

Mehrwegebäume — Motivation

- Wir haben gute Strukturen (AVL-Bäume) kennen gelernt, die die Anzahl der Operationen begrenzen
- Was ist, wenn der Baum zu groß für den Hauptspeicher ist?
- Externe Datenspeicherung → Speicherhierarchie:

- Zugriffsverhältnis zwischen Hauptspeicher und Festplatte:
 $t(P) / t(H) = (7 \cdot 10^{-3}) / (20 \cdot 10^{-9}) = 3.5 \cdot 10^5$

G. Zachmann Informatik 2 - SS 06 Bäume 92

- D.h., Zugriff auf (externe) Knoten sehr teuer
- Bsp. :
 - File-Tree des File-Systems
 - Datenbanken (DB2 (IBM), Sybase, Oracle, ...)
- Man hätte gerne einen Baum, der noch geringere Tiefe hat als $\log_2(n)$
- Idee:
 - "Erhöhe die Basis des Logarithmus"
 - oder: speichere mehrere "aufeinanderfolgende" Baum-Knoten auf derselben Seite (Page) auf der Platte
 - reduziere damit die Anzahl der Seitenzugriffe

G. Zachmann Informatik 2 - SS 06 Bäume 93

m-Wege-Bäume

- Ein **m-Wege-Suchbaum** ist eine Verallgemeinerung eines binären Suchbaumes (d. h., ein binärer Suchbaum ist ein 2-Wege-Suchbaum)

Anstatt 5 nur noch 2 Zugriffe auf Platte / CD

G. Zachmann Informatik 2 - SS 06 Bäume 94

Definition

- In einem **m-Wege-Baum** haben alle Knoten den Grad $\leq m$
- der Baum ist entweder leer oder besitzt folgende Eigenschaften:
 - jeder Knoten hat folgende Struktur:
 - K_i sind die Schlüssel, $1 \leq i \leq b$
 - b ist die Anzahl der Schlüssel im Knoten
 - P_i sind die Zeiger zu den Unterbäumen des Knotens
 - Schlüssel innerhalb eines Knotens sind aufsteigend geordnet:
 - $K_1 \leq K_2 \leq \dots \leq K_b$
 - alle Schlüssel im Unterbaum P_i sind kleiner als K_{i+1} , für $0 \leq i < b$
 - alle Schlüssel im Unterbaum P_b sind größer als K_b
 - Die Unterbäume P_i , für $0 \leq i \leq b$, sind ebenfalls m-Wege-Bäume

G. Zachmann Informatik 2 - SS 06 Bäume 95

Beispiel

G. Zachmann Informatik 2 - SS 06 Bäume 96

Bemerkungen

- Probleme:
 - Baum ist nicht ausgeglichen
 - Blätter sind auf verschiedenen Stufen
 - Bei Veränderungen gibt es keinen Ausgleichsalgorithmus
 - Schlechte Speicherausnutzung, kann zu verketteten Listen degenerieren
- Anzahl der Knoten im **vollständigen** m-Wege Baum mit Höhe h :

$$N = \sum_{i=0}^{h-1} m^i = \frac{m^h - 1}{m - 1}$$
- Maximale Anzahl n von Schlüsseln:
 - Sei b = Anzahl der Schlüssel im Knoten mit Grad m : $b \leq m-1$
 - $n \leq N(m-1) = m^h - 1$
- völlig degenerierter Baum: $n = N = h$
- Grenze für die Höhe eines m-Wege-Baumes: $\log_m(n+1) \leq h \leq n$

G. Zachmann Informatik 2 - SS 06 Bäume 97

Beispiel: Einfügen in einen 4-Wege-Baum

- Einfügen von 20, 35, 5, 95, 1, 25, 85

- Schlüssel in internen Knoten sind Schlüssel und Separatoren
- innerhalb eines Knotens: sequentielle Suche, auch binäre Suche möglich

G. Zachmann Informatik 2 - SS 06 Bäume 98

B-Bäume

- vielleicht die Datenstruktur, die in der Praxis am meisten benutzt wird
- B-Bäume sind **ausgeglichene** m-Wege-Bäume
- Da ein Knoten die Größe einer Übertragungseinheit hat (eine Page der Festplatte, typ. 4-16 kBytes), sind 50 Keys pro Knoten üblich
 - sehr flache Bäume
 - kurzer Weg von der Wurzel zu den Blättern

