Technische Universität Darmstadt
Fachgebiet Theoretische Informatik

Prof. Johannes Buchmann
Nabil Alkeilani Alkadri
Nina Bindel
Patrick Struck

Algorithmen und Datenstrukturen

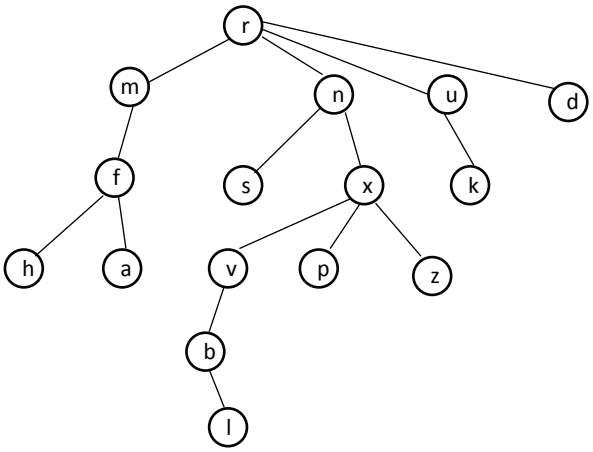
SoSe 2018

TECHNISCHE UNIVERSITÄT DARMSTADT

6. Lösungsblatt — 21.05.2018
v1.1

P1 Bäume

Gegeben sei der folgende Baum mit Wurzel r.



1. Geben Sie die Höhe des Baums sowie der Knoten f und x an.
2. Was ist der Grad der Knoten m, x und n?
3. Wie viele innere Knoten und wie viele Blätter besitzt der Baum?
4. Wie heißt der Elternknoten der Knoten b bzw. k?
5. Welche Knoten sind im Teilbaum mit Wurzel n enthalten?
6. Geben Sie den/die Geschwisterknoten des Knoten p an.

Lösung.

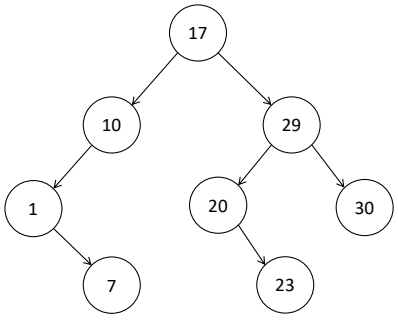
1. Die Höhe des Baumes entspricht der Höhe seiner Wurzel r. Diese ist 5 ((Pfad über n, x, v, b und l).
   Der Knoten f hat die Höhe 1, x hat die Höhe 3.
2. Der Grad eines Knoten ist die Anzahl seiner Kindknoten. Daher hat m der Grad 1, x hat Grad 3 und n hat Grad 2.
3. Innere Knoten: 7. Blätter: 8
4. Der Elternknoten des Knoten b ist v, wobei der von k ist u.
5. Der Teilbaum mit Wurzel n enthält die Knoten: n (Wurzel), s, x, v, p, z, b und l.
6. Der Knoten p hat zwei Geschwisterknoten: v und z.

P2 Binäre Suchbäume

1. Fügen Sie nacheinander Knoten mit den Schlüsseln 17, 10, 1, 29, 7, 20, 30, 23 in einen leeren binären Suchbaum ein. Skizzieren Sie den resultierenden Baum.
2. Welche Höhe besitzt der Baum?
3. Wieviele Blätter besitzt der Baum?
4. Traversieren Sie den Baum jeweils nach dem Inorder-, Preorder- und dem Postorder-Verfahren. Fällt Ihnen bei einem etwas auf?
5. Entfernen Sie nun nacheinander die Knoten mit den Schlüsseln 1, 23, 29. Geben Sie den entstehenden Baum, sowie für jeden Löschvorgang den zutreffenden Fall des Löschens an.

Lösung.

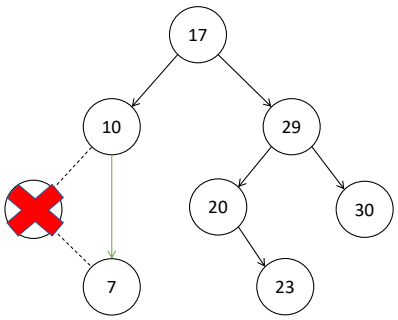
1. Der Baum mit den gegebenen Schlüsseln sieht wie folgt aus:



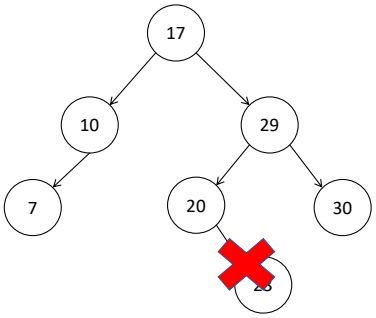
2. die Höhe ist 3.
3. Der Baum besitzt 3 Blätter: 7, 23 und 30.
4. Inorder: 1, 7, 10, 17, 20, 23, 29, 30
   Preorder: 17, 10, 1, 7, 29, 20, 23, 30

Postorder: 7, 1, 10, 23, 20, 30, 29, 17
Die Inorder-Traversierung ist sortiert.

