ADS: Algorithmen und Datenstrukturen

Teil X

Peter F. Stadler & Konstantin Klemm

Bioinformatics Group, Dept. of Computer Science & Interdisciplinary Center for Bioinformatics, **University of Leipzig**

13. Januar 2010

Suchverfahren für große Datenmengen

- Bisher betrachtete Datenstrukturen (Arrays, Listen, Binärbäume) und Algorithmen waren auf im Hauptspeicher vorliegende Daten ausgerichtet
- effiziente Suchverfahren für große Datenmengen auf Externspeicher erforderlich (persistente Speicherung)
- große Datenmengen können nicht vollständig in Hauptspeicher-Datenstrukturen abgebildet werden
- Zugriffsgranulat sind Seiten bzw. Blöcke von Magnetplatten: z.B. 4-16 KB
- Zugriffskosten: Faktor 100000 langsamer als für Hauptspeicher (5 ms verglichen mit 50 ns)

Sequentieller Dateizugriff

Sequentielle Dateiorganisation

- Datei besteht aus Folge gleichartiger Datensätze
- Datensätze sind auf Seiten/Blöcken gespeichert
- ggf. bestimmte Sortierreihenfolge (bzgl. eines Schlüssels) bei der Speicherung der Sätze (sortiert-sequenzielle Speicherung)

Sequenzieller Zugriff

- Lesen aller Seiten / Sätze vom Beginn der Datei an
- sehr hohe Zugriffskosten, wenn nur ein Satz benötigt wird

Optimierungsmöglichkeiten

- "dichtes Packen" der Sätze innerhalb der Seiten
- Clusterung zwischen Seiten, d.h. "dichtes Packen" der Seiten einer Datei auf physisch benachbarte Plattenbereiche, um geringe Zugriffszeiten zu ermöglichen

Schneller Zugriff auf einzelne Datensätze erfordert Einsatz von zusätzlichen Indexstrukturen, z.B. Mehrwegbäume

Alternative: gestreute Speicherung der Sätze ($\rightarrow$ Hashing)

Mehrwegbäume

Ausgangspunkt: Binäre Suchbäume (balanciert)

- entwickelt für Hauptspeicher
- ungeeignet für große Datenmengen

Externspeicherzugriffe erfolgen auf Seiten

- Abbildung von Schlüsselwerten/Sätzen auf Seiten
- Index-Datenstruktur für schnelle Suche

Alternativen:

- m-Wege-Suchbäume
- B-Bäume
- B*-Bäume

Grundoperationen: Suchen, Einfügen, Löschen

m-Wege-Suchbäume I

Definition : Ein m-Wege-Suchbaum oder ein m-ärer Suchbaum B ist ein Baum, in dem alle Knoten einen Grad $\leq m$ besitzen.
Entweder ist B leer oder B hat folgende Eigenschaften:

1. Jeder Knoten des Baums mit b Einträgen, $b \leq m-1$, hat folgende Struktur: Die P_i, $0 \leq i \leq b$, sind Zeiger auf die Unterbäume des Knotens und die K_i und D_i, $1 \leq i \leq b$ sind Schlüsselwerte und Daten.
2. Die Schlüsselwerte im Knoten sind aufsteigend geordnet: $K_i \leq K_{i+1}$, $1 \leq i < b$.
3. Alle Schlüsselwerte im Unterbaum von P_i sind kleiner als der Schlüsselwert K_{i+1}, $0 \leq i < b$.
4. Alle Schlüsselwerte im Unterbaum von P_b sind größer als der Schlüsselwert K_b.
5. Die Unterbäume von P_i, $0 \leq i \leq b$ sind auch m-Wege-Suchbäume.

Die D_i können Daten oder Zeiger auf die Daten repräsentieren

B-Bäume

Definition: Sei $t \geq 1$ eine ganze Zahl. Ein B-Baum B der Klasse t ist entweder ein leerer Baum oder ein orientierter Baum der Höhe $h > 0$ mit folgenden Eigenschaften:

1. Jeder Pfad von der Wurzel zu einem Blatt hat die gleiche Länge $h - 1$.
2. Jeder Knoten außer der Wurzel und den Blättern hat mindestens t Söhne. Die Wurzel ist ein Blatt oder hat mindestens 2 Söhne.
3. Jeder Knoten hat höchstens $2t$ Söhne
4. Jedes Blatt mit der Ausnahme der Wurzel als Blatt hat mindestens $t - 1$ und höchstens $2t - 1$ Einträge.