G. Zachmann Informatik 2 - SS 06 Bäume 99

Definition von B-Bäumen

- jeder Weg von der Wurzel zu einem Blatt ist gleich lang
- jeder Knoten (außer Wurzel) enthält mindestens k und höchstens $2k$ Schlüssel
- jeder Knoten (außer der Wurzel) hat zwischen $k+1$ und $2k+1$ Kinder (Unterbäume)
- die Wurzel ist entweder ein Blatt oder hat mindesten 2 Kinder
- jeder Knoten hat die Struktur:

b

|

K_1

|

K_2

|

...

|

K_b

|

frei

P_0

P_1

P_2

P_b

- b ist die Anzahl der Schlüssel im Knoten, $k \leq b \leq 2k$
- Schlüssel in Knoten sind aufsteigend sortiert ($K_1 < K_2 < \dots < K_b$)
- (L = Größe einer Seite)

G. Zachmann Informatik 2 - SS 06
Bäume 100

Bemerkungen

- CLRS benutzt eine etwas andere Definition:
 - Für einen B-Baum vom Grad t gilt:
 - jeder Knoten, außer der Wurzel, hat mindestens $t-1$ Schlüssel, also mindestens t Kinder
 - jeder Knoten besitzt höchstens $2t-1$ Schlüssel, also höchstens $2t$ Kinder
 - Formeln müssen „angepasst“ werden, damit sie stimmen
 - die Prüfung läuft gemäß den Folien, bei Unsicherheiten die Definition angeben

G. Zachmann Informatik 2 - SS 06
Bäume 101

Die Klasse $\tau(k, h)$

- die Klasse $\tau(k, h)$ bezeichnet alle B-Bäume mit $k > 0$ und der Höhe $h \geq 0$
- Beispiel: B-Baum der Klasse $\tau(2, 3)$

G. Zachmann Informatik 2 - SS 06
Bäume 102

Höhe eines B-Baumes

- Für die Höhe eines B-Baumes mit n Schlüsseln gilt:

$$\log_{2k+1}(n+1) \leq h \leq \log_{k+1}\left(\frac{n+1}{2}\right) + 1$$

G. Zachmann Informatik 2 - SS 06
Bäume 103

Erklärung

- in einem **minimalen** B-Baum hat jeder Knoten die **kleinstmögliche** Anzahl an Kindern:

$$N_{\min}(k, h) = 1 + 2 + 2 \left((k+1) + (k+1)^2 + \dots + (k+1)^{h-2} \right)$$

$$= 1 + 2 \sum_{i=0}^{h-2} (k+1)^i = 1 + 2 \frac{(k+1)^{h-1} - 1}{k}$$
- in einem **maximalen** B-Baum hat jeder Knoten die **größtmögliche** Anzahl $(2k+1)$ an Kindern:

$$N_{\max}(k, h) = 1 + (2k+1) + (2k+1)^2 + \dots + (2k+1)^{h-1}$$

$$= \sum_{i=0}^{h-1} (2k+1)^i = \frac{(2k+1)^h - 1}{2k}$$

G. Zachmann Informatik 2 - SS 06 Bäume 104

- die Höhe definiert eine obere und untere Schranke für die Anzahl der Knoten $N(B)$ eines beliebigen B-Baumes der Klasse $\tau(k, h)$:

$$1 + 2 \frac{(k+1)^{h-1} - 1}{k} \leq N(B) \leq \frac{(2k+1)^h - 1}{2k} \quad \text{für } h \geq 1$$

$$N(B) = 0 \quad \text{für } h = 0$$
- von $N_{\min}(B)$ und $N_{\max}(B)$ lässt sich ableiten:

$$n \geq n_{\min} = 1 + 2 \frac{(k+1)^{h-1} - 1}{k} k = 2(k+1)^{h-1} - 1$$

$$n \leq n_{\max} = \frac{(2k+1)^h - 1}{2k} 2k = (2k+1)^h - 1$$
- Somit gilt für die Schlüsselanzahl in einem B-Baum:

$$2(k+1)^{h-1} - 1 \leq n \leq (2k+1)^h - 1$$

G. Zachmann Informatik 2 - SS 06 Bäume 105

Einfügen in einen B-Baum

- füge anfangs in ein leeres Feld der Wurzel ein
- die ersten $2k$ Schlüssel werden geordnet in die Wurzel eingefügt
- der nächste, der $(2k+1)$ -te, Schlüssel passt nicht und erzeugt einen "Überlauf"
- die Wurzel wird **geteilt**, jeder der beiden Knoten bekommt k Keys, ein neuer Knoten wird Wurzel
- die ersten k Schlüssel kommen in den linken Unterbaum
- die letzten k Schlüssel kommen in den rechten Unterbaum
- Der **Median** der $(2k+1)$ Schlüssel wird zum neuen **Separator** im neuen Wurzel-Knoten

G. Zachmann Informatik 2 - SS 06 Bäume 106

```

    graph TD
      Root[K_{k+1}] --> Left[K_1 ... K_k]
      Root --> Right[K_{k+2} ... K_{2k+1}]
      Root --- Median[K_{k+1}]
  
```

- neue Schlüssel werden in den Blättern sortiert gespeichert
- läuft ein Blatt über $(2k+1)$ Keys, wird ein **Split** durchgeführt:
 - Das Blatt wird geteilt (ergibt 2 Knoten à k Keys),
 - diese werden im Vaterknoten anstelle des ursprünglichen Blattes verankert, und
 - der Median-Schlüssel wandert in den Vaterknoten
- ist auch der Elternknoten voll, wird das Splitten fortgesetzt
- Im worst-case bis zur Wurzel

G. Zachmann Informatik 2 - SS 06 Bäume 107

Beispiel

- $k = 2 \rightarrow 2 \leq \# \text{Schlüssel} \leq 4$ (außer Wurzel)
- Einzufügende Schlüssel:
 - 77, 12, 48, 69
 - 33, 89, 97
 - 91, 37, 45, 83
 - 2, 5, 57, 90, 95

G. Zachmann Informatik 2 - SS 06 Bäume 108

- Einzufügende Schlüssel:
 - 99
 - 50

G. Zachmann Informatik 2 - SS 06 Bäume 109

Variante statt Split (B*-Baum)

- Idee: Ausgleich mit Nachbarknoten statt Split
- füge ein:
 - 8
 - 6

G. Zachmann Informatik 2 - SS 06 Bäume 110

Löschen in einem B-Baum

- Bedingung durch Def.: minimale Belegung eines Knotens erhalten
- Methoden:
 - Ausgleich mit Nachbarknoten (Rotation)
 - Mischung (Merge)
- Beispiel: K_k wird gelöscht:
 - Wende Rotation an
 - Schlüssel K'_k wird in den Vaterknoten verschoben, Schlüssel K_k kommt in den rechten Knoten (mit $k-1$ Knoten)

G. Zachmann Informatik 2 - SS 06 Bäume 111

- um wiederholte Anpassungen (durch neue Löschooperationen) zu vermeiden, kann mehr als ein Schlüssel rotiert werden
 - beide Knoten haben danach ungefähr die gleiche Schlüsselanzahl
- Beispiel: 33 soll entfernt werden

Unterlauf

G. Zachmann Informatik 2 - SS 06 Bäume 112

- Zweite Art der Unterlauf-Behandlung: **Mischen**
 - wird benutzt, wenn es einen Unterlauf gibt und beide Nachbarknoten genau k Schlüssel haben
 - d.h., man kann nichts vom linken / rechten Knoten "stehlen"
 - Beispiel: K_k löschen
- Der Separator und Schlüssel $k-1$ werden mit den k Schlüsseln im anderen Knoten eingefügt

G. Zachmann Informatik 2 - SS 06 Bäume 113

Allgemeiner Lösch-Algorithmus

- x = zu löschendes Element
- Suche x
 - x ist in einem Blatt:
 - lösche x
 - $\#E \geq k$ (nach dem Löschen) \rightarrow OK
 - $\#E = k-1$ und $\#E' > k \rightarrow$ Rotation
 - $\#E = k-1$ und $\#E' = k \rightarrow$ Mischung
 - x ist in einem internen Knoten:
 - ersetze x mit K_b oder K_1' (nächstgößerer oder -kleiner Schlüssel)
 - entferne den Separator (K_b oder K_1') aus dem entsprechenden Kind-Knoten
 - behandle diesen Kind-Knoten mit Rekursion

