Univ.-Prof. Dr. Hermann Gehring
unter Mitarbeit von:
Dr. Andreas Bortfeldt

Modul 32561
Algorithmen und Datenstrukturen

Kurs 00814
Kurseinheit 3:
Grundlegende Algorithmen Baumstrukturen

LESEPROBE

Fakultät für
Wirtschafts-
wissenschaft

FernUniversität in Hagen

5 Baumstrukturen

5.1 Nichtlineare Datenstrukturen und Bäume

Als nichtlinear bezeichnet man sämtliche Datenstrukturen, die nicht die Eigenschaft der Linearität (vgl. hierzu Kap. 3) besitzen. Zu den nichtlinearen Datenstrukturen gehören insbesondere:

nichtlineare Datenstrukturen

- graphenartige Strukturen (Graphen),
- baumartige Strukturen (Bäume).

Bäume sind spezielle Graphen. Auf den Unterschied von Bäumen und Graphen wird unten noch kurz eingegangen.

In der betrieblichen Datenverarbeitung besitzen Bäume eine ungleich höhere Bedeutung als Graphen. Die folgenden Ausführungen beschränken sich deshalb auf Bäume.

Jedem Baum kann eine ganzzahlige Ordnung k, $k \geq 2$, zugeordnet werden. Ein k-närer Baum, d.h. ein Baum der Ordnung k, lässt sich wie folgt definieren:

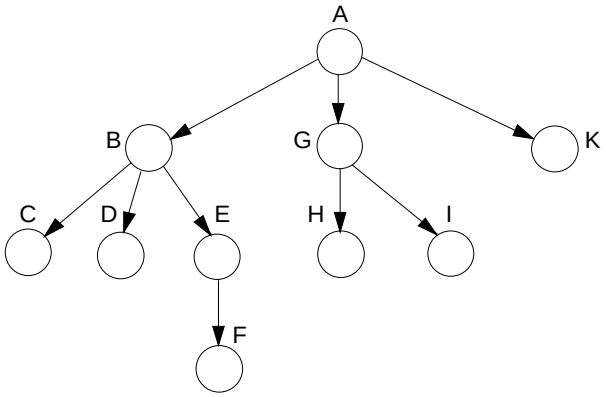
k-närer Baum

> Unter einem k-nären Baum versteht man eine Menge von Knoten und eine Menge von gerichteten, die Knoten verbindenden Kanten. Führt eine Kante von dem Knoten A zum Knoten B, so heißt A Vorgänger von B und B Nachfolger von A. Für die Anordnung der Knoten und Kanten gilt:
>
> - Es existiert ein ausgezeichneter Knoten, genannt Wurzel, der keinen Vorgänger und maximal k Nachfolger besitzt.
> - Ausgezeichnete Knoten sind außerdem die sogenannten Blätter; ein Blatt besitzt genau einen Vorgänger und keinen Nachfolger.
> - Die übrigen Knoten heißen innere Knoten; ein innerer Knoten besitzt genau einen Vorgänger und mindestens einen sowie maximal k Nachfolger.
>
> Da Wurzel und innere Knoten maximal k Nachfolgerknoten besitzen, spricht man von einem k-nären Baum oder von einem Baum der Ordnung k.

Aus der Definition resultiert folgende charakteristische Eigenschaft k-närer Bäume: Jeder von der Wurzel verschiedene Knoten kann durch genau eine von der Wurzel ausgehende endliche Folge von Kanten erreicht werden.

Auch Graphen als allgemeinere nichtlineare Datenstrukturen enthalten Knoten und Kanten, welche die Knoten verbinden. Anders als bei Bäumen können jedoch Knoten auch durch mehrere, unterschiedliche Kantenfolgen verbunden sein. Auf die Behandlung weiterer Unterschiede von Graphen und Bäumen sei an dieser Stelle verzichtet.

Ein Beispiel für einen k-nären Baum, $k = 3$, zeigt die Abb. 5.1. Die Knoten sind mit Großbuchstaben bezeichnet. A ist die Wurzel, die Knoten B, E, G sind innere Knoten und die übrigen Knoten stellen Blätter dar.



k-närer Baum
***k* = 3**

Abb. 5.1. Beispiel für einen k-nären Baum (*k* = 3).

Ein binärer Baum ist ein spezieller k-närer Baum, nämlich ein k-närer Baum der Ordnung *k* = 2. Jeder k-näre Baum (k > 2) ist auch als Binärbaum darstellbar. Daher stehen im Folgenden Binärbäume im Vordergrund. Die Definition eines binären Baums sei nochmals separat aufgeführt.

binärer Baum

Ein binärer Baum besteht aus einer Menge von Knoten und einer Menge von gerichteten, die Knoten verbindenden Kanten, für deren Anordnung gilt:

- Es existiert ein ausgezeichneter Knoten, die Wurzel, der keinen Vorgänger und maximal zwei Nachfolger besitzt.
- Ausgezeichnete Knoten sind außerdem die Blätter; ein Blatt besitzt genau einen Vorgänger und keinen Nachfolger.
- Die übrigen Knoten, genannt innere Knoten, besitzen je einen Vorgänger und einen oder zwei Nachfolger.

In Abb. 5.2 ist ein Beispiel für einen binären Baum zu sehen. Der Knoten A ist die Wurzel, die Knoten E, G, I, K, M sind Blätter und die restlichen Knoten stellen innere Knoten dar.

