

Informatik Seminar 2008 Iwanowski

Algorithmen - Thema 8:

Zeit vs. Platz III

B-Bäume

Sebastian Barzyk WI2393

B-Baum Geschichte

- Entwicklung: 1972 - Bayer und McCreight
- Einsatz: Erstes relationales DBMMS
 (B-Bäume als Indexverfahren)
- Name: Unklar! Evtl. Balanced oder Bayer

B-Baum Motivation

Datenstrukturen liegen auf einer Festplatte (z.B. Datenbank):

=> Flaschenhals:

- Schreib-/Lesekopf bewegen

=> Hauptziel bei der Organisation des Datenzugriffs:

- Schreib-Lesekopf-Bewegungen minimieren

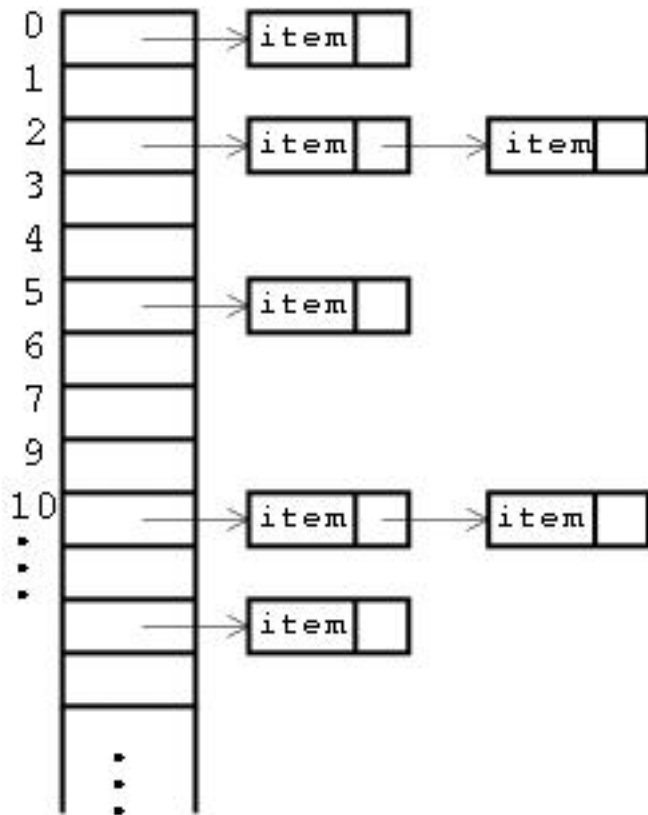
Platz vs. Zeit Tausch besonders lohnenswert!

B-Baum Motivation

Gesucht: Datenstruktur zur Indizierung von Datensätzen einer Datenbank.

B-Baum Motivation

Hash-Table?



Vorteile:

- Best-Case: **$O(1)$**

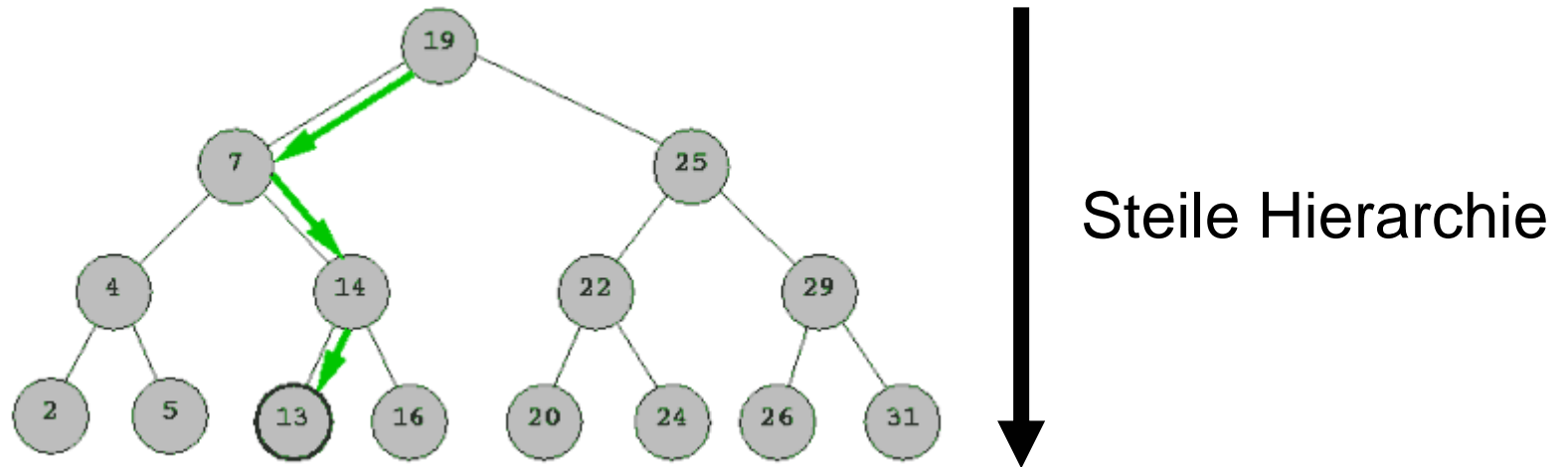
Nachteile:

- Bei stetig wachsender DB-
Größe sind Entartungen kaum
vermeidbar:

Worst-Case \Rightarrow **$O(n)$**

B-Baum Motivation

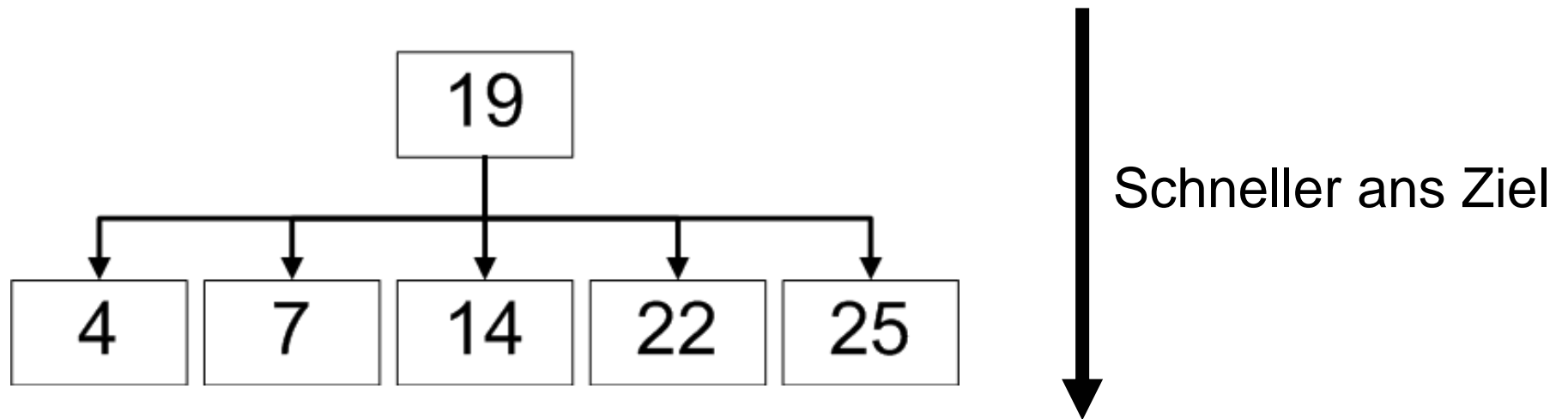
Binärer Suchbaum?



Vorteil: Wenn balanciert (Rot Schwarz etc.)
=> $O(\log n)$

B-Baum Motivation

N-Baum?

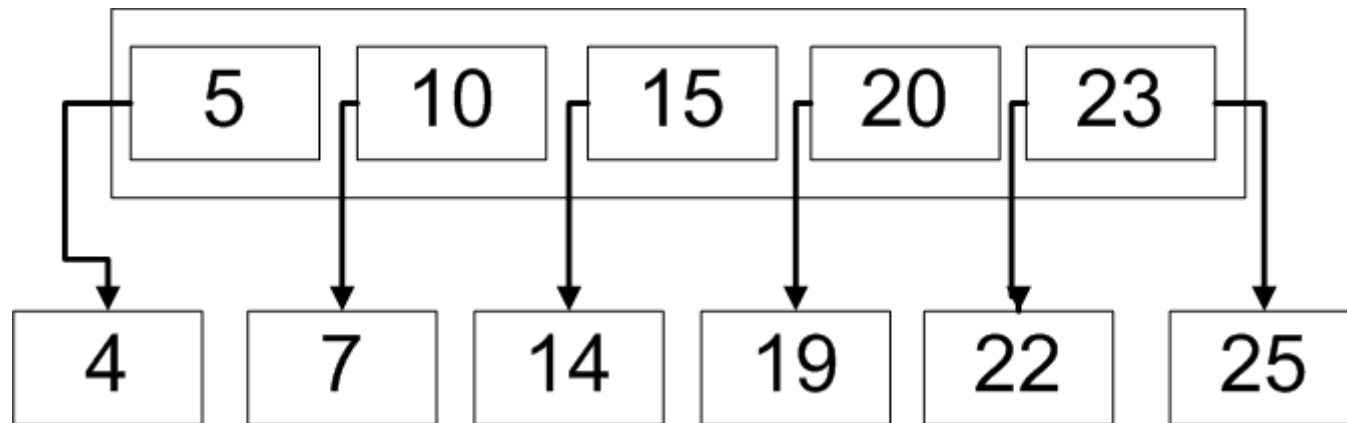


Moment! Das ist kein Suchbaum!

