Kap. 3 Bäume

Kap. 3.1 Ungerichtete (nicht-orientierte) Bäume

Def.: Ein nicht-orientierter (n-o) Baum ist ein ungerichteter, zusammenhängender, zyklenfreier Graph.

Frage: Prüfalgorithmus?

Def.: (rekursiv) für n-o Baum

○ Knoten ist n-o Baum

verbinde 2 disjunkte Bäume durch neue Kante

1

seien $B_i = (K_i, \overline{E}_i)$, $i = 1, 2$ Bäume

mit $K_1 \cap K_2 = \varnothing$ $\{$ auch $\overline{E}_1 \cap \overline{E}_2 = \varnothing$ $\}$

und $k_i \in K_i$ beliebig, $i = 1, 2$

dann ist

$$B_{k_1, k_2} = (K, \overline{E}) \text{ mit}$$

$$K := K_1 \cup K_2$$

$$\overline{E} := \overline{E}_1 \cup \overline{E}_2 \cup \{\overline{(k_1, k_2)}\}$$

ein neuer n-o Baum

Hinweis: Konstruktionsvorgang für n-o Baum nicht eindeutig, weil Entfernung beliebiger Kante einen Baum in 2 Bäume zerlegt

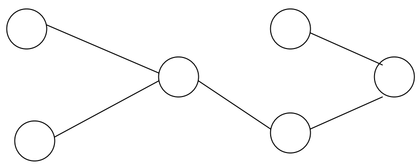
n! Möglichkeiten, denselben n-o Baum zu konstruieren

2

Gleichheit von n-o Bäumen?

$B_1 = (K_1, \overline{E}_1)$ $\quad\quad$ $B_2 = (K_2, \overline{E}_2)$

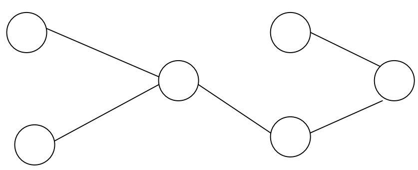
$B_1 = B_2 \;:\Leftrightarrow\; K_1 = K_2 \wedge \overline{E}_1 = \overline{E}_2$



3

Kap. 3.2 Orientierte Bäume

Forderung: gerichtete Kanten und Erreichbarkeit von Wurzeln aus, z.B.



Satz: Die Wahl eines Knotens w als Wurzel in einem n-o Baum bestimmt alle Kantenrichtungen eindeutig.

Ab jetzt: „Baum“ bedeutet „orientierter Baum“

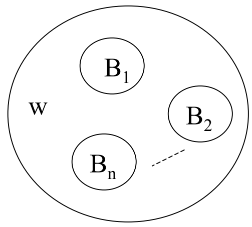
4

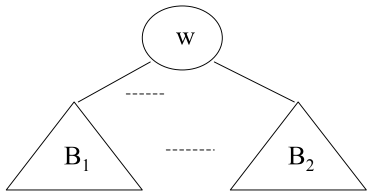
1. Def.: <u>X</u>-Baum über Objekten der Art <u>X</u> ist ein <u>X</u>-Objekt (Wurzel) mit $n \geq 0$ „Unterbäumen“.

Abbruch der Rekursion mit $n = 0$

Darstellungen:

$\{w, B_{1,} B_2 ..., B_n\}$
Reihenfolge unwichtig





Selektoren?

identische Unterbäume erlaubt, Bags!

5

Einschub: Bags über Art Y :
Y* mit Äquivalenzrelation ~ definiert durch:

$y = y_1 y_2 ... y_j \sim y_1` y_2` ... y_j` = y`$
$\Leftrightarrow y_1` y_2` ... y_j`$ ist Permutation von $y_1 y_2 ... y_j$

Zeige: ~ ist Äquivalenzrel. auf Y*

Bags sind Äquivalenzklassen: Y*/~

bzw. $\tilde{Y}$, Bags (Y) und $\tilde{y} \in \tilde{Y}$

Zeige: ~ ist Kongruenzrel. bzgl. **einfügen**, **entfernen** von $y_k \in$ Y

6

d.h. y, y` $\in$ Y* und y ~ y`

einfügen (y_k, y) ~ einfügen (y_k, y′)

entfernen (y_k, y) ~ entfernen (y_k, y′)