Aufbau der Knoten im B-Baum

Innerer Knoten

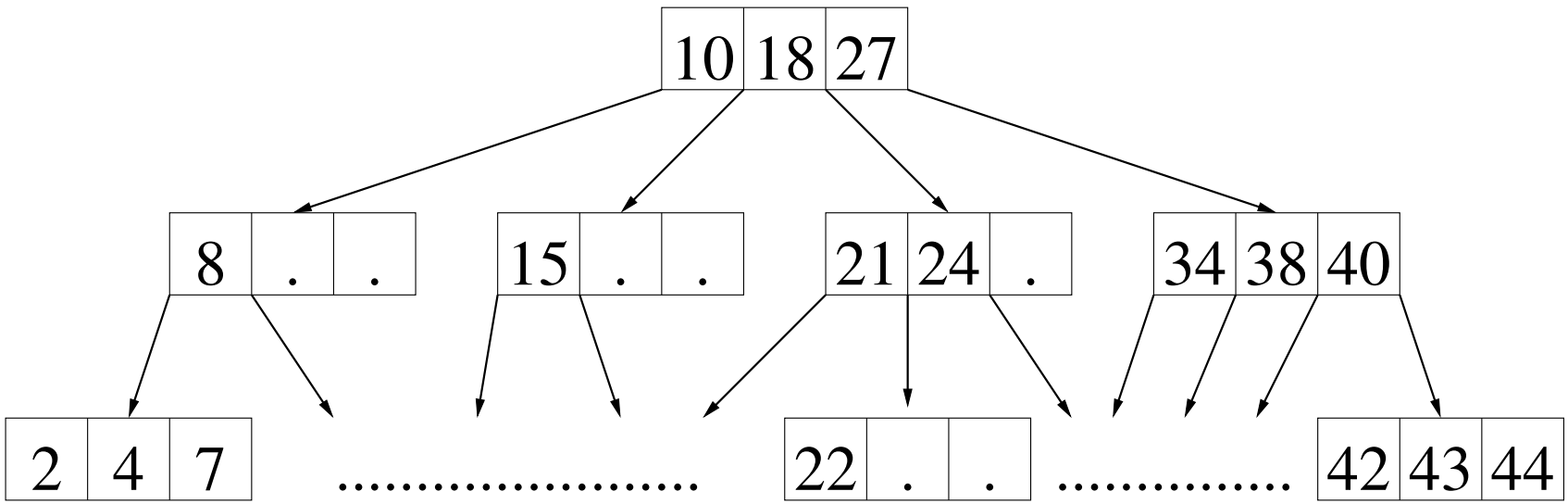
P_0	K_1	D_1	P_1	K_2	D_2	P_2	$\ldots$	K_b	D_b	P_b	freier Platz

Blattknoten

K_1	D_1	K_2	D_2	$\ldots$	K_b	D_b	freier Platz

- $P_0 \ldots P_b$ Zeiger auf Unterbäume
- $K_1 \ldots K_b$ Schlüssel
- $D_1 \ldots D_b$ (Zeiger auf) Daten
- $t - 1 \leq b \leq 2t - 1$

Beispiel: B-Baum mit $t = 2$



Suchen im B-Baum

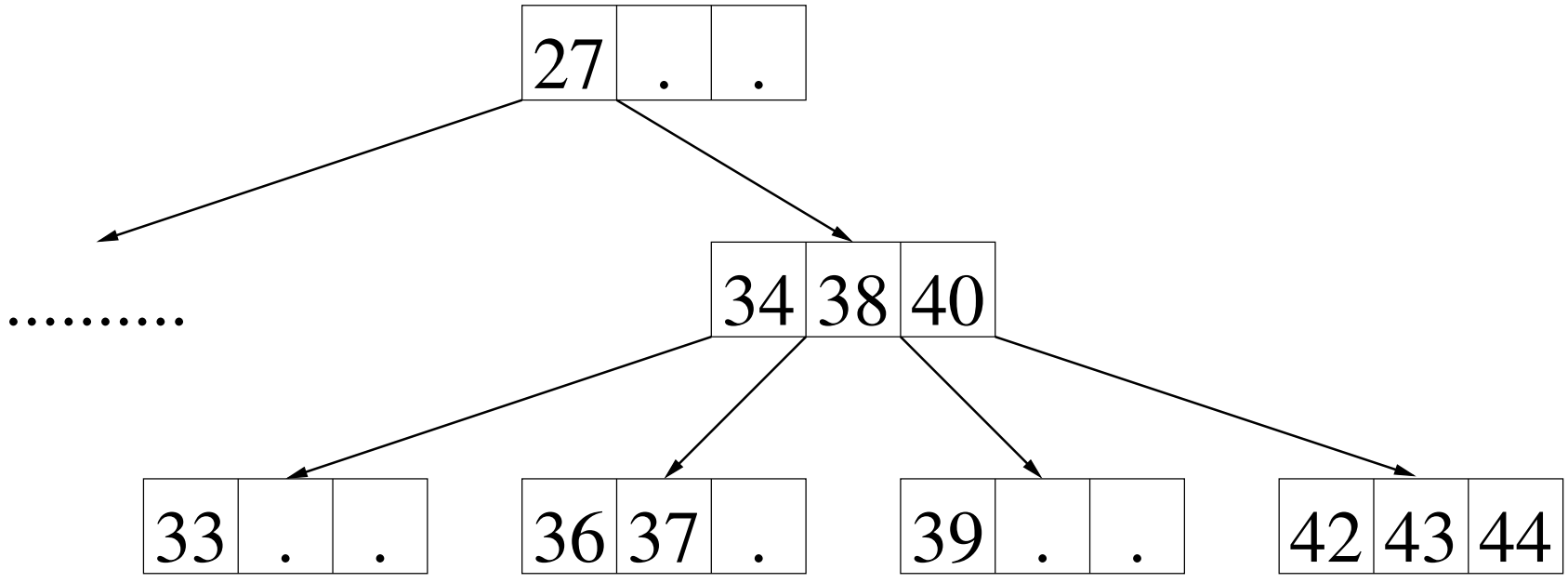
Verallgemeinerung der Suche im binären Suchbaum

1. Suche nach Schlüssel k beginnt mit der Wurzel als aktuellem Knoten x.
2. Finde größten Index $i \in \{1, \ldots, b\}$, so dass $k \geq K_i$ ist. Falls $k = K_i$, so wurde k gefunden (Ende, Suche erfolgreich).
3. Existiert so ein Index nicht, so setze $i = 0$.
4. Falls x bereits ein Blatt ist, Ende (Suche erfolglos).
5. Sonst: verzweige $x \leftarrow P_i$ und gehe zurück zu (2).