Wie soll der Sub-Baum ausgewählt werden?

B-Baum Motivation

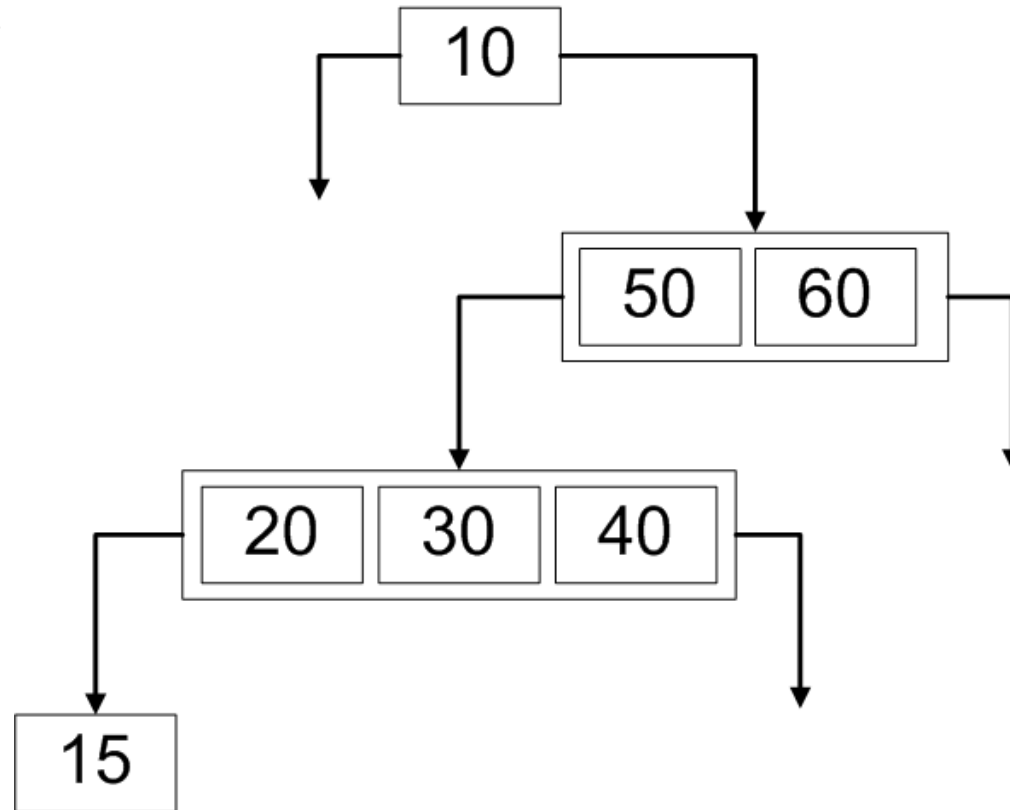
N-Such-Baum:



Wie viele Schlüssel werden benötigt?

Entartung

Balance?



Abwegen: Ab wann soll balanciert werden?

Entartung

Ab wann soll balanciert werden?

B-Baum: Jeder Knoten soll mindestens halb-voll sein.

B*-Baum: Jeder Knoten soll mindestens zu $2/3$ gefüllt sein.

Ordnung

Ordnung des Baumes

Ordnung = t = Maximale Anzahl an Kindern.

