Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Kapiteltests zum Leitprogramm

Binäre Suchbäume

Björn Steffen
Timur Erdag
überarbeitet von Christina Class

Binäre Suchbäume

Kapiteltests für das ETH-Leitprogramm

Adressaten und Institutionen Das Leitprogramm richtet sich an Studierende an einer Fachhochschule im 2. Semester des Informatik- oder Elektrotechnikstudiums.

Vorkenntnisse Die Leser und Leserinnen

- beherrschen die wesentlichen Aspekte einer Programmiersprache.
- kennen das Konzept der verketteten Liste.
- verstehen die Rekursion und deren Anwendung.

Dauer Für die vollständige Bearbeitung des Leitprogramms werden ungefähr 6–8 Stunden benötigt.

Betreuerin und Überarbeitung
Christina Class

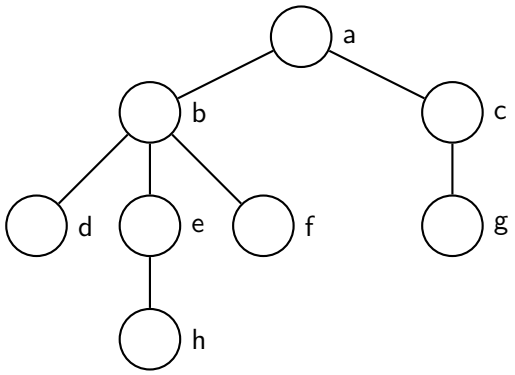
Autoren
Björn Steffen
Timur Erdag

10. April 2008

2 Bäume

Aufgabe 2.1
Wie viele Knotentypen kennen Sie? Zählen Sie sie auf und beschreiben Sie sie.

Aufgabe 2.2
Geben Sie den Namen aller Komponente im folgenden Baum an:



Aufgabe 2.3
Gegeben sei der Baum aus Aufgabe 2.2. Geben Sie für jeden Knoten die Tiefe und den Grad an.

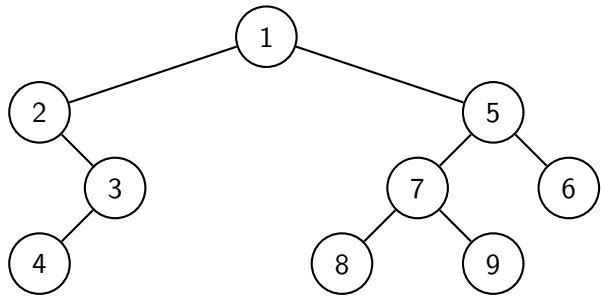
Aufgabe 2.4
Zeichnen Sie im Baum der Aufgabe 2.2 die Niveaus ein und bestimmen Sie die Höhe des Baumes.

Aufgabe 2.5
Welche Aussage können Sie über die Ordnung des Baumes in Aufgabe 2.2 machen? Begründen Sie.

Aufgabe 2.6
Der Baum in Aufgabe 2.2 ist nicht ausgefüllt. Erklären Sie warum.

3 Binärbäume

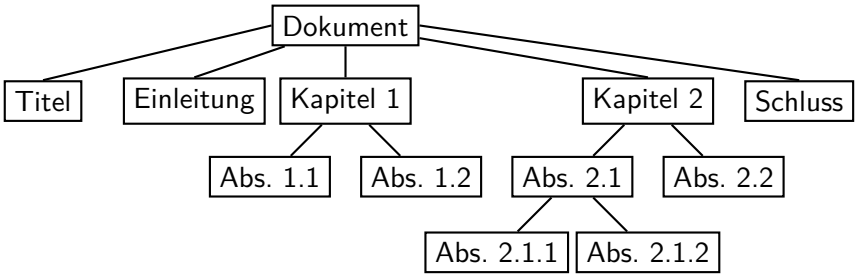
Aufgabe 3.1
Gegeben ist der folgenden Binärbaum. Geben Sie die Preorder-, Postorder- und Inorder-Reihenfolge der Knoten an.



Aufgabe 3.2
Zeichnen Sie einen Binärbaum mit den folgenden Eigenschaften:

- Jeder Knoten speichert einen Buchstaben.
- Die Preorder-Reihenfolge der Knoten des Baumes ist: ÜBUNGEN.
- Die Inorder-Reihenfolge lautet: NUGBEÜN.