Einfügen im B-Baum

- Einfügen eines Schlüssels k in einen B-Baum geschieht immer in einem Blattknoten x.
- Wie bei der Suche wird ein Pfad von der Wurzel bis zu x durchlaufen.
- Trifft der Durchlauf auf einen vollen Knoten y, so wird y geteilt, bevor der Durchlauf fortgesetzt wird.
- Ein Knoten ist voll, wenn er die maximale Anzahl $2t - 1$ an Schlüsseln enthält.

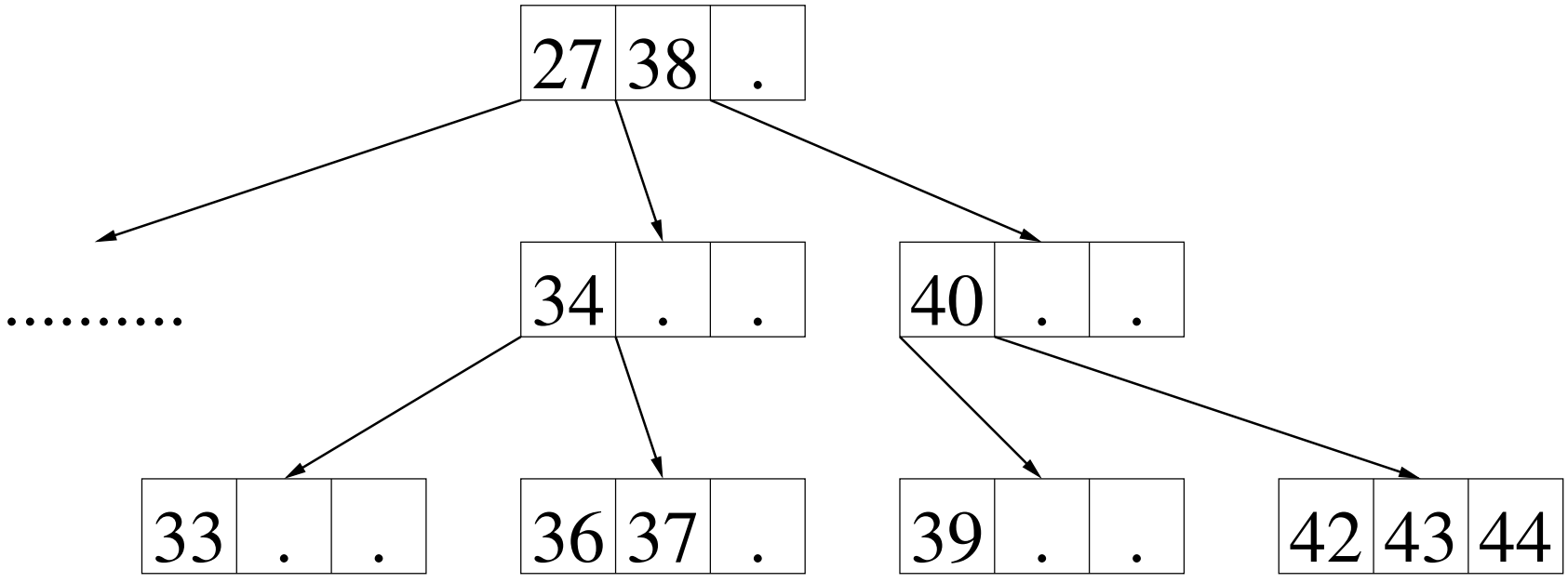
Teilen eines Knotens



Voraussetzungen für das Teilen eines Knotens x:

1. x ist voll (hat $2t - 1$ Schlüssel)
2. Elternknoten von x ist nicht voll

Teilen eines Knotens

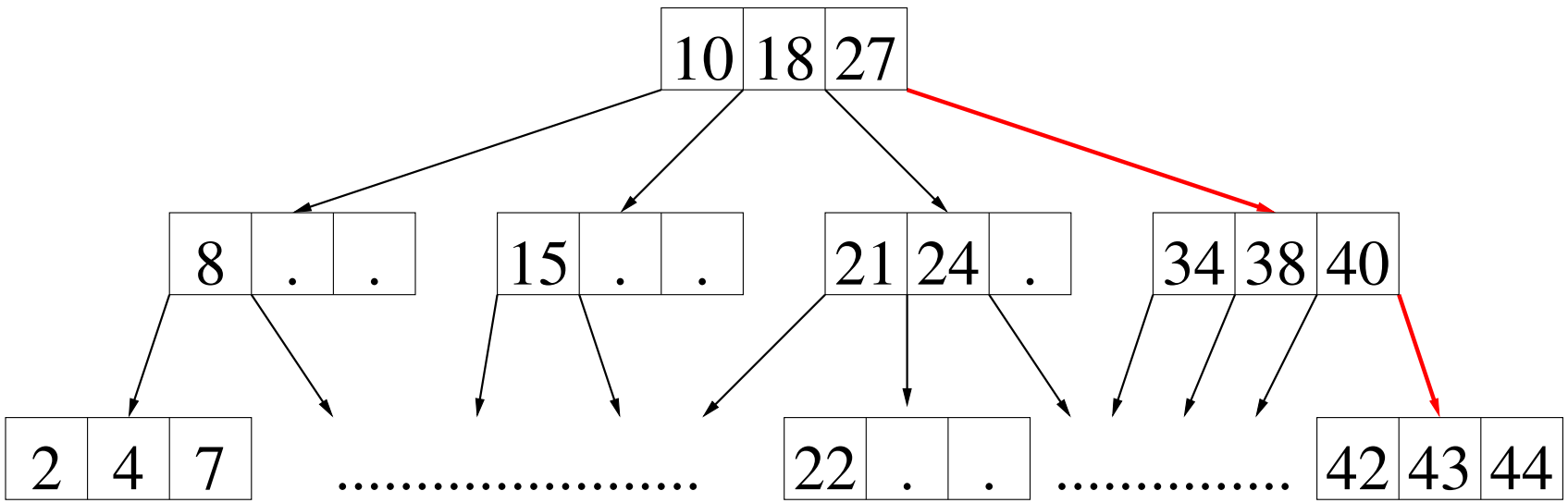


Voraussetzungen für das Teilen eines Knotens x:

1. x ist voll (hat $2t - 1$ Schlüssel)
2. Elternknoten von x ist nicht voll

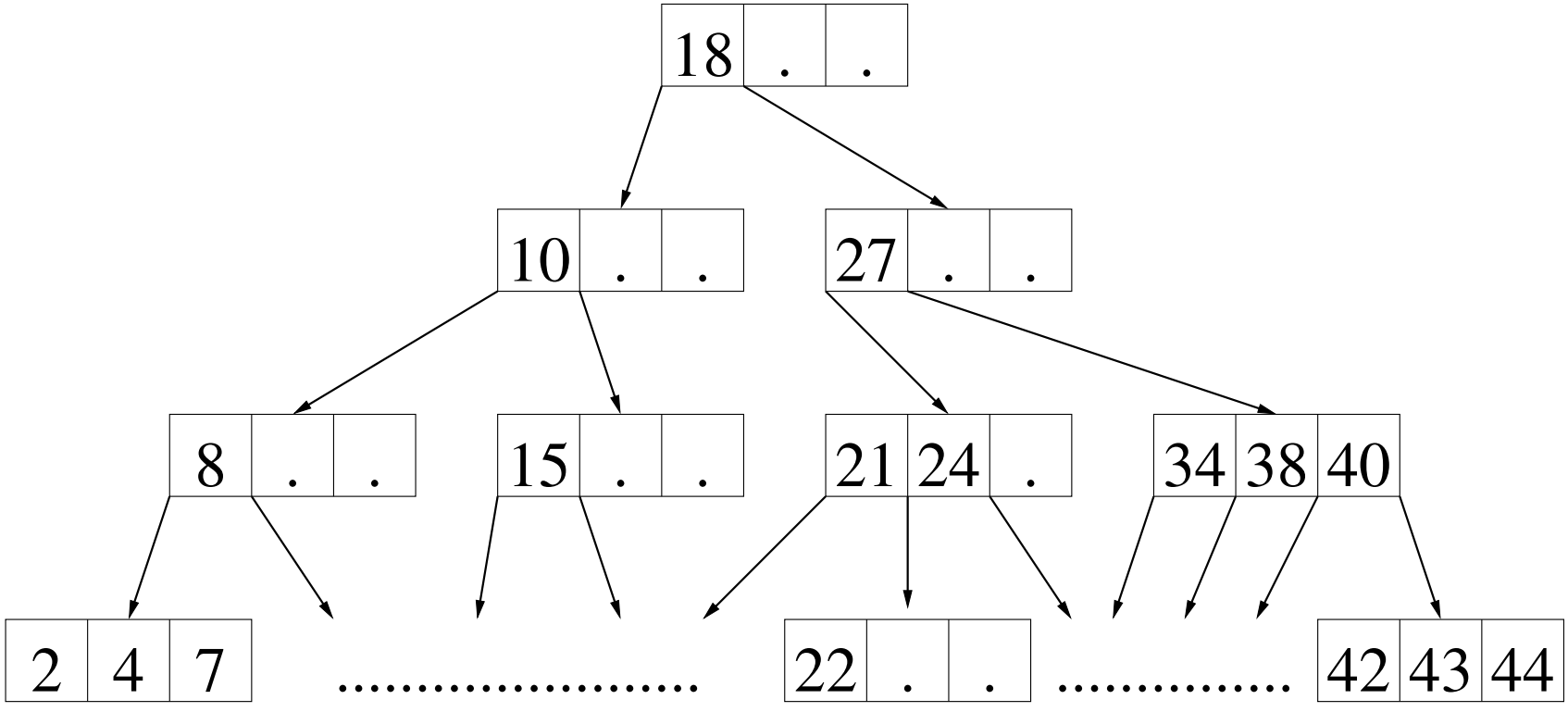
Beispiel: Einfügen im B-Baum

Einfügen von $k = 45$



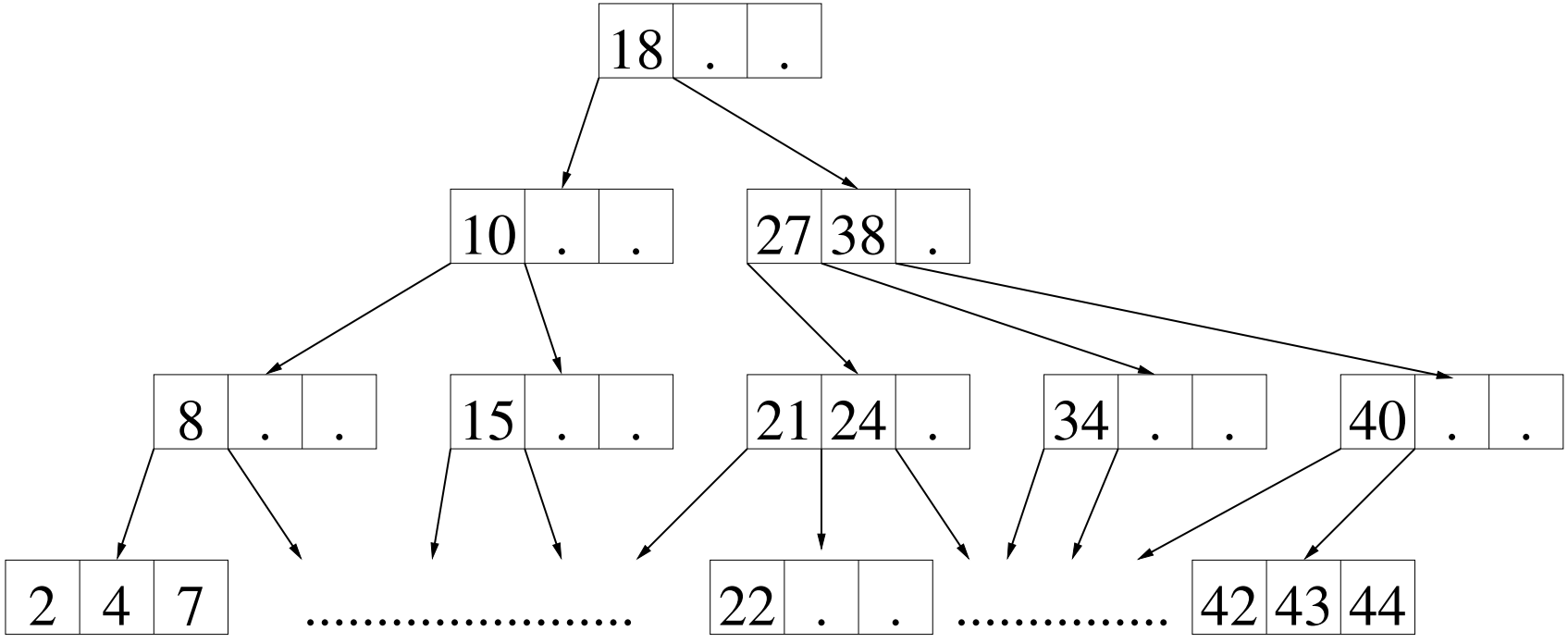
Beispiel: Einfügen im B-Baum

Einfügen von $k = 45$



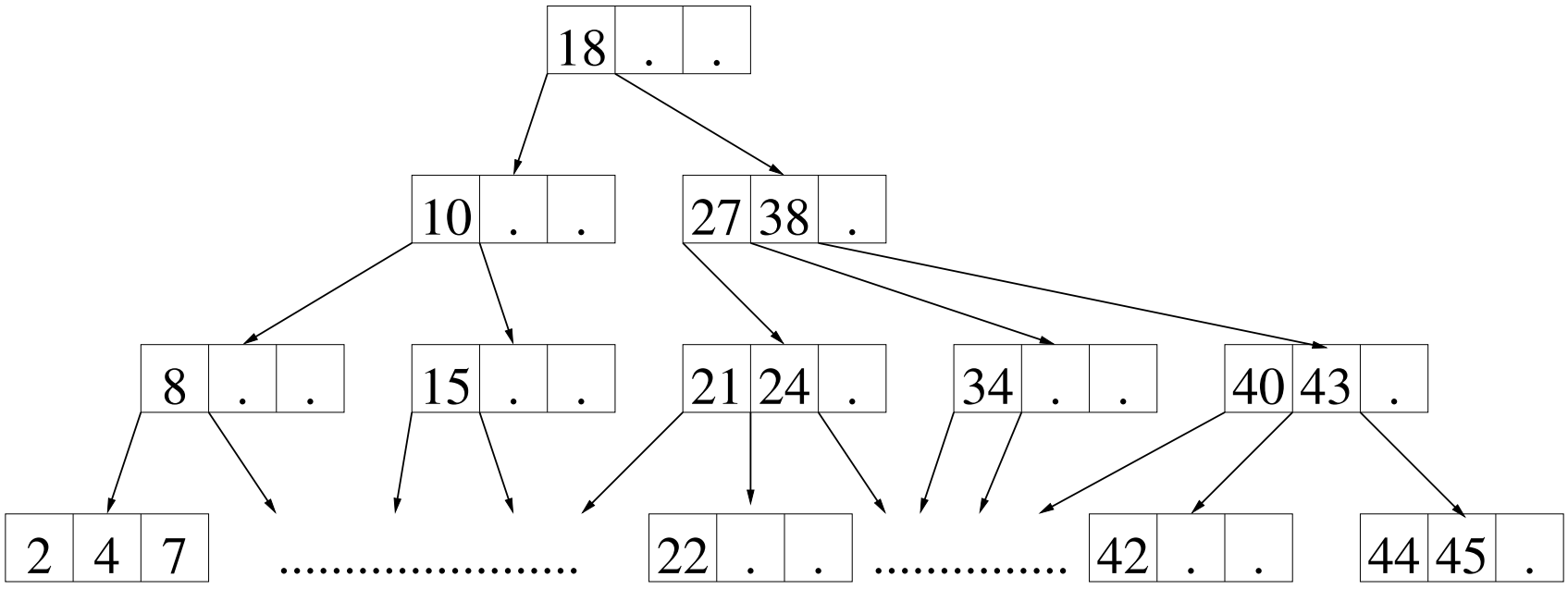
Beispiel: Einfügen im B-Baum