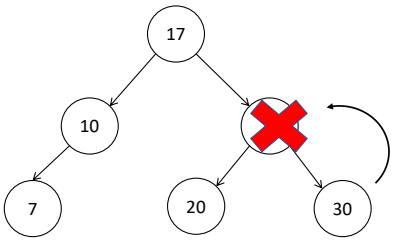
5. Löschen von Knoten 1 (Löschen von Knoten mit einem Kind):



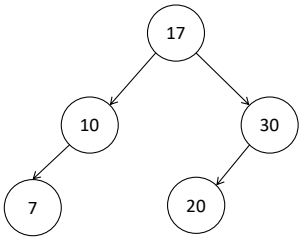
Löschen von Knoten 23 (Löschen eines Blatts):



Löschen von Knoten 29 (Löschen von einem Knoten mit zwei Kindern):



Der Baum sieht am Ende wie folgt aus:



P3 Implementierung Höhe

1. Entwickeln Sie in Pseudocode einen rekursiven Algorithmus, der die Höhe eines binären Baums ausgehend von der Wurzel v berechnet.
2. Bestimmen Sie die Komplexität des Algorithmus abhängig von der Anzahl der Knoten.

Lösung.

1. Der Pseudocode ist in Algorithmus 1 gegeben.
2. Da sämtliche Knoten durchlaufen werden müssen, ist die Komplexität des Algorithmus $\mathcal{O}(n)$.

P4 Eigenschaften von Binärbäumen

1. Geben Sie eine Formel für die Anzahl der Knoten in einem vollständigen Binärbaum an.
2. Wie viele Blätter hat ein vollständiger Binärbaum der Höhe h?

Algorithm 1 Pseudocode zur Berechnung der Höhe eines eines binären Baums ausgehend von der Wurzel.

```
1:  procedure HEIGHT(v)
2:      heightLeft, heightRight ← 0
3:      if v = NIL then                                  ▷ Baum existiert nicht
4:          return -1
5:      end if
6:      if v.left = NIL ∧ v.right = NIL then             ▷ Rekursionsanker
7:          return 0
8:      end if
9:      heightLeft ← HEIGHT(v.left)                      ▷ rekursiver Aufruf
10:     heightRight ← HEIGHT(v.right)                    ▷ rekursiver Aufruf
11:     return 1 + Max(heightLeft, heightRight)
12: end procedure
```

3. Sie sollen eine bereits sortierte Liste in einen binären Suchbaum umwandeln. Was für ein Problem tritt auf, und wie kann man es vermeiden?

4. Bei bestimmten Arten von Listen lässt sich durch das Prinzip der binären Suche ebenfalls logarithmische Komplexität erreichen. Argumentieren Sie warum dies bei einer verketteten Liste nicht der Fall ist und stattdessen ein binärer Suchbaum verwendet werden muss.

Lösung.

1. Die Formel ist gegeben durch $2^{h+1} - 1$.

2. 2^h Blätter.

3. Würde man einfach die Elemente in der bereits bestehenden Reihenfolge einfügen, würden sich die Knoten sehr ungleichmäßig auf die linken oder rechten Äste verteilen. Dies besitzt eine schlechte Performance. Stattdessen könnten z.B. die Elemente in zufälliger Reihenfolge hinzugefügt werden.

4. Da bei einer verketteten Liste beim Zugriff auf einen bestimmten Index die Liste durchlaufen werden müsste, gibt es keinen Vorteil gegenüber der linearen Suche. Verwendet man hingegen einen Baum oder eine auf Arrays basierende Liste, erreicht man logarithmische Komplexität.

H1 Suchpfade in binären Suchbäumen

Die folgenden Zahlenfolgen sollen Suchpfade bei der Suche nach einem Wert in einem binären Suchbaum darstellen. Bestimmen Sie ob ein binärer Suchbaum existiert, auf dem der angegebene Suchpfad entsteht. Falls nicht, begründen Sie warum dieser nicht existieren kann.

1. 75, 85, 130, 82, 150, 140, 145

2. 14, 24, 20, 23, 21, 22

3. 345, 980, 756, 520, 437, 320, 390

4. 345, 980, 756, 520, 437, 420, 390

5. 345, 980, 756, 520, 687, 590, 623

Lösung.

1. Nein, es existiert kein gültiger Suchbaum für diesen Suchpfad, da die 82 im rechten Teilbaum des Knotens 85 enthalten sein müsste. Dies verletzt jedoch die binäre Suchbaumeigenschaft.

2. Für diesen Pfad existieren binäre Suchbäume.

3. Es existiert kein binärer Suchbaum, da 320 im rechten Teilbaum von 345 enthalten sein müsste, was nicht erlaubt ist.

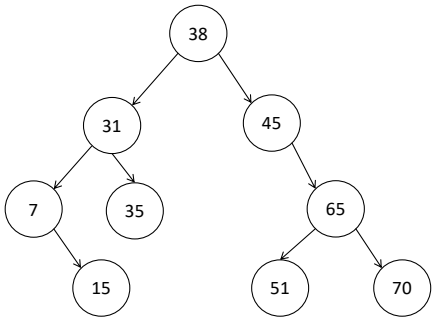
4. Für diesen Pfad existieren binäre Suchbäume.

5. Für diesen Pfad existieren binäre Suchbäume.

H2 Binäre Suchbäume

Fügen Sie nacheinander die Werte 38, 31, 45, 65, 51, 7, 15, 35, 70 in einen leeren binären Suchbaum ein. Skizzieren Sie den enstehenden Baum. Handelt es sich dabei um einen Baum, der eine effiziente Suche ermöglicht?

Lösung.



Da der Baum einigermaßen balanciert ist, bietet er verglichen mit der linearen Suche einen Performancegewinn. Ohne zuätzliche Balancierungsmaßnahmen werden in der Praxis selten perfekt balancierte Bäume erreicht.