Beispiel für binären Baum

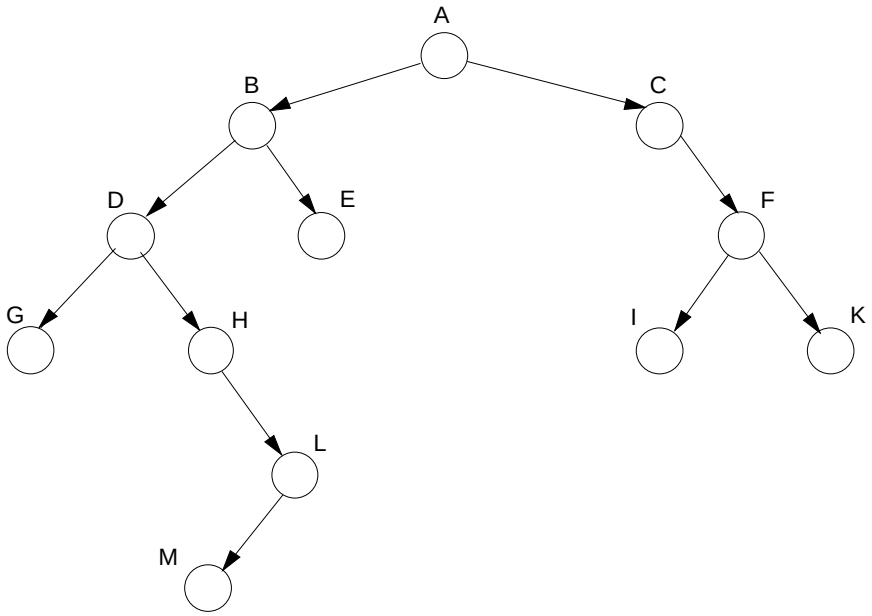


Abb. 5.2. Beispiel für einen binären Baum.

geordnete k-näre Bäume

Die beiden in Abb. 5.1 und Abb. 5.2 angegebenen Bäume nennt man geordnete k-näre Bäume, da die Knoten in einer bestimmten Reihenfolge notiert werden. Zur Beschreibung der Ordnung von Bäumen seien einige Begriffe eingeführt.

- Die Nachfolger eines gegebenen Knotens bezeichnet man auch als seine Söhne, den Knoten selbst als Vater seiner Söhne. In Abb. 5.1 gehören beispielsweise zu dem Vater B die Söhne C, D und E.

Brüder eines Knotens

- Die Söhne eines Vaters heißen, falls man auf die Beziehung zwischen den Söhnen abstellt, Brüder. Beispielsweise sind die Knoten B, G und K in Abb. 5.1 Brüder.

Niveau eines Baums

- Alle Knoten, die über die gleiche Anzahl von Kanten mit der Wurzel verbunden sind, liegen auf einem Niveau oder einer Ebene. Die Anzahl der Kanten charakterisiert eine Ebene. So liegen beispielsweise die Knoten C, D, E, H und I in Abb. 5.1 auf der 2. Ebene, während der Wurzelknoten A die 0. Ebene definiert.
- Mittelbare Nachfolger eines Knotens X sind alle Knoten Y, die von X aus durch eine Folge aus mindestens zwei Kanten erreichbar sind. So ist etwa in Abb. 5.1 F ein mittelbarer Nachfolger von B. Entsprechend werden mittelbare Vorgänger definiert.

Für die Ordnung des k-nären Baums in Abb. 5.1 gilt:

Regeln für die Notation eines k-nären Baums

(1) Die Notation beginnt bei dem Wurzelknoten und erfolgt in alphabetischer Reihenfolge.
(2) Söhne kommen vor Brüdern.
(3) Brüder werden von links nach rechts notiert.

Gemäß Regel (1) wird zunächst die Wurzel mit A bezeichnet. Es folgt – nach Regel (2) – der Übergang zu den Söhnen, von denen der linke entsprechend Regel (3) mit B zu benennen ist. Nun greift Regel (2) und es wird in die 2. Ebene zu den Söhnen von B übergegangen. Von denen ist der linke mit C zu benennen. Da C keine Söhne besitzt, ist Regel (3) anzuwenden und der rechts neben C gelegene Bruder mit D zu bezeichnen usw.

Regeln für die Notation eines binären Baums

Übungsaufgabe 5.1

Geben Sie die Regeln an, welche die Ordnung des in Abb. 5.2 dargestellten binären Baums festlegen.

In Abb. 5.3 sind zwei strukturgleiche, jedoch verschieden geordnete binäre Bäume dargestellt. Sie unterscheiden sich in der Reihenfolge der Notation der Knoten.

unterschiedlich geordnete binäre Bäume

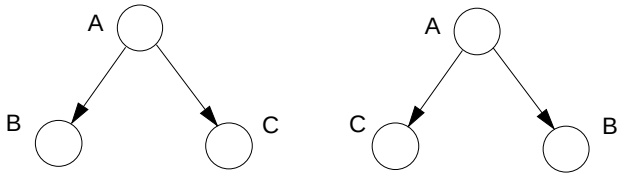
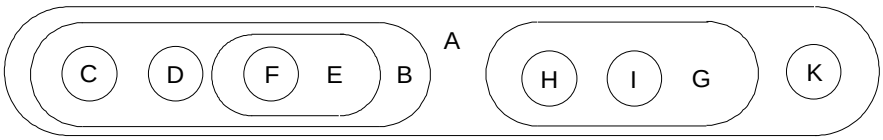


Abb. 5.3. Zwei verschieden geordnete binäre Bäume.

Neben dem bislang verwendeten Baumdiagramm eignen sich noch andere Darstellungsformen zur Wiedergabe von Bäumen. Nämlich geschachtelte Mengendarstellungen, geschachtelte Klammerdarstellungen und Balkendiagramme. Diese Darstellungsformen sind für den in Abb. 5.1 gezeigten k-nären Baum in Abb. 5.4

angegeben. Die Analogie zum Baumdiagramm ist evident und bedarf keiner weiteren Kommentierung.



a) Geschachteltes Mengendiagramm

Darstellungsformen für Bäume

(A (B (C)(D)(E (F)))(G (H)(I))(K))

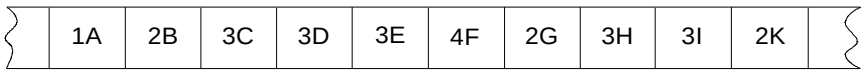
b) Geschachtelter Klammerausdruck

c) Balkendiagramm

Abb. 5.4. Darstellungsformen für k-näre geordnete Bäume.

