ADS: Algorithmen und Datenstrukturen

Teil VII

Peter F. Stadler & Konstantin Klemm

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

08. Dezember 2010

Speicherung in Binärbäumen

Vereinbarung:

- In jedem Knoten des Baumes wird genau ein Schlüssel gespeichert. Der Baum enthält also so viele Knoten wie Schlüssel.

Anordnung der Schlüssel in binären Suchbäumen (kommen später)

- Für jeden Knoten gilt: Die Schlüssel im linken Teilbaum sind sämtlich kleiner als der in der Wurzel und dieser ist wiederum kleiner als die Schlüssel im rechten Teilbaum.

Speicherung von Binärbäumen

(1) Verkettete Speicherung: Freispeicherverwaltung der Struktur wird von der Speicherverwaltung des Programmiersystems übernommen.

(2) Feldbaum-Realisierung: Simulation einer dynamischen Struktur in einem statischen Feld
Eigenschaften:
- statische Speicherplatzzuordnung
- explizite Freispeicherverwaltung

Beispiele: Tafel

Sequenzielle Speicherung von Binärbäumen

(3) Sequentielle Speicherung: Methode kommt ohne explizite Verweise aus. Für fast vollständige oder zumindest ausgeglichene Binärbäume bietet sie eine sehr elegante und effiziente Darstellungsform an.

Ein fast vollständiger Baum mit n Knoten wird sequentiell nach folgendem Numerierungsschema gespeichert. Für jeden Knoten mit Index i, $1 \leq i \leq n$, gilt:

- Vater(i) hat Nummer $\lfloor i/2 \rfloor$ für $i > 1$
- Lsohn(i) hat Nummer $2i$ für $2i \leq n$
- Rsohn(i) hat Nummer $2i + 1$ für $2i + 1 \leq n$

Durchlaufen eines Binärbaums

Baumdurchlauf (Traversierung) = Verarbeitung aller gespeicherten Schlüssel in einer Reihenfolge, die durch die Baumstruktur gegeben ist.

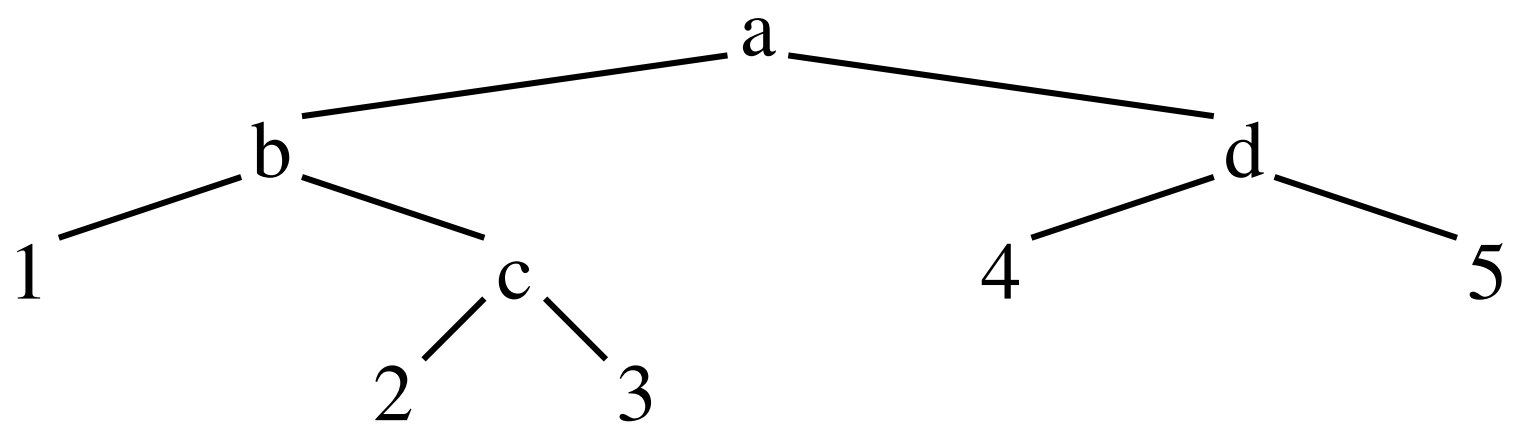
Rekursiv anzuwendende Schritte

- Verarbeite Wurzel: W
- Durchlaufe linken UB: L
- Durchlaufe rechten UB: R

Durchlaufprinzip impliziert sequentielle, lineare Ordnung auf der Menge der Knoten
Es gibt 6 Möglichkeiten, W, L und R anzuordnen, aber
Konvention: linker UB vor rechtem UB
Verbleibende 3 Möglichkeiten:

- Vorordnung (preorder): WLR
- Zwischenordnung (inorder): LWR
- Nachordnung (postorder): LRW

