Kap. 2.3.1 B-Bäume

Bayer, Mc Creight, Herbst 1969

Problemstellung: Wie bei AVL-Bäumen:
Verwaltung einer Menge $\Sigma = \{(x, \alpha)\}$
$x \in X$, Schlüsselmenge
α : assoziierte Info

Operationen: **insert** (x, α)
search (x)
delete (x)
zusätzlich sequentielle Verarbeitung
nach $<$ oder $\leq$ auf X:
reset
next
eof
} scan, cursor über Σ

$<$ ist meist lexikographische Ordnung.

1

Beispiele:

1. Telefonbuch: {(N,T)}
   Vor- u. Nachteile von $<$?
2. Dateienkatalog
3. Kundendatei: $<$?
4. Freie Speichergebiete
5. Lagerhaltung mit Teile #
6. Zeitreihen von Aktienkursen, Meßwerten mit mehrdimensionalem X

2

Problem: $\Sigma = \{(x, \alpha)\}$ ist meist sehr große, dyn. Menge von **Variablen**, aber zur Übersetzungszeit unbekannt, Deklaration und Manipulation interaktiv.

Manuelle Lösung: Sortierte Kartei mit Indexkarten

HS-Lösung: AVL-Baum

3

Grundidee: Transporteinheit = 1 Plattenseite = 1 Baumknoten

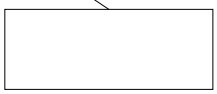
insert in leeren Baum

x_1, α_1	x_2, α_2	. . .	$x_{2k} \alpha_{2k,}$

(x, α) einsortieren in $<$,
Maximal 2k Einträge.
Ab jetzt ignoriere α_i
durch insert des nächsten Elementes:

$x_1 x_2$ ... x_{2k}	x_{2k+1} ?

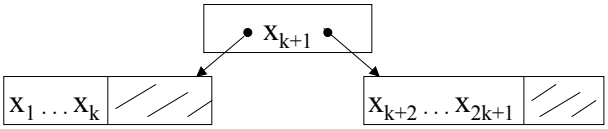
Überlauf



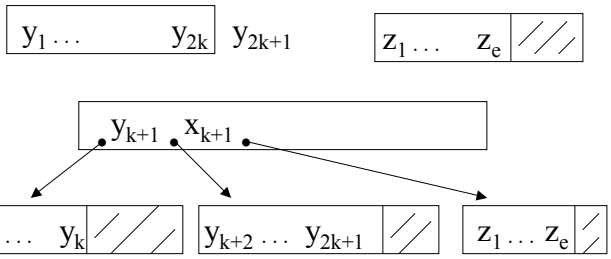
Scientific American!

4

Spaltung:



u. rekursive Fortsetzung bei Spaltung eines weiteren Knotens



5

Hinw: Höhenwachstum nicht durch „Austreiben“ neuer Blätter nach unten, sondern durch Überlauf u. Spaltung der Wurzel.

- Baumstruktur?
- Knotenstruktur?

Def: von ***B-Bäumen***: Seien h,k ganz, $h \geq 0$, $k > 0$. Ein ***B-Baum T*** der Klasse τ (k, h) ist entweder

a) $\varnothing$, d.h. $h = 0$ oder
b) Ein geordneter Baum, in dem
   i) jeder Pfad von Wurzel zu einem Blatt hat Länge $h - 1$
   ii) Wurzel ist selbst ein Blatt oder hat ≥ 2 Söhne; andere interne Knoten haben $\geq k + 1$ Söhne
   iii) jeder Knoten hat $\leq 2\,k+1$ Söhne

6

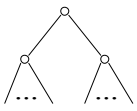
Lemma: Der minimale, maximale Baum in τ (k, h) habe N_{min} (k, h), N_{max} (k, h) Knoten.

Dann gilt: $N_{min}^{(k,h)} = 1 + \frac{2}{k} ((k+1)^{h-1} - 1)$

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

7

Bew: N_{min}: Wurzel hat 2 Söhne,
sonst k+1 Söhne



1
2
$2 \cdot (k+1)$
$2 \cdot (k+1)^2$
$\vdots$

$$\sum = 1 + 2 \cdot ((k+1)^0 + (k+1)^1 + \\ + \ldots (k+1)^{h-2}) \\ = 1 + \frac{2}{k} ((k+1)^{h-1} - 1)$$

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

Hinw: Baum T mit N (T) Knoten kann aus vielen Klassen sein mit unterschiedlichen k und h.
Konstruiere ein $T \in \tau (2, 3) \cap \tau (3, 3)$ sowie
$T` \in \tau (2, 3)$ und $T`` \in \tau (2, 4)$ mit N (T`) = N (T``)