2. Def.: mit Selektor für Wurzel, ohne Selektoren für einzelne Unterbäume:

type o-Baum X = (Wurzel: **X,**
Wald: Bag (**o-Baum X**))

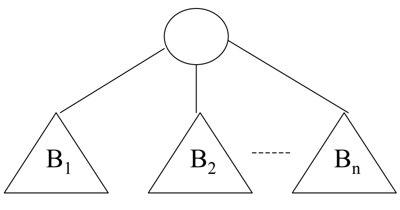
Rek: endet mit leerem Bag

Kap.3.3 Geordnete Bäume

Reihenfolge der Unterbäume wichtig, beliebigeAnzahl!

Type GBaum X = (Wurzel : **X,**
UB : **Gbaum* X**)

i.e. rek. Def. mit Abbruch, wenn **Gbaum* X** ist leere Folge ε



Reihenfolge der B_i wichtig!

(w, B_1, B_2, ..., B_n)

(w, ( ), ( ), ..., ( ))

Selektoren: abgekürzt, ohne UB:

var B : **Gbaum X**

B.Wurzel, B.1, B.2, ..., B.n

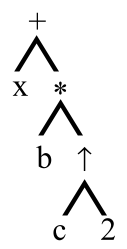
Beispiele für orientierte Bäume:

- Ahnen Stammbaum eines Menschen
- Nachkommenstammbaum
- Zusammensetzung von Maschine aus Einzelteilen
- Organigramm einer Firma
- Isa Hierarchie ohne multiple Vererbung, z.B. Tiere, Pflanzen

Beispiele für geordnete Bäume:

- Nachkommen in Geburtsreihenfolge
- Syntaxbaum
- Dokumentenstruktur
- Montage-Baum für Maschine
- alle Suchbäume der Informatik, z.B. AVL
- manche Pflanzen, z.B. Rosen
- Flüsse mit links vor rechts

$x + b * c \uparrow 2$



geordnete Bäume, weil nicht alle Operationen kommutativ sind, sonst orientierte verwendbar

Kap. 3.4 Binärbäume und m-Bäume

Def. Technik: Cartesisches Produkt plus Rekursion!

type BIN X = ∅ | (LB : **BIN X**;
Wurzel : **X**; LB, RB : **BIN X**)

type m-Baum X = ∅ |
(Wurzel : **X**; UB1, UB2, ..., Ubm : **m-Baum X**)

Begriffe: Knoten, Kante, Wurzel, interne Knoten, Zwischenknoten, Blatt, Pfad, ...
ß (e) : Beginn der Kante e
ε (e) : Ende der Kante e

e = (B.Wurzel, B.UBi.Wurzel)
e ist orientiert, gerichtet

11

Pfad : Kantenfolge $e_1, e_2, ..., e_p$
mit $\varepsilon(e_i) = ß(e_{i+1})$; $i = 1 ... p-1$
p ist Pfadlänge = Anzahl der Kanten

Def.: Baumhöhe h
Sei B : **m-Baum X**, p Länge eines längsten Pfades von B.Wurzel zu Blatt

$h \underset{def}{=} p+1$; $h(B) = p(B)+1$

$h(\emptyset) = 0$
$h(B) = 1 + \max\{h(B.LB), h(B.RB)\}$

Def.: B heißt ***vollständig***, wenn alle Pfade von B.Wurzel zu Blättern gleichlang sind und interne Knoten keine leeren Unterbäume haben

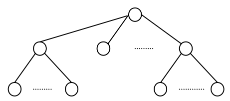
12

Lemma: Anzahl N(B) der Knoten in vollständigem m-Baum der Höhe h:

$$N = \frac{m^h - 1}{m - 1}$$

Bew:

Höhe h	N
1	$1 = m^0$
2	$m = m^1$
3	m^2
$\vdots$	$\vdots$
h	m^{h-1}



13

Lemma: $h \leq 1 + \log_m N$
für vollständigen m-Baum mit $N \geq 1$ Knoten

Bew: $m^h - 1 = N \cdot (m-1)$
$m^h = Nm - N + 1 \leq Nm$
$h \leq \log_m(Nm) = \log_m N + \log_m m$

Fazit: Höhe wächst sehr langsam bei vollständigen m-Bäumen, z.B. m = 100:

H	N
H = 1	N = 1
2	101
3	101001
4	$1010101 \sim 10^6$
5	$101010101 \sim 10^8$

14

Exponentielles Wachstum wird durch Baumorganisation beherrschbar!!