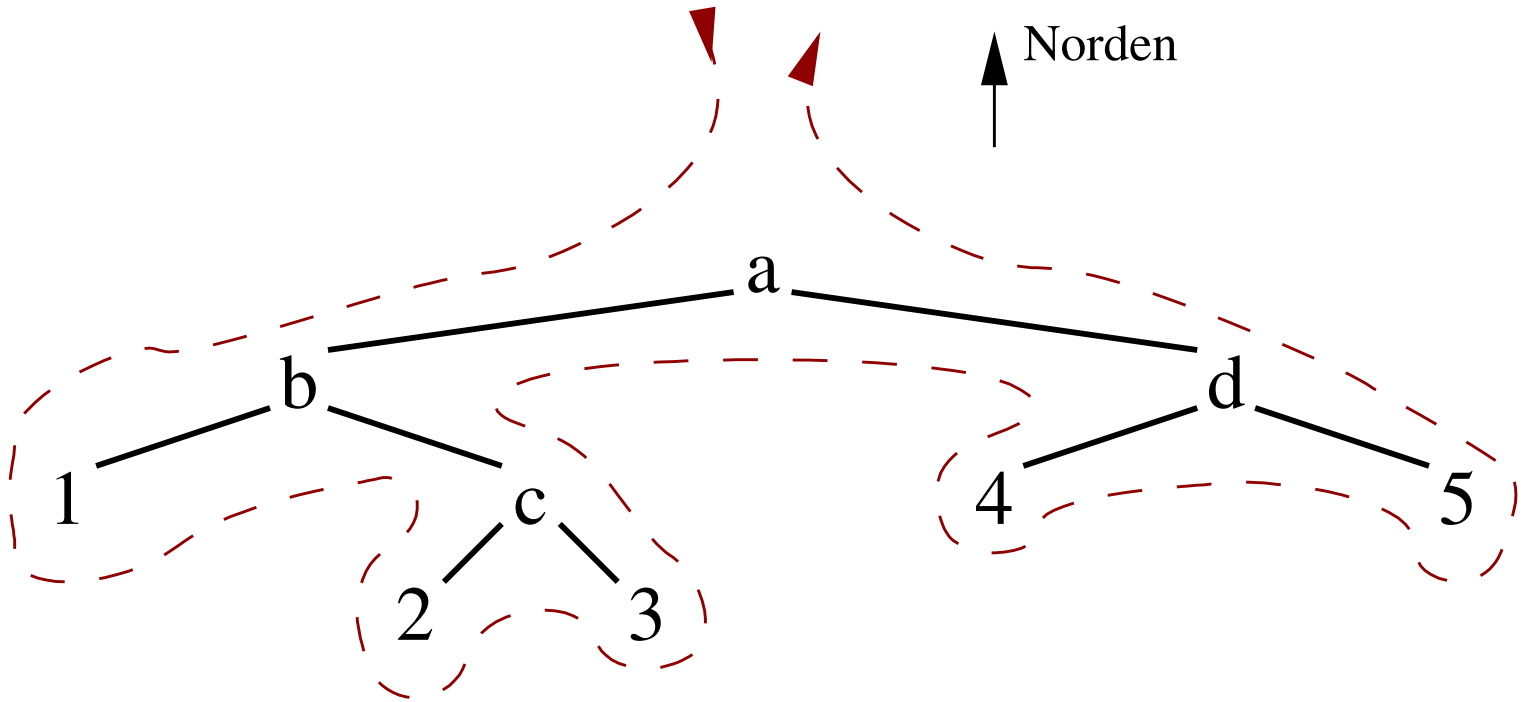
Durchlaufmöglichkeiten I



Vorordnung, preorder, WLR:	a	b	1	c	2	3	d	4	5
Zwischenordnung, inorder, LWR:	1	b	2	c	3	a	4	d	5
Nachordnung postorder, LRW:	1	2	3	c	b	4	5	d	a

Durchlauf: Anschauliche Darstellung

"Wanderung" um den Baum, Knoten werden aufgerufen bei bestimmter relativer Bewegung:



bei Passage Richtung Süden, WLR:	a	b	1	c	2	3	d	4	5
an Südseite, inorder, LWR:	1	b	2	c	3	a	4	d	5
bei Passage Richtung Norden, LRW:	1	2	3	c	b	4	5	d	a

Rekursive Realisierung

Rekursive Version für Inorder-Traversierung (LWR)

```
DurchlaufInOrder(Baum)
  Falls Baum leer, dann fertig
  sonst {
    DurchlaufInOrder(LinkeTochter(Baum))
    Drucke Inhalt des aktuellen Knotens
    DurchlaufInOrder(RechteTochter(Baum)) }
```