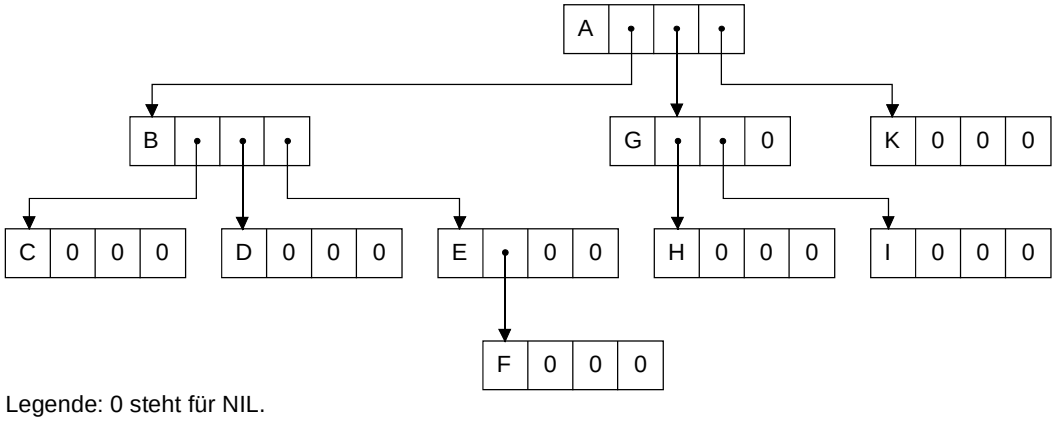
Geordnete k-näre Bäume können sowohl sequentiell als auch verkettet gespeichert werden. Bei der sequentiellen Speicherung fasst man Niveau und Bezeichnung eines Knotens zu einem Datenobjekt zusammen und legt die so gebildeten Objekte entsprechend ihrer Ordnung in einer linearen Liste ab. Bei der verketteten Speicherung stellt man die Ordnung zwischen den Knoten mit Hilfe von Zeigern her. In ein Datenobjekt sind folglich die Knotenbezeichnung und *k* Zeiger einzubeziehen. Abb. 5.5 veranschaulicht die sequentielle und die verkettete Speicherungsform für den in Abb. 5.1 dargestellten k-nären geordneten Baum.

sequentielle und verkettete Speicherung eines geordneten Baums



a) Sequentielle Speicherung

Speicherungsformen eines k-nären Baums



Legende: 0 steht für NIL.

b) Verkettete Speicherung

Abb. 5.5. Sequentielle und verkettete Speicherung eines k-nären geordneten Baums ($k = 3$).

Übungsaufgabe 5.2

Betrachtet werde die in Abb. 5.5 a) angegebene sequentielle Speicherung eines geordneten k-nären Baums. Dieser Darstellung lässt sich beispielsweise entnehmen, dass die Knoten C, D und E Söhne des Knotens B sind. Begründen Sie diese Aussage.

Interpretation von Knoten und Kanten

Auf die Bedeutung bzw. die inhaltliche Interpretation der Knoten und Kanten k-närer Bäume wurde bislang nicht explizit eingegangen. Hierzu sei folgendes bemerkt:

- Knoten repräsentieren Datenobjekte beliebiger Komplexität.
- Kanten definieren Beziehungen zwischen den Datenobjekten und zwar typischerweise fachlich-logisch ausgewiesene hierarchische Beziehungen.

Operationen auf geordneten Bäumen betreffen sowohl Knoten als auch Kanten. Die Bearbeitung von Objekten lokalisiert sich auf die Knoten. Dagegen spielen die in den Kanten steckenden Informationen beim Aufsuchen von Knoten und beim Zugreifen zu den sich dahinter verbergenden Objekten eine Rolle. Typische Operationen auf geordneten Bäumen sind u.a.:

Operationen auf geordneten Bäumen

(1) Aufsuchen eines Knotens bzw. Datenobjektes mit gegebenem (Primär-) Schlüsselwert.
(2) Aufsuchen aller Knoten bzw. Datenobjekte in einer bestimmten Reihenfolge.
(3) Zugreifen auf den Vorgänger eines gegebenen Knotens.
(4) Zugreifen auf einen bestimmten Nachfolger eines gegebenen Knotens.
(5) Aufsuchen aller Nachfolger eines gegebenen Knotens.
(6) Einfügen eines Knotens oder eines Teilbaums in den Baum.
(7) Entfernen eines Knotens oder eines Teilbaums aus dem Baum.

Übersicht über die folgenden Kapitel

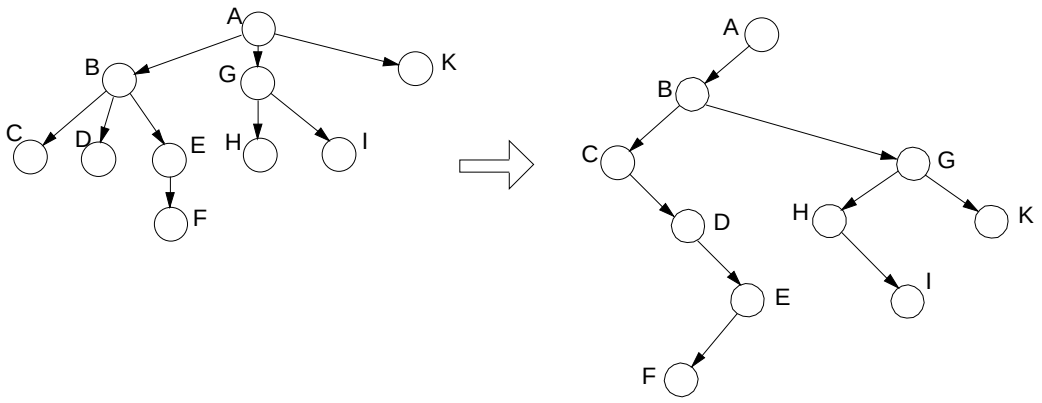
Die folgenden Kapitel beschäftigen sich mit diesen Operationen. Gegenstand des Kap. 5.3 ist das Traversieren von Bäumen, d.h. das Aufsuchen aller Knoten in einer bestimmten Reihenfolge. In Kap. 5.4 stehen Such- und Update-Operationen auf Bäumen und dafür besonders geeignete Baumstrukturen, nämlich Suchbäume, im Vordergrund. Zunächst werden jedoch in Kap. 5.2 alternative Darstellungen von Binärbäumen betrachtet. Kap. 5.2 beschränkt sich, wie die Kapitel danach, auf Binärbäume, da jeder k-näre Baum in einen Binärbaum umgewandelt werden kann und da Betrachtungen, die für Binärbäume gelten, auf k-näre Bäume übertragbar sind. Zur Umwandlung eines k-nären Baums in einen Binärbaum gilt:

Umwandlung eines k-nären Baums in einen binären Baum

Bezeichne n einen Knoten eines k-nären Baums mit den k Nachfolgern n_1, ..., n_k, so entsteht aus dem k-nären Baum auf folgende Weise ein Binärbaum:

- An den Knoten n wird im Binärbaum der Knoten n_1 als linker Nachfolger angehängt.
- Die übrigen Knoten n_i, $i = 2, ..., k$, werden im Binärbaum jeweils als rechte Nachfolger an die Knoten n_{i-1} angehängt; auf den Knoten n_1 folgt also n_2 als rechter Nachfolger, auf n_2 der Knoten n_3 usw.