Aufgabe 3.3
Gegeben sei der folgende Baum, der die Struktur der Kapitel und Abschnitte eines Dokuments zeigt.



Schreiben Sie eine Algorithmus in Pseudocode der für den gezeigten Baum die folgende Ausgabe liefert:

```
Dokument
    Titel
    Einleitung
    Kapitel 1
        Abschnitt 1.1
        Abschnitt 1.2
    Kapitel 2
        Abschnitt 2.1
            Abschnitt 2.1.1
            Avschnitt 2.1.2
        Abschnitt 2.2
    Schluss
```

Aufgabe 3.4
Die **Pfadlänge** eines Baumes B ist gegeben durch die Summe der Tiefen aller Knoten des Baumes.

$$\text{Pfadlänge} = \sum_{v \in B} \text{depth}(v)$$

Vervollständigen Sie den folgenden Algorithmus der die Pfadlänge eines Binärbaumes in *einem* Durchlauf berechnet. Das heisst, jeder Knoten darf dabei nur *einmal* besucht werden.

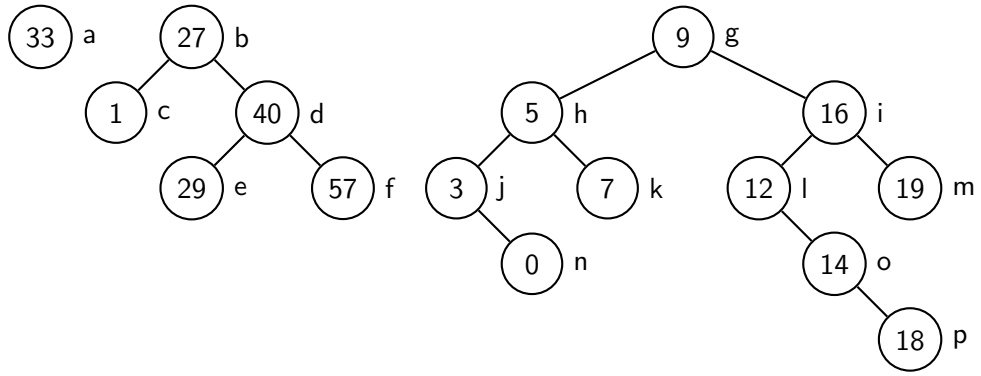
```
1: algorithm pathLength(v)
   {Berechnet die Pfadlänge des Baumes mit Wurzel v}
2:     return pathLengthInternal(v, ...)
3: end algorithm

4: algorithm pathLengthInternal(v, t)
   {Berechnet die Pfadlänge des Teilbaumes mit Wurzel v auf Tiefe t}
5:     ...
6:     return ...
7: end algorithm
```

4 Binäre Suchbäume

Aufgabe 4.1
Zählen Sie die zwei Eigenschaften auf, welche einen Binärbaum zu einem binären Suchbaum macht.

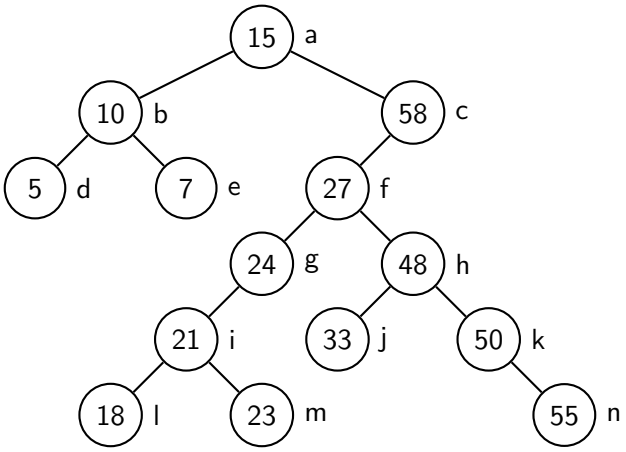
Aufgabe 4.2
Entscheiden Sie, ob folgende drei Bäume korrekte binäre Suchbäume sind und begründen Sie:



Aufgabe 4.3
Fügen Sie die folgenden Knoten in der gegebenen Reihenfolge ein. Ausgangslage ist ein leerer Baum:

Reihenfolge	Knoten	Schlüssel
1.	a	27
2.	b	17
3.	c	37
4.	d	21
5.	e	43
6.	f	20
7.	g	17
8.	h	0

Aufgabe 4.4
Gegeben ist folgender Baum:



Entfernen Sie nun die Knoten in der gegebenen Reihenfolge: a, c, n, h, l, b, d, g. Geben Sie jedoch auch hier zuerst an, welchen Fall Sie benutzen.

