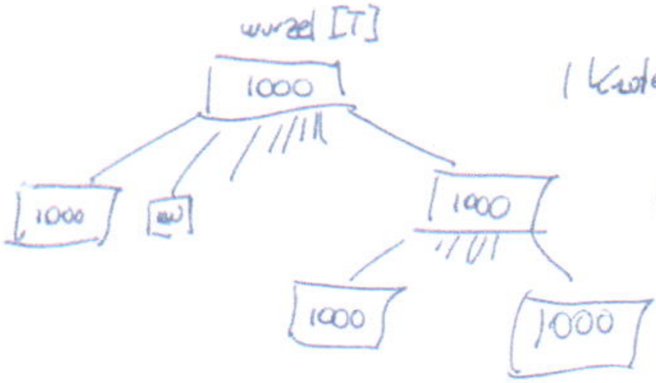
dann moderate Zugriffskosten pro Block.
Speicherzellen

→ Idee: Bei Zugriff gleich mehr Daten „schaufeln"!



rotiert - eventuell ganzer Umlauf erforderlich!

Konsequent umgesetzt:



1 Knoten, 1000 Objekte

1001 Knoten, 1001000 Objekte

1002001 Knoten, 1002001000 Objekte

⇒ B-Bäume!

Mehr nicht hier, siehe Literaturhinweise

Historisch + Literatur:

Knuth 1973

Aho, Hopcroft, Ullman 1974

Sedgwick 1988

Aktuelle Forschung:

Speichergröße nicht immer bekannt, man hat nicht mehr nur starre Speicherhierarchien.

Bender, Demaine, Farach-Colton (2000):
(* 1970) (* 1981) (* 1964)

„Cache-oblivious B-trees"

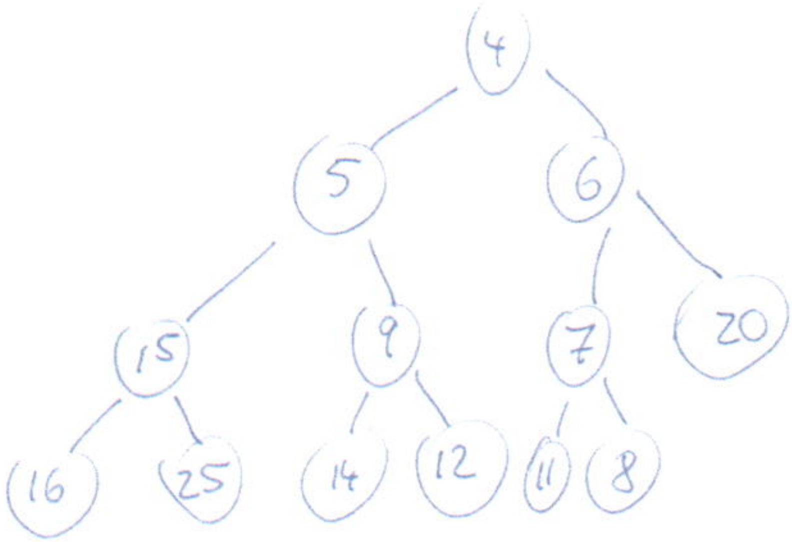
Startup-Firma „Tokutek"

"Tokutek was founded to dramatically enhance the performance of databases and file systems. Tokutek's software is based on eight years of research in cache-oblivious algorithmics and will dramatically accelerate key database and file system operations."

↳ Links!

4.6.4 Heaps („Haufen")

Binärbaum, aber mit anderen Ordnungsprinzip:



Der ~~Objekt~~ Wert in jedem Knoten ist ~~größer~~ kleiner als in jeden Kind!

↪ Verschiedene Operationen, nicht hier!