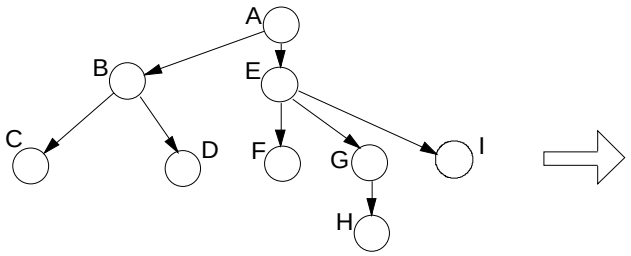
Angewandt auf den k-nären Baum in Abb. 5.1 ergibt sich folgender Binärbaum:



Man beachte, dass sich die gleiche Reihenfolge der Notation der Knoten ergibt, falls man die Ordnung des k-nären Baums auf den Binärbaum überträgt. Die Vorschrift zur Umwandlung erhält also die Ordnung.

Übungsaufgabe 5.3

Wandeln Sie den nachfolgend angegebenen k-nären, $k = 3$, Baum in einen Binärbaum um.



5.2 Darstellung von Binärbäumen

Darstellung von Binärbäumen durch

Die Darstellung eines Binärbaums beeinflusst unmittelbar die Manipulation dieser Datenstruktur. Im Einzelnen umfasst die Darstellung folgende Spezifikationen und Festlegungen:

Festlegung der Wurzel

(1) Spezifikation der Wurzel, die in der Regel als Einstieg in einen Binärbaum dient.

der linken und rechten Nachfolger und der Speicherungsform

(2) Spezifikation der Struktur durch Angabe der linken und rechten Nachfolger sämtlicher Knoten.

(3) Festlegung der Darstellungsform; an Alternativen stehen zur Verfügung:
   - geflechtartige Darstellung mit Hilfe von Zeigervariablen,
   - schichtenweise, sequentielle Darstellung mit niveauweiser Nummerierung,
   - Darstellung mit nicht niveauweiser Nummerierung.

Die drei genannten Darstellungsweisen werden nachfolgend präzisiert.

5.2.1 Geflechtartige Darstellung mit Hilfe von Zeigervariablen

Nachfolgerbeziehung als Zeigervariable

In diesem Fall enthalten die den Knoten zugeordneten Datenobjekte zwei Zeigervariablen, die auf den linken und auf den rechten Nachfolger verweisen. Treten Nachfolger nicht auf, so erhalten die entsprechenden Zeiger den Wert NIL. Insgesamt entsteht auf diese Weise eine dynamisch abgespeicherte Datenstruktur.

Als Beispiel sei die Ablage von Personalsätzen in Form eines binären Baums betrachtet. Der Inhalt der Sätze ist lediglich angedeutet. Die benötigten Datendefinitionen lauten:

Datendefinition für einen binären Baum

```
DATA
    TYPE
        PERSDATA = RECORD
                       pnr : ARRAY [1..8] OF CHAR;
                       pname : ARRAY [1..20] OF CHAR;
                       geb_datum : ARRAY [1..8] OF CHAR;
                       END;
        PERSLINK = ↑PERSON;
        PERSON   = RECORD
                       pers : PERSDATA;
                       links, rechts : PERSLINK;
                       END;
    VARIABLE
        wurzel, link : PERSLINK;
        person : PERSON;
        persdata : PERSDATA;
```

Zeiger auf linken und rechten Nachfolger

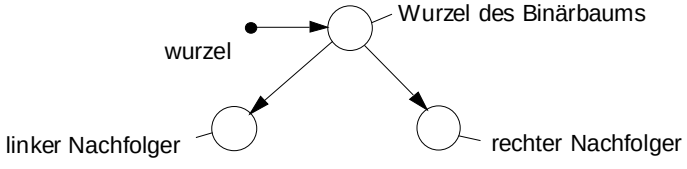
Ein Personalsatz vom Typ PERSON enthält neben der Personalnummer *pnr*, dem Namen *pname* und weiteren Nutzdaten zwei Zeiger, genannt *links* und *rechts*. *links* verweist auf den linken und *rechts* auf den rechten Nachfolger. Die Zeigervariable *wurzel* ermöglicht den Einstieg in den Binärbaum. Sie ist beim Aufbau des Baums geeignet zu initialisieren. *link* ist eine Hilfsvariable, und die Variable *person* ermöglicht die Manipulation von Personalsätzen. Die Variable *persdata* dient der Behandlung nicht zeigerbezogener Personaldaten.

An einem Beispiel sei der Aufbau eines Binärbaums mit Hilfe dieser Darstellungsform demonstriert. Da er nur grundsätzliche Zusammenhänge aufzeigen soll, wurde er bewusst sehr einfach gehalten.

einfacher Binärbaum

Beispiel 5.1

Es soll ein aus drei Knoten bestehender Binärbaum der folgenden Struktur aufgebaut werden:



Jeder der drei Knoten repräsentiert einen Personalsatz des oben definierten Typs. Die Sätze sind von einem externen Medium einzulesen und – in der Reihenfolge des Einlesens – der Wurzel, dem linken Nachfolger und schließlich dem rechten Nachfolger zuzuweisen.