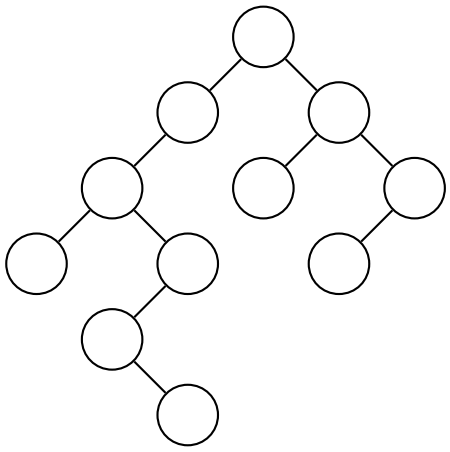
5 Balancierte Suchbäume

Aufgabe 5.1
Zeichnen Sie einen binären Suchbaum mit den folgenden Schlüsseln:

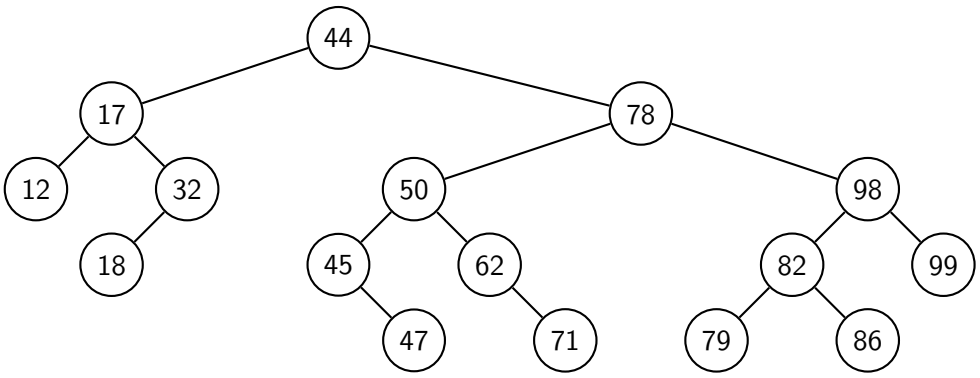
$$3, 5, 8, 13, 18, 24, 27, 34, 45, 48, 51, 63, 67$$

Der Suchbaum sollte so ausgeglichen wie möglich sein. Beschreiben Sie kurz, wie Sie vorgehen um den Suchbaum zu zeichnen.

Aufgabe 5.2
Zeichnen Sie beim folgenden Binärbaum die Balancefaktoren aller Knoten ein und bestimmen Sie, ob diese ausgeglichen sind. Ist der Baum AVL-ausgeglichen?



Aufgabe 5.3
Bestimmen Sie die Balancefaktoren aller Knoten des folgenden Suchbaumes. Handelt es sich um einen AVL-Baum?



A Lösungen zum Kapitel 2

Lösung zur Aufgabe 2.1

i. Wurzel: In jedem Baum (ausser dem leeren Baum) steht sie zuoberst und hat dementsprechend keinen Vaterknoten.
ii. Blatt: Knoten ohne Kinder.
iii. Innerer Knoten: Jeder Knoten, welcher mindestens ein Kind hat.

Lösung zur Aufgabe 2.2

Sie sehen in folgender Tabelle die Lösung. Die Kanten werden hier nicht erwähnt.