Iterative Realisierung

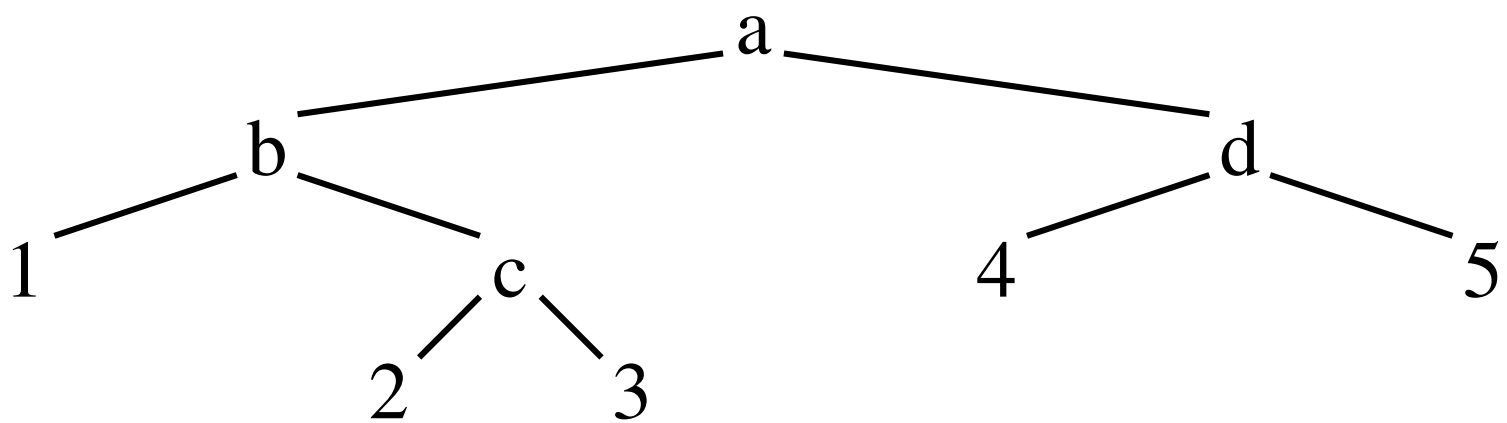
Iterative Version von LWR (inorder)

Ziel: effizientere Ausführung durch eigene Stapelverwaltung

Vorgehensweise:

- Nimm, solange wie möglich, linke Abzweigung und speichere die Knoteninhalte des zurückgelegten Weges auf einem Stapel (Aktion 1).
- Wenn es links nicht mehr weitergeht, wird der oberste Knoten des Stapels ausgegeben und vom Stapel entfernt. Der Durchlauf wird mit dem rechten Unterbaum des entfernten Knotens fortgesetzt (Aktion 2).

Beispiel für iteratives LWR



Gefädelte Binärbäume I

Ziel: Weitere Verbesserung von iterativen Durchlaufalgorithmen

Problem: Es gibt in vielen Fällen keinen Zeiger auf den Nachfolger oder Vorgänger einer Traversierung im Baum

Methode benutzt einen Faden", der die Baumknoten in der Folge der Durchlaufordnung verknüpft. Zwei Typen von Fäden:

- *Rechtsfaden* verbindet jeden Knoten mit seinem Nachfolgerknoten in Durchlaufordnung
- *Linksfaden* stellt Verbindung zum Vorgängerknoten in Durchlaufordnung her.

Lösung 1: Explizite Speicherung von 2 Fäden

Gefädelte Binärbäume II

Lösung 2: Vermeidung von Redundanz

Eine zweite Art der Fädelung kommt ohne zusätzliche Zeiger aus und erfordert daher geringeren Speicherplatzaufwand. Die Algorithmen werden lediglich geringfügig komplexer.

Beobachtung 1: Binärbaum mit n Knoten hat n+1 freie Zeiger (null)

Beobachtung 2: für die Zwischenordnung können Fadenzeiger in inneren Knoten durch Folgen von Baumzeigern ersetzt werden

Idee: Benutze freie Zeiger und Baumzeiger für Fädelung

- pro Knoten zusätzliche Variablen Lfaden, Rfaden statt Lchild, Rchild
- zeigen auf linken bzw. rechten Nachbarn in Durchlaufreihenfolge.
- Achtung: Normale Baumzeiger müssen von Fädelzeigern unterschieden werden.

Gefädelte Binärbäume III

Algorithmus für die Inorder-Traversierung

```
Start bei Wurzelknoten
Schleife bis der Knoten rechts außen erreicht ist:
  Solange wie möglich nach links verzweigen
  Knoten ausgeben
  Falls Knoten Rfaden hat:
    Rfaden einen Schritt folgen
    Knoten ausgeben
  Sonst falls Knoten rechten Sohn hat:
    einen Schritt nach rechts verzweigen
```

Zusammenfassung

Definitionen

- Baum, orientierter Baum (Wurzel-Baum), Binärbaum
- vollständiger, fast vollständiger, strikter, ausgeglichener Binärbaum
- Höhe, Grad, Stufe / Pfadlänge

Speicherung von Binärbäumen

- verkettete Speicherung
- Feldbaum-Realisierung
- sequentielle Speicherung

Baum-Traversierung

- Preorder (WLR): Vorordnung
- Inorder (LWR): Zwischenordnung
- Postorder (LRW): Nachordnung

Gefädelte Binärbäume: Unterstützung der (iterativen) Baum-Traversierung durch Links/Rechts-Zeiger auf Vorgänger/Nachfolger in Traversierungsreihenfolge.