Erzeugen eines einfachen Binärbaums

Dies leistet beispielsweise nachstehende Anweisungsfolge:

```
NEW(wurzel);                    {Speicherplatz für Wurzelknoten}
INPUT(persdata);                {ersten Satz einlesen}
wurzel↑.pers := persdata;       {Wurzel ist erster Satz}
NEW(link);                      {Bereitstellen eines neuen Knoten}
INPUT(persdata);                {Zweiter Satz einlesen}
link↑.pers := persdata;
link↑.links := NIL;             {kein linker Nachfolger}
link↑.rechts := NIL;            {kein rechter Nachfrager}
wurzel↑.links := link;          {Verbindung mit Wurzel}
NEW(link);
INPUT(persdata);                {dritten Satz einlesen}
link↑.pers := persdata;
link↑.links := NIL;             {kein linker Nachfolger}
link↑.rechts := NIL;            {kein rechter Nachfolger}
wurzel↑.rechts := link;         {Verbindung mit Wurzel}
```

Nach dem Erzeugen eines dynamischen Objekts, auf das die Zeigervariable *wurzel* verweist, werden die Personaldaten des ersten Satzes in dieses Objekt geschrieben. Erst nach dem Erzeugen des linken bzw. rechten Nachfolgers einschließlich der Zuweisung der Personaldaten kann jeweils die Verbindung mit der Wurzel hergestellt werden.

Übungsaufgabe 5.4

effizienter Aufbau eines Binärbaums

Bei der Anweisungsfolge in Beispiel 5.1 wirkt die etwas stupide Wiederholung gleicher Anweisungen (Kreieren eines dynamischen Objekts, Einlesen eines Personalsatzes und Zuweisen von Daten) wenig elegant und aufwendig. Klar dürfte auch sein, dass man einen größeren Binärbaum nicht mit einem derartigen linearen Algorithmus aufbauen wird. Entwickeln Sie einige Ideen, die man einem effizienten Algorithmus zugrunde legen könnte.

5.2.2 Schichtenweise sequentielle Darstellung mit niveauweiser Nummerierung

Abspeicherung des vollständigen Binärbaums

Eine bezüglich des Laufzeitverhaltens effiziente Datenstruktur, die kein Zeigerkonzept voraussetzt, ist die schichtenweise sequentielle Darstellung mit niveauweiser Nummerierung. Bei dieser Darstellungsform wird stets ein vollständiger Binärbaum gespeichert. Der Baum enthält somit besetzte Knoten und gegebenenfalls auch nicht besetzte Knoten bzw. Dummy-Knoten. Dummies werden nicht mit Werten belegt. Die Anzahl der Knoten einschließlich Dummies beträgt 2^n - 1,

n = 1, 2, 3, 4,... Für n = 1 liegt ein degenerierter, nur aus der Wurzel bestehender Baum vor. Für n = 4 ergibt sich der in Abb. 5.6 skizzierte Baum.

Nummerierung von links nach rechts, niveauweise

Die Knoten des Binärbaums in Abb. 5.6 sind niveauweise nummeriert. Ausgehend von der Wurzel bzw. der Ebene Null und fortschreitend zu den Ebenen 1, 2 usw. erhalten die Knoten fortlaufende Nummern. Innerhalb einer Ebene wird von links nach rechts nummeriert. Nummeriert man die Knoten auf diese Weise, so werden die Kanten implizit mitberücksichtigt. Mit Hilfe von Adressrechnungen ist es nämlich jederzeit möglich, den Vorgänger eines Knotens oder seine Nachfolger sowie die Wurzel zu ermitteln. Einer Auswertung von Kanteninformationen bedarf es daher nicht.

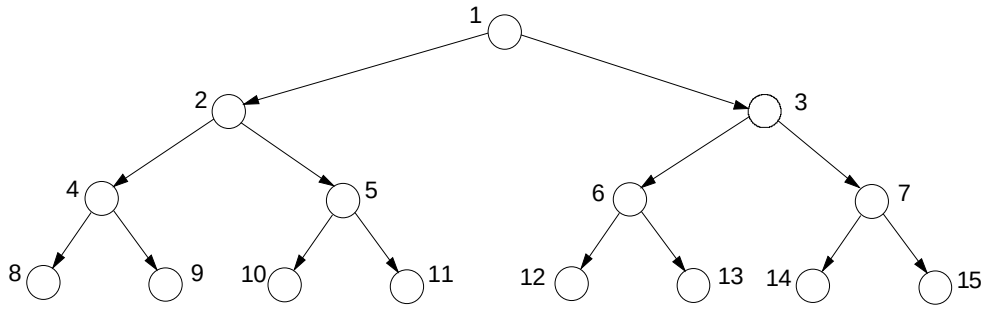


Abb. 5.6. Vollständiger Binärbaum mit niveauweiser Nummerierung.

Nummern entsprechen Indexwerten

Voraussetzung für Adressrechnungen ist die fortlaufende Abspeicherung der Knoten bzw. der durch die Knoten repräsentierten Objekte in einem Feld. Die Abspeicherung ist niveauweise von links nach rechts vorzunehmen. Dann entsprechen die gemäß Abb. 5.6 vergebenen Nummern den sich für die Knoten bzw. Objekte ergebenden Indexwerten.

Bezeichne i, i = 1,..., 2^n - 1, den Feldindex. Für die Ermittlung der Adressen des Vorgängers und der Nachfolger des i-ten Knotens gilt dann:

Adressen von Vorgängern und Nachfolgern

Adresse $f_v(i)$ des Vorgängers des i-ten Knotens:

$f_v(i) = i$ DIV 2, i = 2, ... , 2^n - 1.

Adresse $f_l(i)$ des linken Nachfolgers des i-ten Knotens:

$f_l(i) = 2 \bullet i$, i = 1, ... , 2^{n-1} - 1.

Adresse $f_r(i)$ des rechten Nachfolgers des i-ten Knotens:

$f_r(i) = 2 \bullet i + 1$, i = 1, ... , 2^{n-1} - 1.

Grenzen für den Feldindex *i* bei der Adressberechnung

Übungsaufgabe 5.5

Erläutern Sie kurz die eben für die Adressen $f_v(i)$, $f_l(i)$ und $f_r(i)$ angegebenen Grenzen des Feldindex i.

Prüfung auf Vorgänger/Nachfolger

Ob ein Knoten i einen Vorgänger besitzt oder ob er die Wurzel darstellt, lässt sich auf einfache Weise abprüfen. Es gilt:

Der Knoten *i* besitzt einen Vorgänger, falls:

i DIV $2 \geq 1$, $i = 1, \ldots, 2^n - 1$.