Hinweis: Baum Höhe 4, 8000 B/Knoten ~ 8GB
5 ~ 800 GB (Jukebox)

Pfad zu Blatt einer 8 GB Baum-Datei mit
3 Plattenzugriffen, falls Wurzel im Cache : **≤ 30 ms**

Durchlaufalgorithmen: für B: **BIN X**

Vorordnung: B. Wurzel
(preorder) B. LB in Vorordnung
B. RB in Vorordnung

15

procedure Vorordnung (B : **BIN X**);
if B ≠ ∅ **then**
begin suche B.Wurzel auf;
Vorordnung (B. LB);
Vorordnung (B. RB)
end

Warnung: Nie auf Leerheit der Unterbäume prüfen!!!

Vergleiche: Algorithmen-Struktur und rek. Def. von **BIN X**

analog: **Nachordnung:** LB; W; RB
(Inordnung)
Endordnung : LB; RB; W

Durchlaufalgorithmen für m-Bäume ?

16

Sortierte Binärbäume

Def.: B : BIN X heißt ***sortiert nach < auf X***, falls **Inordnung** die Knoteninhalte von B in < Reihenfolge liefert.

Anm.: Ersetzte < durch ≤, falls gleiche Inhalte für verschiedene Knoten erlaubt sind.

Kriterium für „sortiert“:

$\forall k, \forall l \in k.LB, \forall r \in k.RB : l \leq k \leq r$

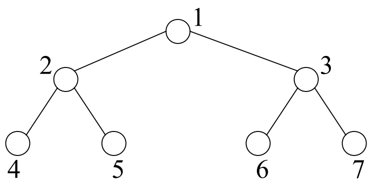
Klassen von internen Sortieralgorithmen:

Zeit/Platz	O(n)	≤ 1.5n	2n
$O(n^2)$			
O(n log n)			

Kap. 3.5 Darstellung von Binärbäumen

3.5.1 Schichtenweise sequentielle Darstellung

Vorauss.: gleicher Speicherbedarf pro Knoten z.B. 1 Zelle



Relativadressen =
Indizes eines **array** [1 : N]

Zeiger ersetzt durch Adressenrechnung!

Relativadr. eines Knotens Y.Wurzel : k

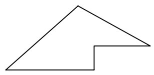
„ Y.LB.Wurzel : $2k$

„ Y.LB.Wurzel : $2k + 1$

„ Vater von Wurzel : $\left\lfloor \frac{k}{2} \right\rfloor$

Existenzbedingung: $1 \leq k \leq N$

Darstellung nur geeignet für fast vollständige Bäume:



Vorteil: keine Zeiger!

Verallg. Auf m-Bäume: Induktionsbew.

Y. Wurzel : k
p-ter Nachf.: $m \cdot (k-1) + p + 1$
Vorgänger: $\left\lceil \frac{k-1}{m} \right\rceil$

19

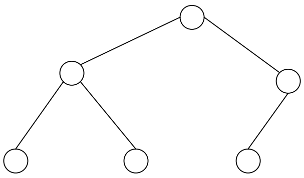
Verwendung: Heapsort, etc.
Vergleiche schichtenweise sequentielle Darstellung mit sortiertem Feld und Bisektion
Problem ist Invarianz der Wurzelposition bei Einfügungen und Löschungen

Operationen:

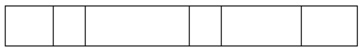
- tausche Vater ↔ Sohn
- anfügen, löschen am Ende : O (log n)
- ersetzen der Wurzel
  Speicher wächst, schrumpft kellerartig

20

Baum von Zeigern mit schichtenweise sequentieller Darstellung
bei unterschiedlichem Platzbedarf pro Knoten.



Zeiger fester
Länge auf Datensätze!