Knoten	Wurzel	Blatt	inner Knoten
a	x		x
b			x
c			x
d		x	
e			x
f		x	
g		x	
h		x	

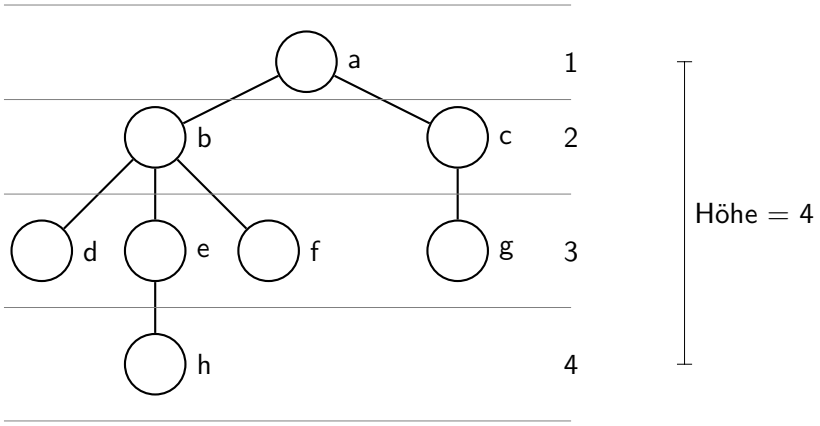
Lösung zur Aufgabe 2.3

Die Tiefe und der Grad der einzelnen Knoten stehen in folgender Tabelle:

Knoten	Grad	Tiefe
a	2	1
b	3	2
c	1	
d	0	3
e	1	
f	0	
g	0	
h	0	4

Lösung zur Aufgabe 2.4

Die Niveaus und die Höhe sind im folgendem Baum eingezeichnet:



Lösung zur Aufgabe 2.5
Der Baum muss mindestens Ordnung 3 haben, da Knoten b den grössten Grad (3) hat.

Lösung zur Aufgabe 2.6
Annahme: Dieser Baum hat Ordnung 3. Dann haben die Knoten a, c und e zu wenig Kinder. Sie müssten je 3 Kinder haben, damit dieser Baum ausgefüllt wäre. Bei jeder höheren Ordnung hätten *alle* inneren Knoten zu wenig Kinder.

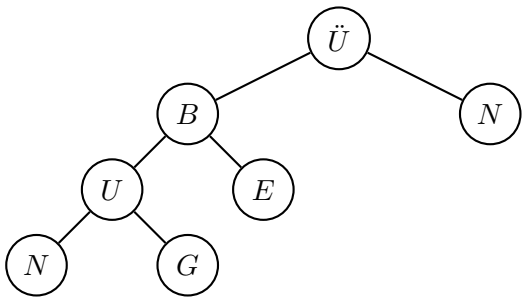
B Lösungen zum Kapitel 3

Lösung zur Aufgabe 3.1

- Preorder: 1 2 3 4 5 7 8 9 6
- Postorder: 4 3 2 8 9 7 6 5 1
- Inorder: 2 4 3 1 8 7 9 5 6

Lösung zur Aufgabe 3.2

Der gesuchte Binärbaum sieht folgerndermassen aus:



Lösung zur Aufgabe 3.3

Die Preoder-Reihenfolge bringt das gewünschte Resultat.

```
1: algorithm preorder(v)
2:     {Durchläuft alle Knoten des Binärbaumes mit Wurzel v in der Preorder-
   Reihenfolge}
3:     if v ≠ null then
4:         Besuche den Knoten v {∗}
5:         preorder( left(v) )
6:         preorder( right(v) )
7:     end if
8: end algorithm
```

Lösung zur Aufgabe 3.4
Der Algorithmus ist gegeben durch:

```
1: algorithm pathLength(v)
2:     return pathLengthInternal(v, 1)
3: end algorithm

4: algorithm pathLengthInternal(v, t)
5:     sum ← 0
6:     sum ← sum + pathLengthInternal(left(v), t + 1)
7:     sum ← sum + pathLengthInternal(right(v), t + 1)
8:     return sum + l
9: end algorithm
```

C Lösungen zum Kapitel 4

Lösung zur Aufgabe 4.1

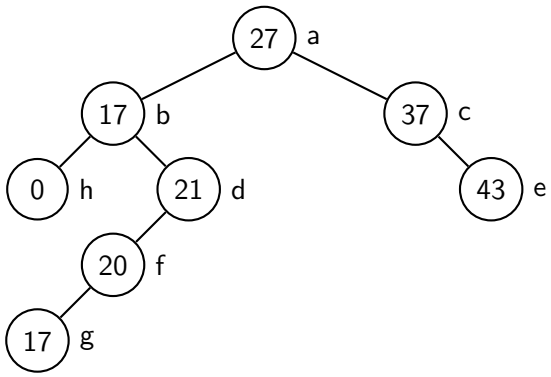
- Jeder Schlüssel im linken Teilbaum eines Knotens ist kleiner als der Schlüssel im Knoten selbst.
- Jeder Schlüssel im rechten Teilbaum eines Knotens ist grösser als oder gleich dem Schlüssel im Knoten selbst.