Der Knoten *i* ist die Wurzel, falls:

i DIV $2 = 0$, $i = 1, \ldots, 2^n - 1$.

Abschließend seien noch die Datendefinitionen für einen Binärbaum mit niveauweiser Nummerierung angegeben. Repräsentieren die Knoten Personalsätze, so lauten sie:

Typdefinition für niveauweise Nummerierung

```
DATA
    CONST m = 1023        {allgemein in der Form 2^n - 1 festlegen}
    TYPE
        INDEX = [0..m];
        PERSON =  RECORD
                    pnr : ARRAY [1..8] OF CHAR;
                    pname : ARRAY [1..20] OF CHAR;
                    geb_datum : ARRAY [1..8] OF CHAR;
                    END;
        BINBAUM = ARRAY [1..m] OF PERSON;
    VARIABLE
        wurzel, i : INDEX;
        baum : BINBAUM;
```

Objektselektion mit Feldindex

Die Indexvariable *wurzel* dient der Indizierung der Wurzel des Baums und der Feldindex *i* der Objektselektion. Den ganzen Baum umschließt die Variable *baum*. Ein Knoten bzw. ein Objekt kann bekanntlich mittels *baum[i]* selektiert werden.

5.2.3 Darstellung mit nicht niveauweiser Nummerierung

Reihenfolge der Knoten legt Abspeicherung fest

Stellt die gewählte Programmiersprache kein Zeigerkonzept zur Verfügung und sind darüber hinaus unausgeglichene Bäume zu manipulieren, so bietet sich die Darstellung mit nicht niveauweiser Nummerierung an. Unausgeglichene Bäume seien hier nur kurz so charakterisiert, dass bei einer Darstellung mit niveauweiser Nummerierung relativ viele Dummy-Knoten auftreten, d.h. viel Speicherplatz vergeudet wird. Bei der alternativen Darstellung mit nicht niveauweiser Nummerierung werden die Knoten nicht niveauweise, sondern in irgendeiner Reihenfolge nummeriert und in der Folge der Nummerierung sequentiell in einem Feld abgespeichert. Konsequenzen der nicht niveauweisen Nummerierung sind:

keine Adressrechnung

- Adressrechnungen wie bei der niveauweisen Nummerierung sind nicht mehr anwendbar; es müssen daher die linken und rechten Nachfolger sämtlicher Knoten explizit angegeben werden.

nur belegte Knoten

- Da die Nachfolger sämtlicher Knoten explizit benannt werden, erübrigt sich die Speicherung vollständiger Bäume; zu speichern sind lediglich die belegten Knoten.

Zur Realisierung von Binärbäumen mit nicht niveauweiser Nummerierung eignen sich beispielsweise folgende Datendefinitionen:

Datendefinitionen für nicht niveauweise Nummerierung

```
DATA
    CONST m = 1000;                    {maximale Anzahl von Knoten}
    TYPE
        INDEX = [0..m];
        PERSDATA = RECORD
                       pnr : ARRAY [1..8] OF CHAR;
                       pname : ARRAY [1..20] OF CHAR;
                       geb_datum : ARRAY [1..8] OF CHAR;
                       END;
        PERSON =   RECORD
                       pers : PERSDATA;
                       links, rechts : INDEX;
                       END;
        BINBAUM = ARRAY [1..m] OF PERSON;
    VARIABLE
        wurzel, i : INDEX;
        baum : BINBAUM;
        persdata : PERSDATA;
```

verkettete Speicherung

Die Variablen *wurzel*, *i* und *baum* besitzen die gleiche Bedeutung wie im Fall der niveauweisen Nummerierung. In der Datenstruktur PERSON geben die Komponenten *links* und *rechts* die Speicherplätze des linken und rechten Nachfolgers eines Knotens an. Besitzt ein Knoten *i* keine Nachfolger, dann setzt man:

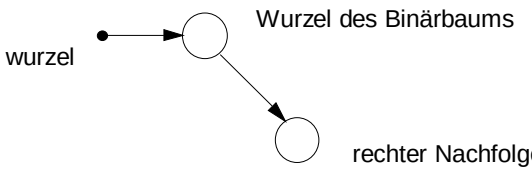
baum[i].links = 0 und *baum[i].rechts* = 0.

Nicht belegte Speicherplätze des Feldes *baum* verwaltet man sinnvollerweise mittels einer Freispeicherliste.

Erzeugung eines einfachen Binärbaums

Übungsaufgabe 5.6

Es soll ein nicht niveauweise nummerierter Binärbaum erzeugt werden, der nur aus der Wurzel und einem rechten Nachfolger besteht. Der Baum hat also folgendes Aussehen:



Den beiden Knoten sind einzulesende Personalsätze (vgl. Beispiel 5.1) zuzuordnen. Wie lautet die zur Erzeugung der beschriebenen Struktur erforderliche Anweisungsfolge?

5.3 Traversieren von Bäumen

5.3.1 Traversierungsbegriff

Durchlaufen sämtlicher Knoten eines Baums

Mit dem Begriff "Traversieren" bezeichnet man das Durchlaufen sämtlicher Knoten eines Baumes in einer bestimmten Reihenfolge. In der Regel wird mit dem Traversieren die Bearbeitung vieler oder aller Knoten bzw. Datenobjekte verbunden sein. Für Binärbäume eignen sich insbesondere auch rekursive Traversierungs-Algorithmen. Ihnen liegen drei grundsätzliche Vorgehensweisen zugrunde:

grundsätzliche Vorgehensweisen

(1) Traversieren in Präordnung (engl. preorder).

(2) Traversieren in symmetrischer Ordnung (engl. inorder).

(3) Traversieren in Postordnung (engl. postorder).

Aufteilung in Wurzel, linker Teilbaum, rechter Teilbaum

Jede dieser Vorgehensweisen beruht auf der Aufteilung eines Binärbaumes in drei Teile: die Wurzel (W), den linken Teilbaum (L) und den rechten Teilbaum (R). Mit einem erzeugten Teilbaum wird in gleicher Weise verfahren. Die rekursive Aufteilung endet mit der Zerlegung von Teilbäumen in nicht weiter unterteilbare Blätter.