Einfügen von $k = 45$



Beispiel: Einfügen im B-Baum

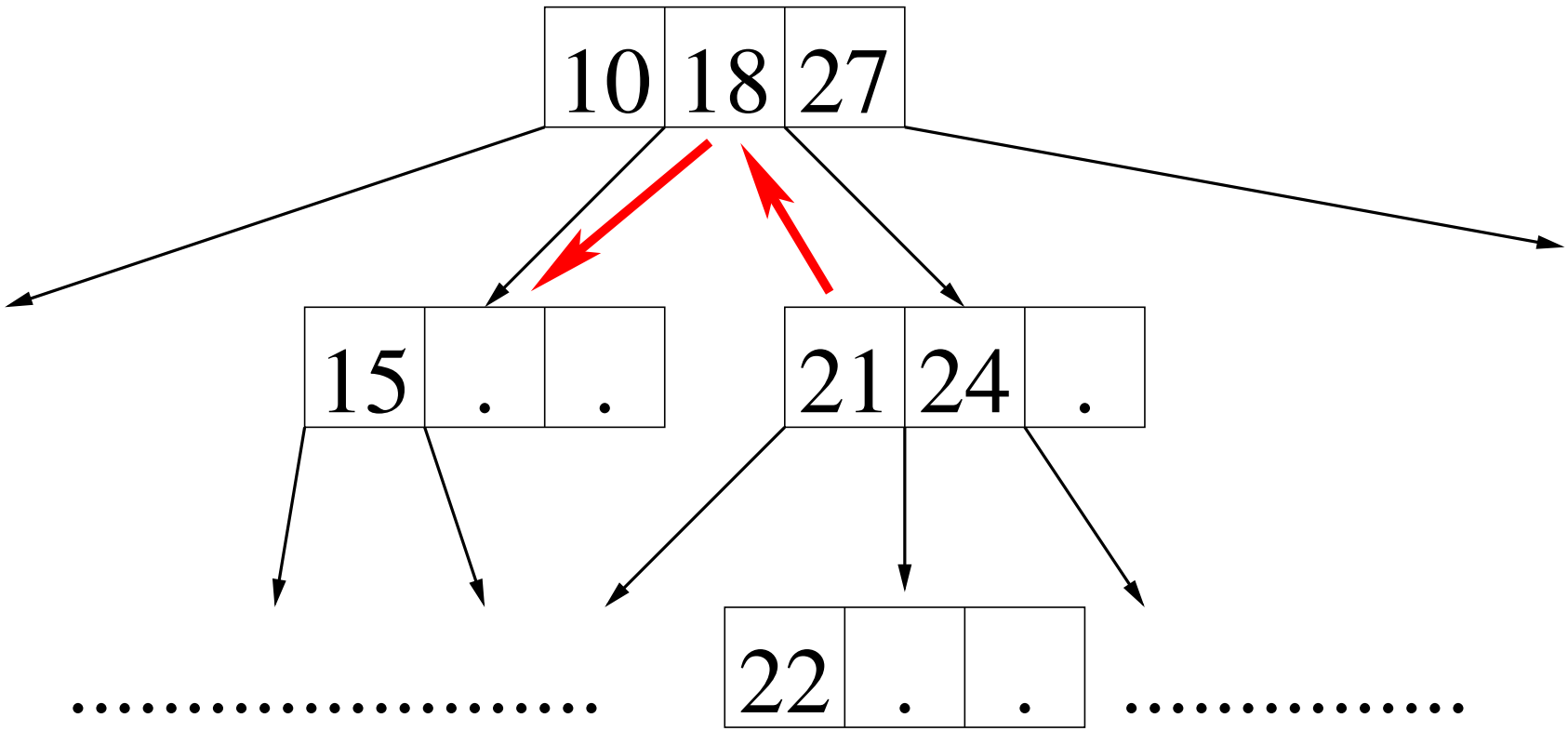
Einfügen von $k = 45$



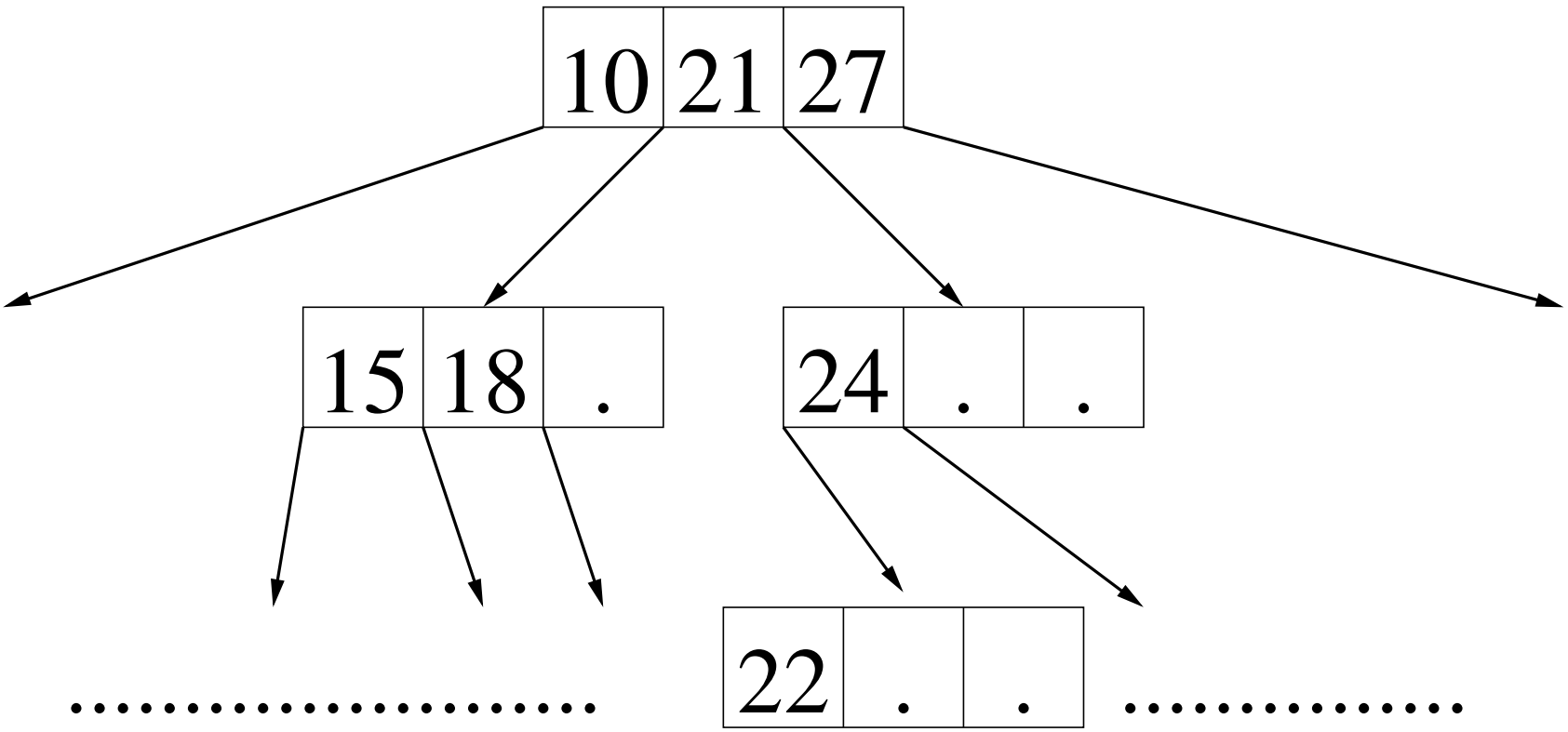
Löschen im B-Baum

- Löschen eines Schlüssels k beginnt mit einer Suche nach k, Durchlauf beginnend an der Wurzel.
- Trifft der Durchlauf auf einen Knoten y mit $t-1$ Schlüsseln (Minimalzahl), so wird y durch *Verschieben* eines Schlüssels auf t Schlüssel gefüllt, falls möglich. Ansonsten wird y mit einem Nachbarn *verschmolzen*.
- Durch das Verschieben/Verschmelzen hat der Knoten, aus dem k gelöscht wird, mindestens t Schlüssel, nach dem Löschen also mindestens $t-1$.