Maximale Anzahl an Schlüsseln
entsprechend: $t-1$

Ordnung

Ordnung des Baumes

Hohe Ordnung

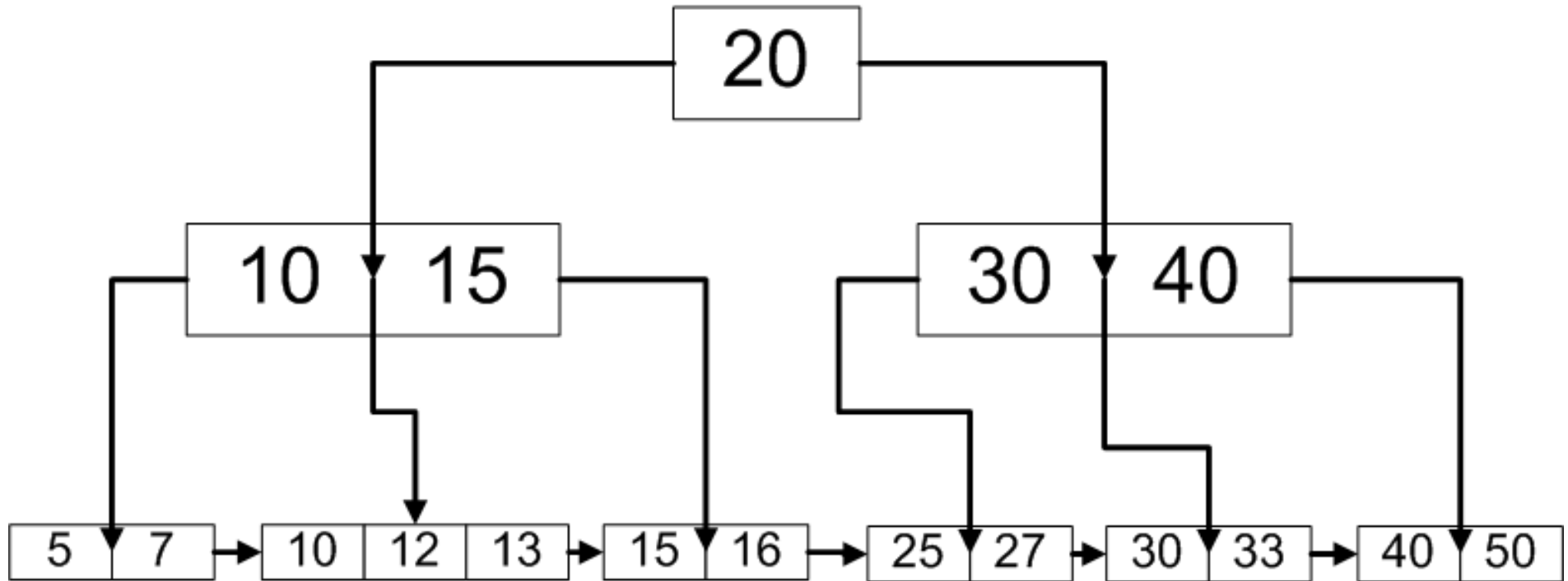
→ Flache Hierarchie

→ Wenig Festplattenzugriffe

=> Ordnung so hoch wie möglich?

B-Baum

B+ Baum:



Definitionen

B-Baum mit der Ordnung t :

- **1.) Jeder Knoten hat $\leq t$ Kinder.**
- **2.) Alle Nicht-Wurzel-Knoten haben $\geq t/2$ Kinder.**
- **3.) Alle Blätter liegen auf derselben Ebene.**
- **4.) Anzahl der Schlüssel eines Knotens ist Anzahl der Kinder des Knotens $- 1$.**
- **6.) Alle Schlüssel-Werte im linken Teilbaum sind kleiner, alle Schlüssel-Werte im rechten Teilbaum sind größer oder gleichgroß.**

Varianten

B+ Baum:

- **B-Baum der lediglich an den Blättern Nutz-Informationen trägt.**
- **(Blätter verkettet als linked List)**

B* Baum:

- **B+ Baum dessen Nicht-Wurzel-Knoten mindestens zu 2/3 gefüllt sein müssen.**

Komplexität

$O(\log n)$!

Proportional zur Höhe des Baumes.

Insert

Bin ich in einem Blattknoten?

Ja:

1. Objekt sortiert einfügen.
 2. Blatt-Größe ok? => FERTIG!
- Sonst:
Hat dieses Blatt Eltern?

Ja:

FERTIG!

(Eltern haften
Für Ihre Kinder)

Nein:

| SPLIT |

Nein:

1. Füge in den passenden Sub-Baum ein.
2. Ist der Sub-Knoten in den eingefügt wurde nun zu groß?

Ja:

| SPLIT |

Nein:

FERTIG!

Delete

Bei inneren Knoten:

1. Wenn Lösch-Key in Key-Liste: Ersetze ihn mit dem nächstem Key in seinem Blatt.
2. Delegiere das löschen an den geeigneten Sub-Baum.
3. Wenn Sub-Baum zu klein ist -> | Balanciere |

Bei Blättern:

1. Lösche Element in der Liste

Delete

| Balanciere |

Einer unserer Sub-Bäume hat zu wenig Elemente!

- Hat das rechte Geschwisterchen Elemente zu verschenken?

Ja: 1.1 Seperator löschen

1.2 Seperator einfügen

FERTIG

2. Hat das linke Geschwisterchen Elemente zu verschenken?

Ja: 2.1 Seperator an den Anfang des fehlerhaften Knotens

2.2 Ersetze Seperator in meiner Key-Liste mit dem Letzten Element vom linken Geschwisterchen.

2.3 Letztes Kind des linken Geschwisterchens als erstes Kind im fehlerhaften Knoten einfügen.

Delete

| Balanciere |

Keines meiner Geschwister hat was zu verschenken! ☹️☹️☹️

Merge:

- Lösche Seperator-Key (Key zwischen Problemknoten und Geschwisterchen)

2. Problem-Knoten in den Geschwister-Knoten mergen.

FERTIG! ... Wenn ich jetzt zuwenig Elemente habe regeln das meine Eltern schon für mich 😊

**Danke für Ihre
Aufmerksamkeit**