Charakteristisch für das Traversieren in Präordnung ist das Zerlegungsmuster (W, L, R). Es wird also zunächst die Wurzel angesprochen, dann der linke Teilbaum und schließlich der rechte Teilbaum. Die rekursive Formulierung lautet:

Durchlaufen in Präordnung

Traversieren in Präordnung (W, L, R):
(1) Aufsuchen der Wurzel W.
(2) Durchlaufen des linken Teilbaums L in Präordnung
(3) Durchlaufen des rechten Teilbaums R in Präordnung.

rekursiver Ansatz

Auf das Aufsuchen der Wurzel W im Schritt (1) folgt der Schritt (2). Er besteht – im Sinne eines rekursiven Ansatzes – im Traversieren des linken Teilbaums in Präordnung. Es wird also zunächst die Wurzel des linken Teilbaums aufgesucht und dann wiederum Schritt (2) angewandt. Erst wenn der linke Teilbaum des gegebenen Binärbaums abgearbeitet ist, wird mit dem rechten Teilbaum in analoger Weise verfahren.

An einem einfachen Beispiel sei das Traversieren in Präordnung demonstriert. Es ist in Abb. 5.7 zu sehen. Dort sind die Knoten in der Reihenfolge des Aufsuchens fortlaufend nummeriert.

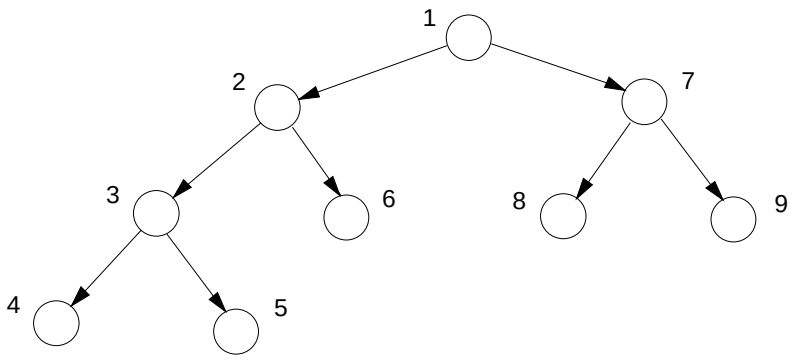


Abb. 5.7. Traversieren eines Binärbaums in Präordnung.

Für das Traversieren in symmetrischer Ordnung gilt das Zerlegungsschema (L, W, R). Zunächst wird also der linke Teilbaum abgearbeitet. Es folgen das Aufsuchen der Wurzel und das Durchlaufen des rechten Teilbaums. Die rekursive Vorschrift für das Durchlaufen lautet;

Durchlaufen in symmetrischer Ordnung

Traversieren in symmetrischer Ordnung (L, W, R):

(1) Durchlaufen des linken Teilbaums L in symmetrischer Ordnung.
(2) Aufsuchen der Wurzel W.
(3) Durchlaufen des rechten Teilbaums R in symmetrischer Ordnung.

Abbildung 5.8 zeigt, in welcher Reihenfolge die Knoten des in Abb. 5.7 angegebenen Binärbaums beim Traversieren in symmetrischer Ordnung durchlaufen werden. Man sieht, dass die rekursive Verschachtelung erst bei dem linken Knoten in der 3. Ebene endet. Dieser Knoten besitzt als Blatt keinen linken Teilbaum mehr und wird daher als erster angesprochen. Nun wird die rekursive Verschachtelung quasi von unten aufgerollt bis die Wurzel des gesamten Binärbaums erreicht ist. Es folgt die Abarbeitung des rechten Teilbaums in analoger Weise.

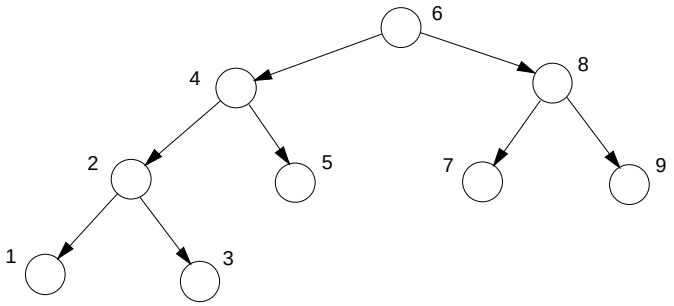


Abb. 5.8. Traversieren eines Binärbaums in symmetrischer Ordnung.

Beim Traversieren in Postordnung rückt das Aufsuchen der Wurzel an das Ende der Tätigkeitsfolge. Als Zerlegungsschema erhält man daher (L, R, W). Für das Durchlaufen gilt folgende rekursive Vorschrift:

Traversieren in Postordnung

Traversieren in Postordnung (L, R, W):

(1) Durchlaufen des linken Teilbaums L in Postordnung.
(2) Durchlaufen des rechten Teilbaums R in Postordnung.
(3) Aufsuchen der Wurzel W.

Die Knoten des als Beispiel betrachteten Binärbaums werden im Fall des Traversierens in Postordnung in der in Abb. 5.9 angegebenen Reihenfolge durchlaufen.

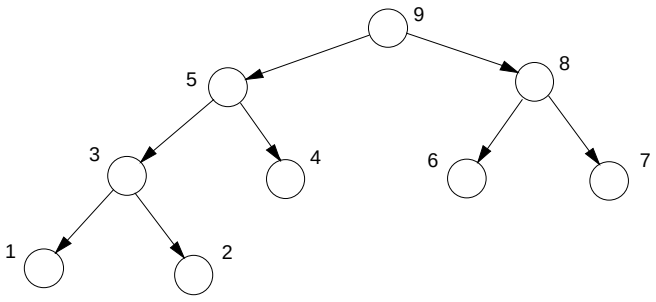


Abb. 5.9. Traversieren eines Binärbaums in Postordnung.