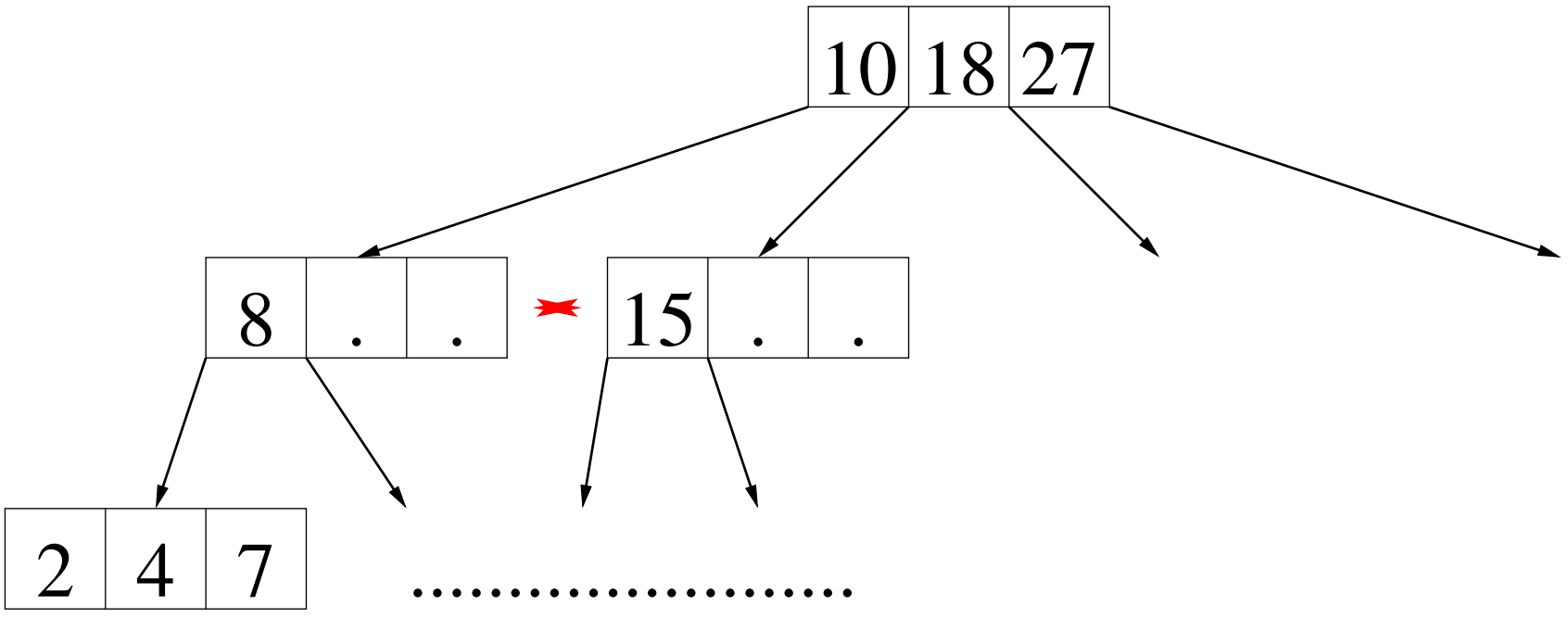
Verschieben eines Schlüssels



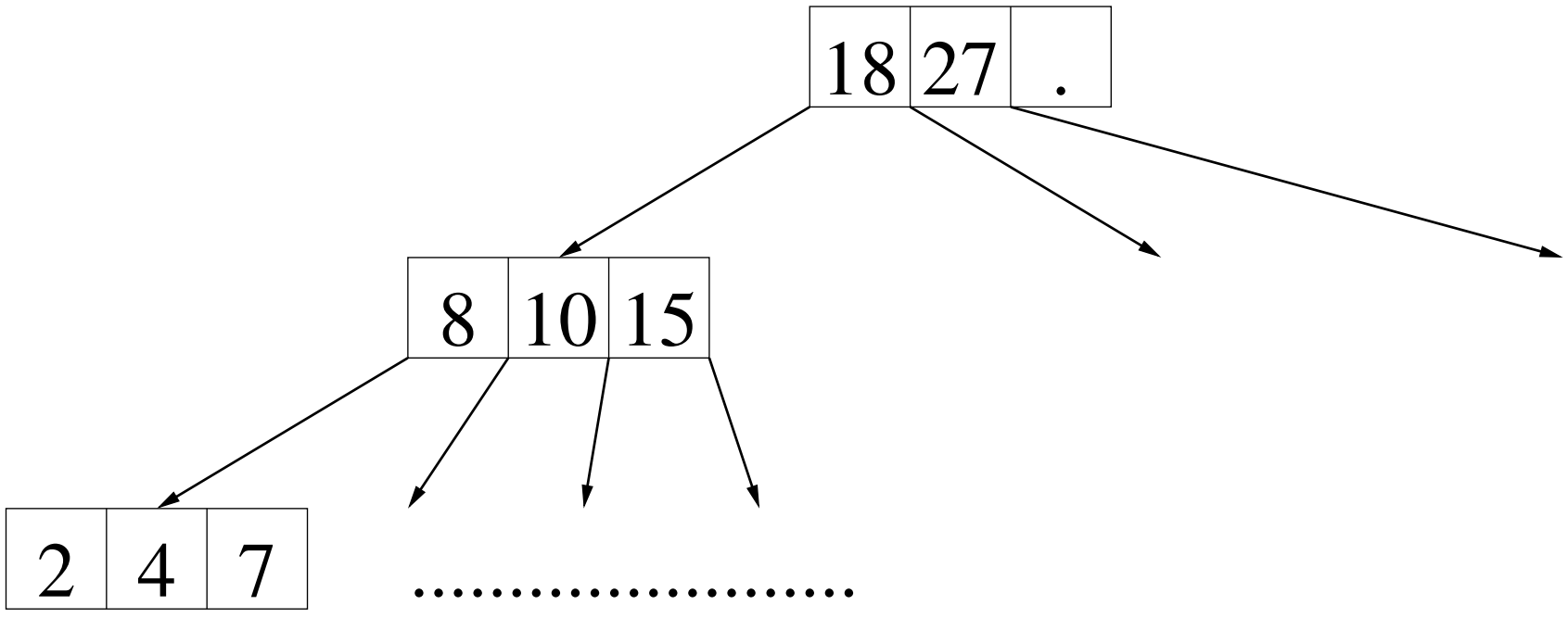
Verschieben eines Schlüssels



Verschmelzen zweier Knoten



Verschmelzen zweier Knoten



Höhe von B-Bäumen

Betrachte B-Baum der Höhe h mit minimalem Verzweigungsgrad t.

- 1 Knoten auf Stufe 0, mind. 2 Knoten auf Stufe 1
- mindestens $2t^{i-1}$ Knoten auf Stufe $i \geq 2$
- Gesamtzahl Knoten ist mindestens

$$1 + 2 + 2\sum_{i=2}^{h-1} t^{i-1}$$

- führt zu oberer Schranke für Höhe:

$$h \leq 1 + \log_t \frac{n+1}{2}$$

  $\Rightarrow$ Höhe wächst logarithmisch mit Zahl der Schlüssel
- Im Vergleich zum Binärbaum kann Höhe deutlich reduziert werden durch große Werte von t (Seitengröße).
- Beispiel: $t = 1024$ reduziert Höhe um Faktor 10 im Vergleich mit Binärbaum.

Variante: B*-Bäume

Hauptunterschied zu B-Baum: in inneren Knoten wird nur die Wegweiser- Funktion ausgenutzt

- innere Knoten führen nur (K_i, P_i) als Einträge
- Information (K_i, D_i) wird in den Blattknoten abgelegt. Dabei werden alle Schlüssel mit ihren zugehörigen Daten in Sortierreihenfolge in den Blättern abgelegt.
- Für einige K_i ergibt sich eine redundante Speicherung.