8

Def: Knotenaufbau:

1 B-Baum Knoten ~ 1 Plattenseite

Indexelement: (x, α)

i) Wurzelseite hat 1 bis 2k Indexelemente, andere Seiten haben k bis 2k Indexelemente

ii) Seite P mit ℓ Indexelementen (Schlüsseln) hat $\ell + 1$ Söhne, außer wenn P Blatt ist

iii) $x_1, x_2, ..., x_\ell$ sind auf P geordnet nach < auf X. Falls P kein Blatt ist, sind auf P $\ell + 1$ Zeiger $p_0, p_1, ..., p_\ell$ auf Söhne von P

iv) Sei P (p_i) Seite, auf die p_i zeigt, T (p_i) Unterbaum von T mit Wurzel P (p_i) K (p_i) Menge der Schlüssel (keys) in T (p_i), dann gilt:

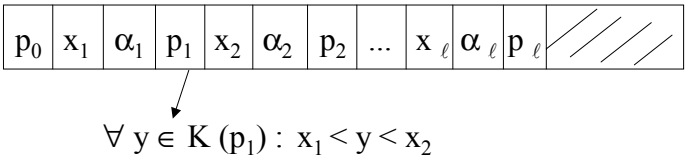
9

$\forall\, y \in K(p_0) \;:\; y < x_1$

$\forall\, y \in K(p_i) \;:\; x_i < y < x_{i+1}$ für $i = 1, ..., \ell - 1$

$\forall\, y \in K(p_\ell) \;:\; x_\ell < y$

Seitenorganisation:



$\forall\, y \in K(p_1) : x_1 < y < x_2$

10

Durchlaufalg: ~ Nachordnung für lexikogr. Schlüsselordnung
T (p), Wurzel P (p) mit ℓ (p) Schlüsseln:

if T (p) $\neq \varnothing$ **then**
begin durchlaufe T (p_0);
for i:= 1 **to** ℓ (p) **do**
begin verarbeite (x_i, α_i);
durchlaufe T (p_i)
end
end

Hinw: Rekursion braucht Keller von Seiten entsprechend Pfad durch B-Baum, i.e. maximal h Seiten.

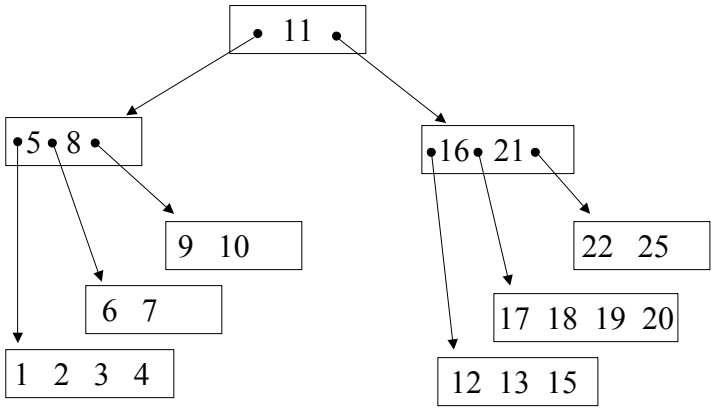
11

Suchalgorithmus: finde beliebiges y in T (p) :
Parameter: y, p:
if T(p) = $\varnothing$ **then** Mißerfolg
else if $y < x_1$ **then** suche y in T (p_0)
else if $y = x_i$ **then** {y gef.} verarbeite (x_i, α_i)
else if $x_i < y < x_{i+1}$ **then** suche y in T (p_i)
else {$x_\ell < y$} suche y in T(p_ℓ)

Hinw: Die Fälle $y < x_1$, $y = x_i$, $x_i < y < x_{i+1}$
erfordern Suche innerhalb der Seite P(p).
Seiten so organisieren, daß binäre Suche
in P(p) möglich ist.

12

Beispiel:



13

Anzahl Indexelemente in T:

sei $T \in \tau\,(k, h)$:

$I_{min}\,(k, h),\ I_{max}\,(k, h)$?