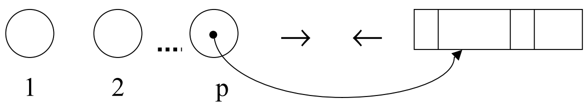
Datensätze kompakt

21

- Knoteninhalte variabler Länge,
  Zeiger fester Länge
- Tausch Vater ↔ Sohn ~ Zeigertausch
- anfügen am Ende ✔

? Löschung braucht Speicherverwaltung bei Heapsort?

Speicherorg. Als gegenläufige Keller:



22

3.5.2 Geflechte von Knoten und Zeigern

a) $\longrightarrow$ (↑LB, Wurzel, ↑RB)

hat i.a. variable Länge, auf Halde!

b) Zeigertripel fester Länge

$\longrightarrow$ (↑LB, ↑Wurzel, ↑RB)

ermöglicht Sortieralgorithmen mit Tausch-
operation!
Frage: Sortieren ohne Tauschoperation?

23

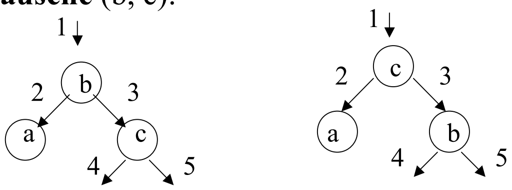
Wichtigkeit der Tausch-Operation

$\left.\begin{array}{l}\text{einsortieren}\\ \text{finden}\\ \text{entfernen}\end{array}\right\}$ Aufwand O(h(B)) mit Elementar-Operationen wie vergleiche, tausche, move

Vergleiche, move ~ Länge der Objekte
tausche ~ Länge der Objekte **plus** Speicherverwaltung

24

tausche (b, c):

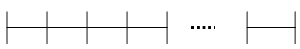


Aufwand: ändern von 5 Zeigern = O(1)
wenn b, c gefunden sind.
wichtig für AVL-Bäume!

25

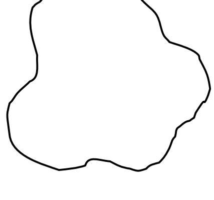
Kap. 3.6 Wachstum und Bäume

Lineares Wachstum in Zeit:



z.B. Körpergröße
$c \cdot t^1$

Quadratisches Wachstum:



$c_1 t \cdot c_2 t = dt^2$
z.B. Fläche eines Ölteppichs,
Waldbrand, ...

26

Polynomiales Wachstum: n Dimensionen

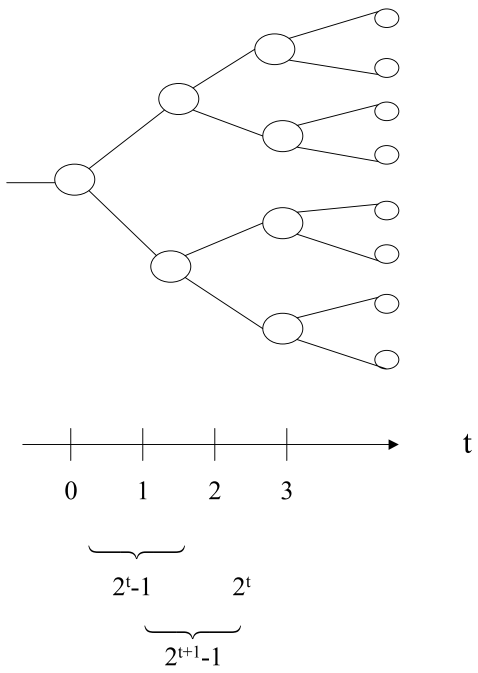
$c_1 t \ * \ c_2 t \ldots * c_n t = c_1 c_2 \ldots c_n t^n = dt^n$

mit dimensionsabhängigen Wachstumskonstanten,
z.B. Volumen eines Baumstamms.

Exponentielles Wachstum
Verzweigung
Autonomie (Rekursion)

Parallelität

27



2^t Knoten
≈ Pfadlänge t

28

Naturprinzip: Je schlechter die Überlebenschancen,
desto höher der Verzweigungsgrad (z.B. Löwenzahn und Mensch)
Bedingung für expon. Wachstum :
Anzahl Nachkommen > 1

Fundamentale Beobachtung:
Exponentielles Wachstum wird durch Baumstrukturen beherrschbar!!!

29