- Die inneren Knoten bilden also einen Index, der einen schnellen direkten Zugriff zu den Schlüsseln gestattet.
- Der Verzweigungsgrad erhöht sich beträchtlich, was wiederum die Höhe des Baumes reduziert
- Durch Verkettung aller Blattknoten lässt sich eine effiziente sequentielle Verarbeitung erreichen, die beim B-Baum einen umständlichen Durchlauf in symmetrischer Ordnung erforderte

Beispiel: Tafel

Digitale Suchbäume I

Prinzip digitaler Suchbäume (kurz: Digitalbäume):

- Zerlegung des Schlüssels - bestehend aus Zeichen eines Alphabets - in Teile
- Aufbau des Baumes nach Schlüsselteilen
- Suche im Baum durch Vergleich von Schlüsselteilen
- jede unterschiedliche Folge von Teilschlüsseln ergibt eigenen Suchweg im Baum
- alle Schlüssel mit dem gleichen Präfix haben in der Länge des Präfixes den gleichen Suchweg

Digitale Suchbäume II

Was sind Schlüsselteile ?

- Schlüsselteile können gebildet werden durch Elemente (Bits, Ziffern, Zeichen) eines Alphabets oder durch Zusammenfassungen dieser Grundelemente (z. B. Silben der Länge k)
- Höhe des Baumes $= \frac{l}{k} + 1$, wenn l die max. Schlüssellänge und k die Schlüsselteillänge ist

Beispiel: Tafel