Lösung zur Aufgabe 4.2

Die zwei Bäume links und in der Mitte sind korrekte binäre Suchbäume. Der Baum rechts hingegen ist aus zwei Gründen kein binärer Suchbaum: Sowohl Knoten n als auch Knoten p sind an der falschen Stelle.

Lösung zur Aufgabe 4.3

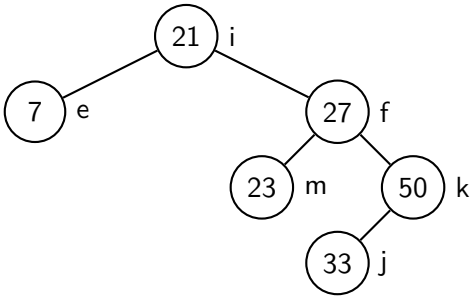
So sollte Ihr Baum nun aussehen:



Lösung zur Aufgabe 4.4

Welcher Fall beim Entfernen angewendet wurde, entnehmen Sie der Tabelle. Der folgende Baum gibt an, wie der binäre Suchbaum am Ende aller Löschoperationen aussieht.

Knoten	Fall
a	3
c	2
n	1
h	3
l	3
b	3
d	1
g	2

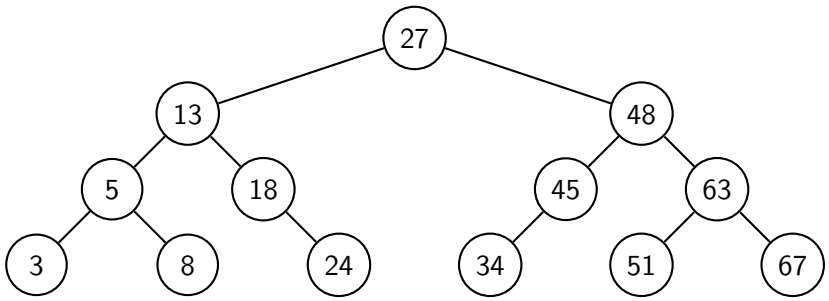


D Lösungen zum Kapitel 5

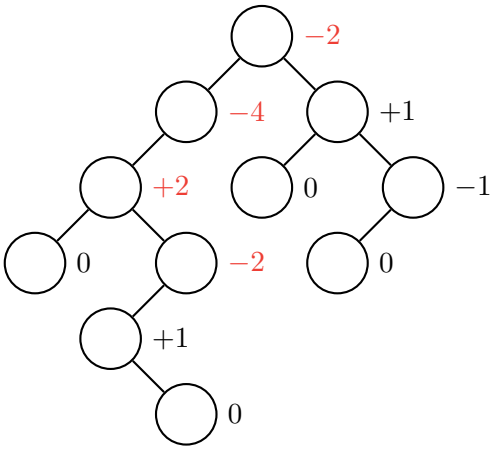
Lösung zur Aufgabe 5.1
Das folgende rekursive Vorgehen eignet sich am besten, um einen möglichst ausgeglichenen binären Suchbaum für die gegebenen Schlüssel zu zeichnen:

1. Der Median (mittlere Wert) der Schlüssel wird zur Wurzel.
2. Die Schlüssel die kleiner als dieser Median sind, bilden den linken Teilbaum der Wurzel, während die grösseren Schlüssel den rechten Teilbaum bilden.
3. Das gleiche Vorgehen wird dann angewendet um die Teilbäume zu bilden.

Ein möglichst ausgeglichener binärer Suchbaum mit den angegebenen Schlüsseln könnte folgendermassen aussehen:



Lösung zur Aufgabe 5.2
Der Baum ist nicht AVL-ausgeglichen, da mehrere Knoten einen Balancefaktor ≥ 2 haben.



Lösung zur Aufgabe 5.3
Die folgende Zeichnung zeigt den Suchbaum mit den eingetragenen Balancefaktoren. Es ist ein AVL-Baum, da alle Knoten ausgeglichen sind.