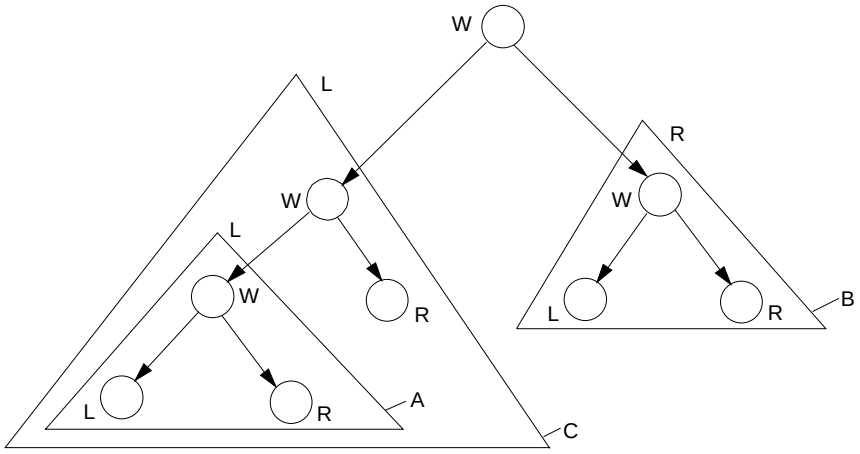
Darstellungsformen der rekursiven Verschachtelung

Um die vorgestellten Traversierungsalgorithmen weiter zu verdeutlichen, sei die rekursive Verschachtelung für den als Beispiel gewählten Fall der symmetrischen Ordnung näher betrachtet. Die Verschachtelungsstruktur lässt sich auch grafisch darstellen und zwar in zweifacher Weise:

- als geschachteltes Mengendiagramm, welches in einer speziellen Form zu gestalten ist, und
- als Zerlegungsschema, welches sich aus dem geschachtelten Mengendiagramm ergibt.

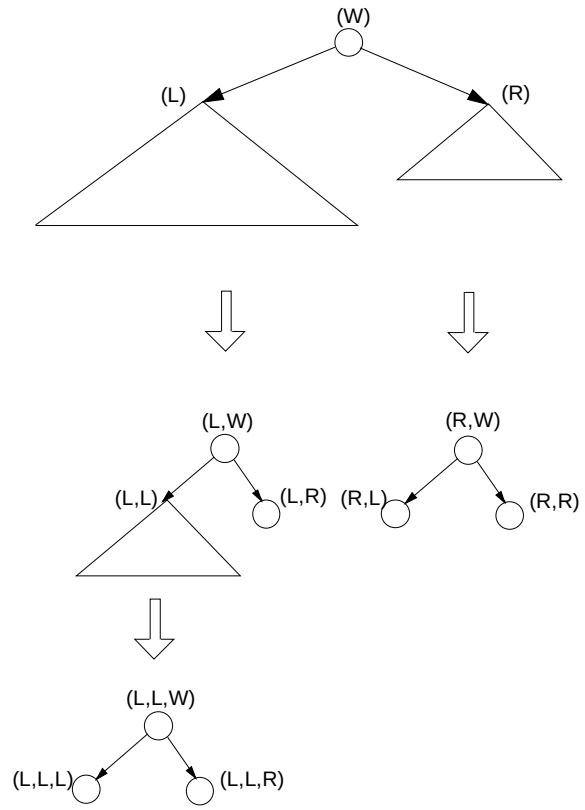
Entsprechende Darstellungen für den in Abb. 5.7 gezeigten Binärbaum finden sich in Abb. 5.10 a) und b).

Darstellung als Mengendiagramm



a) Rekursiv geschachteltes Mengendiagramm

Zerlegungsschema



b) Zerlegungsschema

Abb. 5.10. Schematische Darstellung der rekursiven Verschachtelung beim Traversieren in symmetrischer Ordnung.

Abb. 5.10 a) zeigt eine rekursive Definition des gesamten Binärbaums in folgenden Stufen:

Stufen der rekursiven Definition des Binärbaums

- Unterste Stufe: Zusammensetzen der elementaren Teilbäume A und B aus je drei Knoten, einem linken Knoten, einem Wurzelknoten und einem rechten Knoten.
- Zwischenstufe: Zusammensetzen des höheren Teilbaums C aus dem linken, elementaren Teilbaum A, einem Wurzelknoten und einem rechten Knoten.
- Oberste Stufe: Zusammensetzen des gesamten Binärbaums aus dem linken, höheren Teilbaum C, einem Wurzelknoten und dem rechten, elementaren Teilbaum B.

Bei komplexeren Binärbäumen können noch weitaus mehr Zwischenstufen auftreten.

rekursive Zerlegung in Teilprobleme

Wendet man den symmetrischen Traversierungsalgorithmus auf den rekursiv definierten Binärbaum an, so ergibt sich die in Abb. 5.10 b) skizzierte rekursive Zerlegung in immer feinere Teilprobleme. Die Zerlegung endet mit dem Erreichen von Teilproblemen, die Knoten darstellen. In der Abbildung sind die beim symmetrischen Traversieren auszuführenden Tätigkeiten durch in Klammern gesetzte Buchstaben bzw. Buchstabenfolgen angegeben. Die Buchstabenfolgen drücken Zerlegungsstufen aus, wobei die Buchstaben die bereits bekannten Bedeutungen behalten:

L Durchlaufen des linken Teilbaums,

W Aufsuchen der Wurzel,

R Durchlaufen des rechten Teilbaums.

So ist beispielsweise die Buchstabenfolge (L, L, W) wie folgt zu interpretieren: Zuerst Zugang zum linken Teilbaum des gesamten Binärbaums, dann Zugang zum linken Teil dieses Teilbaums und schließlich Aufsuchen der Wurzel des linken Teils.

Insgesamt folgt somit für die Reihenfolge des Ansprechens der Knoten:

(L, L, L) → 1
(L, L, W) → 2
(L, L, R) → 3
(L, W) → 4
(L, R) → 5
(W) → 6
(R, L) → 7
(R, W) → 8
(R, R) → 9

Dies entspricht der in Abb. 5.8 angegebenen Reihenfolge.

Reihenfolge der Knoten beim Traversieren in symmetrischer Ordnung

Übungsaufgabe 5.7

Tragen Sie in den nachstehend angegebenen Binärbaum die Reihenfolge des Ansprechens der Knoten für den Fall des Traversierens in symmetrischer Ordnung ein.