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

$\Rightarrow \quad I_{min}\,(k, h) = 2\,(k + 1)^{h-1} - 1$

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

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

sei I Anzahl Indexelemente in T

$I_{min} \le I \le I_{max}$

$\Rightarrow \quad \frac{I+1}{2} \ge (k + 1)^{h-1}$

$I + 1 \le (2k + 1)^{h}$

14

Höhenungleichung:

⇒ **Logarithmisches Höhenwachstum:**

$\log_{2k+1}(I+1) \leq h \leq 1 + \log_{k+1}(\frac{I+1}{2})$

Beispiel: Sei h = 4, k = 200

d.h. zu Blatt von T 2-3 Plattenzugriffe mit Caching

$I \geq 2 \cdot 200^3 = 1.6 \cdot 10^7$

$I \leq 400^4 = 256 \cdot 10^8 = 2.6 \cdot 10^{10}$

bei 20 Bytes/ Element:

320 MB = $3.2 \cdot 10^8$ B ≤ I ≤ $5.2 \cdot 10^{11}$ = 520 GB

⇒ ***B-Baum kann um Faktor 1000 wachsen, ohne wesentliche Änderung des Zugriffsverhaltens !!!***

⇒ ***Bedeutung für DBen***

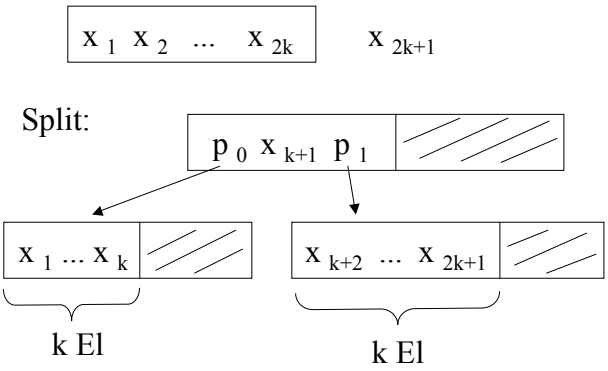
15

Einfüge-Algorithmus: $T \in \tau(k, h)$

Annahme: x_i, p_i, α_i feste Länge

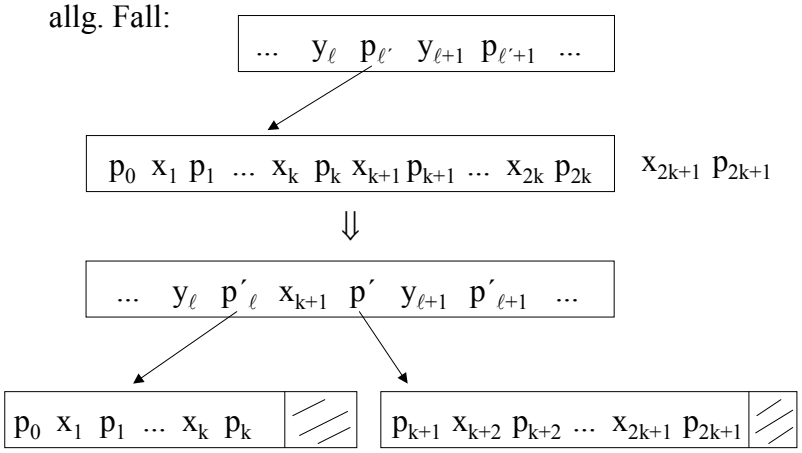
Start : T = ∅

Aufbau der Wurzel:



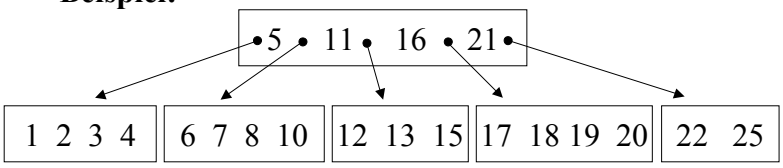
16

allg. Fall:



17

Beispiel:



durch Eintrag von 9 entsteht Baum auf Folie 13

Übung: Wähle Zahlenfolge u. konstruiere
$T \in \tau(2, h)$ durch Einfügung in $\varnothing$.

Übung: Andere Reihenfolge der Einfügung derselben Zahlen führt i. a. zu anderem Baum. Beispiel?

18

Kostenanalyse Einfügung:

f_{min} = minimale Anzahl zu *holender* Seiten

w_{min} = minimale Anzahl zu *schreibender* Seiten

f_{max}, w_{max} analog

- Suchvorgang: $f_{min} = 1$

$f_{max} = h$

$w_{min} = w_{max} = 0$

- Einfügen: $f_{min} = h, f_{max} = h$

$w_{min} = 1, w_{max} = 2h + 1$

19

Fall w_{max} zeigt **Höhenwachstum**:

spalte Pfad einschl. Wurzel!

tritt sehr selten auf, für $T \in \tau$ (k, h) nur h-1 mal

Amortisierte, durchschnittliche Aufbaukosten:

Annahme: in T wird nur eingefügt u. gesucht, praxisnahe!