G. Zachmann Informatik 2 - SS 06 Bäume 114

Beispiele

- lösche:
 - 21
 - 11 \rightarrow Unterlauf
 - \rightarrow Mischen

G. Zachmann Informatik 2 - SS 06 Bäume 115

B+-Bäume

- in einem B-Baum haben die Schlüssel K_i 2 Funktionen:
 - K_i dienen als Separatoren
 - Nutzdaten werden mit im Baum gespeichert (auch in inneren Knoten)
- ein B+-Baum teilt diese Eigenschaften auf:
 - K_i werden nur als Separatoren benutzt (ergibt einen flacheren Baum)
 - Schlüssel werden redundant gespeichert (vereinfachte Löschoptionen wiegen den zusätzlichen Speicherbedarf auf)
 - die Blätter werden zu einer Liste verlinkt
 - auf die Daten kann man in sequentieller Ordnung zugreifen
- interne Knoten sind ein sog. **Index** auf die Blätter

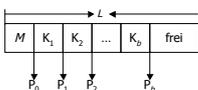
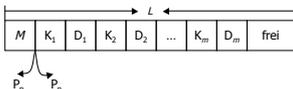
G. Zachmann Informatik 2 - SS 06 Bäume 117

Definition eines B+-Baum

- Seien k, l, h ganze Zahlen mit $h \geq 0, k > 0, l > 0$
- Ein B+-Baum der Klasse $\tau(k, l, h)$ ist entweder ein leerer Baum oder ein sortierter Baum mit den folgenden Eigenschaften:
 - alle Wege von der Wurzel zu einem Blatt haben die gleiche Länge
 - jeder innere Knoten (außer Wurzel) enthält mindestens k und höchstens $2k$ Schlüssel
 - jeder innere Knoten (außer der Wurzel) hat zwischen $k+1$ und $2k+1$ Kinder
 - alle Blätter (außer der Wurzel als Blatt) haben mindestens l und höchstens $2l$ Schlüssel
 - Schlüssel können **mehrfach** auf einem Weg zu den Blättern vorkommen
 - in inneren Knoten nur Separatoren ohne Nutzdaten ("satellite data")
 - an Blättern zusammen mit den Nutzdaten
 - Alle Blätter sind zu einer sortierten Liste verlinkt

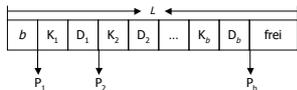
G. Zachmann Informatik 2 - SS 06 Bäume 118

Format der Knoten

- die **internen** Knoten haben folgendes Format:
 
 - M gibt den Typ des Knoten und die Anzahl der Schlüssel ($k \leq b \leq 2k$) an
- die **Blätter** haben folgendes Format:
 
 - M gibt den Typ des Knoten und die Anzahl der Schlüssel ($l \leq m \leq 2l$) an
 - P_p zeigt auf den vorhergehenden und P_n auf den nächsten Blatt-Knoten

G. Zachmann Informatik 2 - SS 06 Bäume 119

Größe der Knoten im B-Baum



- b ist die Anzahl der Elemente, mit Länge L_b
- Elemente mit fester Länge (L_K und L_D)
- Zeiger haben Länge L_P
- Max. Anzahl der Einträge ist also:

$$b_{\max} = \left\lfloor \frac{L - L_b - L_P}{L_K + L_D + L_P} \right\rfloor = 2k$$

G. Zachmann Informatik 2 - SS 06 Bäume 120



Größe der Knoten im B⁺-Baum

- für Knoten mit fester Länge L kann k aus L errechnet werden

- interne Knoten:

$$L = L_M + L_P + 2k(L_K + L_P)$$

$$k = \left\lfloor \frac{L - L_M - L_P}{2(L_K + L_P)} \right\rfloor$$

- Blätter:

$$L = L_M + 2L_P + 2l(L_K + L_D)$$

$$l = \left\lfloor \frac{L - L_M - 2L_P}{2(L_K + L_D)} \right\rfloor$$



Beispiel

- B⁺-Baum der Klasse $\tau(3, 2, 3)$
[$k = 3, l = 2, h = 3$]