Seien I Indexelemente in T

n(I) : Anzahl der Knoten von T

$n(I) \leq \frac{I-1}{k} + 1$

beim Aufbau von T ergeben sich

$s(I) \leq \frac{I-1}{k}$ Spaltungen

20

Bew: erste Seite pro Ebene entsteht durch Höhenzuwachs, alle anderen Seiten entstehen durch Splits

Anzahl Splits = n(I)-h =: s(I)

$$n(I) \leq \frac{I-1}{k}+1$$

$$s(I) \leq \frac{I-1}{k}+1-h \leq \frac{I-1}{k}$$

21

Beim Aufbau von T ergeben sich
2 • s (I) *writes* wegen Spaltungen

w = I + 2 · s (I) *writes* insgesamt

mittlere Anzahl writes pro **insert**

$$w_a = \frac{w}{I} = 1 + 2 \cdot \frac{s(I)}{I} < 1 + \frac{2}{k}$$

heute : k ≥ 100

⇒ weniger als 1.02 writes/ insert

Kostendurchschnitt pro Einfügung:

$f_a = h$; $w_a < 1 + \frac{2}{k}$ ~ 50ms/insert

10^6 inserts ~ 50.000 sec ~ 1 Tag
10 GB ~ 10^8 inserts ~ 100 Tage

22

Löschalgorithmus: delete (x):

Schritt 1: search (x)

Schritt 2: eigentliches delete

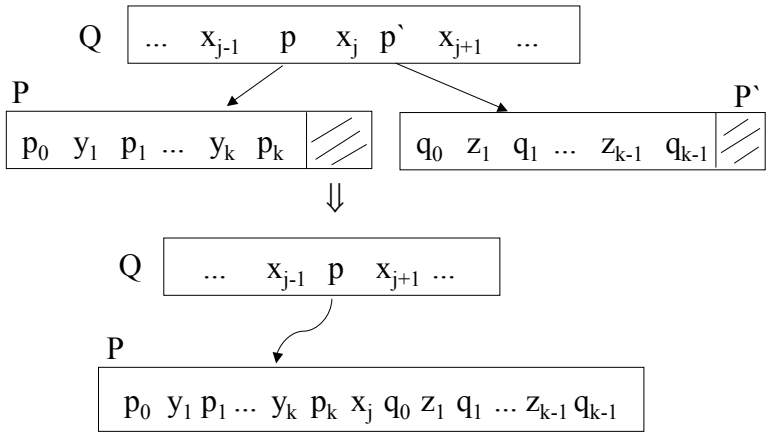
2a) x steht auf Blatt

2a.1) noch mindestens k Elemente auf Blatt nach delete

2a.2) Konkatenation: k-1 El. auf Blatt

k El. auf einer Nachbarseite

23



2k Einträge. Rekursive Fort-pflanzung bis zur Wurzel

24

2a.3) Unterlauf:

k - 1 El. auf P`, ℓ > k auf P
konkateniere P` mit P u. spalte
keine Fortpflanzung der Spaltung zur Wurzel hin
Verschiebung P $\rightarrow$ P` mit Anpassung
von Trennschlüssel in Vaterknoten.

25

2b) x steht auf internem Knoten:

Ersetze x durch nächstgrößeres
(nächstkleineres) El. y {y auf Blatt}
Jetzt Löschung von Blatt wie 2a mit Unterfällen.
(Siehe auch Löschung von Zwischenknoten
in AVL-Bäumen)

26

Kosten des Löschalg:

1. $f_{min} = h$; $w_{min} = 1$
2. lösche x von internem Knoten
   $f = h$; $w = 2$
3. $f_{max} = 2h - 1$

   $w_{max} = h + 1$

   Pfadkonkatenation plus Unterlauf

   in Wurzelnachfolger

27

Speichernutzung

ungünstigster Fall 50 % Nutzung,
z.B bei Einfügung in Sortierordnung

Verbesserung: Verzögerung von Splits durch Überlauf: i.e. Konkatenation mit Nachbarn u. Split.

100% 50% 50% 100% 50% 50% 100%

$^2/_3$ $^2/_3$

= 66% Mindestausnutzung, durchschnittlich ca. 83%

allg. Fall: Split nur, wenn n Nachbarseiten voll sind

28

Optimale Seitengröße

Parameter k ?

Aufwand pro Seite * h

α: Zugriffszeit, unabhängig von k

β: Übertragungszeit für Tupel (x α p)

γ: Konstante für binäres Suchen in Seite

ν: Besetzungsfaktor für Seite: $1 \leq \nu \leq 2$

Kosten pro Seite : $\alpha + \beta\,(2k + 1) + \gamma \ln (\nu k + 1)$

δh: Seitentransporte pro Operation

Zeit $t \approx \delta h\,(\alpha + \beta\,(2k + 1) + \gamma \ln (\nu k + 1))$

29

approximiere: $h \approx \log_{\nu k+1} (I)$ (Höhenungleichung S.26)

$t \approx t_a = \delta \log_{\nu k+1} (I)\,(\alpha + \beta\,(2k + 1) + \gamma \ln (\nu k + 1))$

mit $\log_{\nu k+1} (I) = \frac{\ln (I)}{\ln (\nu k + 1)}$:

$$t_a = \delta \ln (I) \left[\frac{\alpha + \beta\,(2k + 1)}{\ln (\nu k + 1)} + \gamma \right]$$

$$t_a` = \delta \ln (I) \left[\frac{\ln (\nu k + 1) \cdot 2\beta - (\alpha + \beta\,(2k+1)) \frac{\nu}{\nu k + 1}}{\ln^2 (\nu k + 1)} \right]$$

30

t_a`= 0 liefert:

$$\ln(\nu k + 1) \cdot 2\beta - (\alpha + \beta(2k + 1)) \frac{\nu}{\nu k + 1} = 0$$

$$2 \frac{\nu k + 1}{\nu} \ln(\nu k + 1) = \frac{\alpha}{\beta} + (2k + 1) \qquad /: \beta$$

$$\frac{\alpha}{\beta} = 2 \frac{\nu k + 1}{\nu} \ln(\nu k + 1) - (2k + 1) =: f(k, \nu)$$

wähle k so, daß $f(k, \nu) \approx \frac{\alpha}{\beta}$

mit $\nu : 1 \leq \nu \leq 2$

31

Wichtig: optimale Seitengröße
unabhängig von I, γ, δ

f hat flachen Verlauf, d.h. Berücksichtigung
von Kenngrößen der Platte, z.B. Spurkapazität o. k.

Konkrete Blockgrößen:

Fall 1: (x, α, p) = 45 Bytes

$$\beta = \frac{45\ B \cdot sec}{3\ MB} = 15\ \mu s; \quad \alpha = 20\ ms$$

$$\frac{\alpha}{\beta} = \frac{20\ ms}{15\ \mu s} = 1.3 \cdot 10^3; \quad k \approx 150$$

Seitengröße = 13.5 KB

32

Fall 2: (x, α, p) = 300 Bytes

$\beta = \frac{300\ B \cdot s}{3\ MB} = 10^{-4}\ s;\ \frac{\alpha}{\beta} = 200$

$k \approx 32$ ***Seitengröße 9600 Bytes***

Fall 3: Vergleich Festplatte mit CD-ROM

20 ms	200 ms
3 MB/ s	300 KB/ s

$\frac{\alpha}{\beta}$ bleibt, gleiche Blockgröße

glücklicher Zufall ??

33

Fall 4: Vergleich Festplatte mit Juke-Box

20 ms	10s
3 MB/ s	300 kB/ s

$\frac{\alpha}{\beta}$ wird größer

$\frac{\alpha}{\beta} = \frac{10s}{150\ \mu s} = 6.6 \cdot 10^4;\ k \approx 4000$

Blockgröße: 8000 · 45 Bytes = ***360 KB***

34

Baumhöhe u. Indexelemente:

$2\,(k+1)^{h-1} - 1 \le \mathbf{I} \le (2k+1)^{h} - 1$

sei k = 200

h	I_{min}	≤	I	≤	I_{max}
1	1				400
2	400				160 000
3	$8 \cdot 10^4$				$400^3 = \mathbf{64 \cdot 10^6}$
4	$2 \cdot 200^3 = 1.6 \cdot 10^7$				$400^4 = 2.56 \cdot 10^{10}$

Anm: $\mathbf{64 \cdot 10^6} \cdot 20$ Bytes ≈ 1 GB

$\Rightarrow$ bei Workstations heute ca. 2 Plattenzugriffe

35

Kfz-Datei Flensburg: $< 50 \cdot 10^6$ Fahrer

$= 5 \cdot 10^7$ Einträge, d.h. h = 4,

mit großem Cache 2 Zugriffe

36