TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Lehrstuhl für Sprachen und Beschreibungsstrukturen
Grundlagen: Algorithmen und Datenstrukturen
Prof. Dr. Helmut Seidl, J. Kranz, A. Reuss, R. Vogler

SS 2016
Übungsblatt 8
05.06.2016

Abgabe: **12.06.2016** (bis 23:59 Uhr)

Wiederholung einiger Definitionen: Ein Knoten eines gewurzelten Baumes ist ein Blatt, wenn er keine Kinder hat. Andernfalls ist er ein innerer Knoten.

(In der Literatur wird ein gewurzelter Baum, der nur aus der Wurzel besteht, nicht einheitlich gehandhabt - in einigen Fällen ist dieser einzelne Knoten per Definition ein innerer Knoten, in anderen Fällen ein Blatt, und in wieder anderer Literatur ein Spezialfall, der weder innerer Knoten, noch Blatt ist. Bei unserer Definition ist ein solcher Knoten ein Blatt.)

Die Tiefe eines Knotens ist die Länge des (eindeutigen) Pfades von dem Knoten zur Wurzel, gemessen in der Zahl der Kanten des Pfades. D.h. insbesondere, dass die Wurzel Tiefe 0 hat. (Dies ist nicht einheitlich in der Literatur: in einigen Fällen wird in der Zahl der Knoten des Pfades gemessen, was dann gerade 1 mehr ist als die Zahl der Kanten).

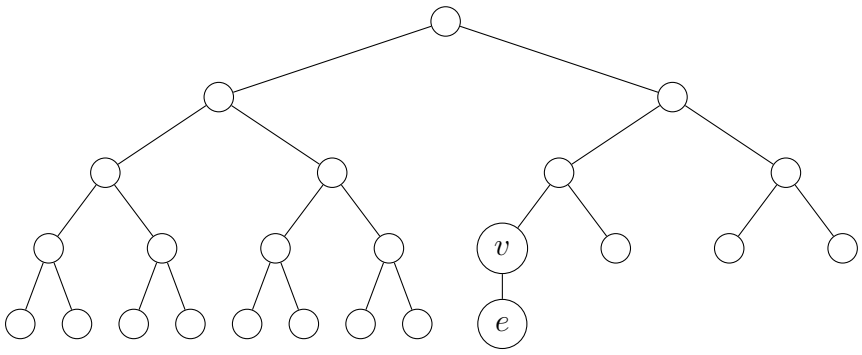
Die Baumtiefe eines gewurzelten Baumes ist das Maximum der Tiefen aller enthaltenen Knoten. Das i-te Level eines gewurzelten Baumes besteht aus allen Knoten, die Tiefe i besitzen.

Ein gewurzelter Baum ist ein Binärbaum, wenn jeder innere Knoten maximal zwei Kinder hat. Wenn jeder innere Knoten genau zwei Kinder hat, ist er ein echter Binärbaum. Ein Binärbaum ist vollständig, wenn er ein echter Binärbaum ist und alle Blätter dieselbe Tiefe haben.

Sei t die Tiefe eines Binärbaumes T. Der Baum T ist ein fast vollständiger Binärbaum, wenn T entweder nur aus der Wurzel besteht oder wenn die ersten $t-1$ Level gemeinsam einen vollständigen Binärbaum bilden, und die Knoten von T so angeordnet werden können, dass es auf dem Level t einen Knoten e mit dem Vaterknoten v gibt, sodass gilt:

- Alle Knoten auf dem Level $t-1$, die „links“ von v stehen, haben zwei Kinder.
- Alle Knoten auf dem Level $t-1$, die „rechts“ von v stehen, haben keine Kinder.

Die folgende Abbildung zeigt einen fast vollständigen Binärbaum mit Baumtiefe 4.



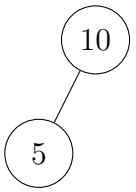
Aufgabe 8.1 (P) AVL-Bebaumung

Gegeben sei ein AVL-Baum der nur aus einem Knoten mit Schlüssel 10 besteht. Fügen Sie nacheinander die Schlüssel 5, 17, 3, 1, 4 ein. Löschen Sie dann den Schlüssel 4, und fügen Sie dann die Schlüssel 8, 2, 7, 6, 9 ein. Löschen Sie dann die Knoten mit den Schlüsseln 2, 1, 8. Zeichnen Sie den AVL-Baum für jede Einfüge- bzw. Löschoperation und geben Sie an, ob Sie keine, eine Einfach- oder Doppelrotation durchgeführt haben.

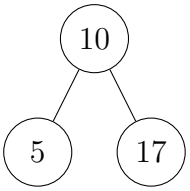
Lösungsvorschlag 8.1

Zu Demonstrationszwecken werden mehr Bäume dargestellt als in der Angabe verlangt ist: Der Baum vor einer Rotation wird mit (i), danach mit (ii) bezeichnet. Bei einer Doppelrotation bestehend aus zwei Einzelrotationen wird der Baum vor der Doppelrotation mit (i), nach der ersten Rotation mit (ii), und nach der zweiten Rotation mit (iii) bezeichnet. Der Einfachheit halber denken wir uns die Knoten-Flags gegebenenfalls nur dazu.

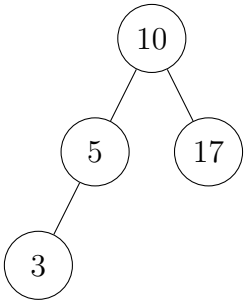
insert von 5 (ohne Rotation):



insert von 17 (ohne Rotation):

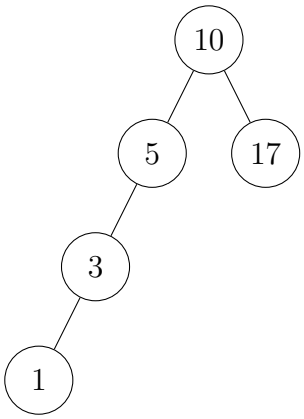


insert von 3 (ohne Rotation):

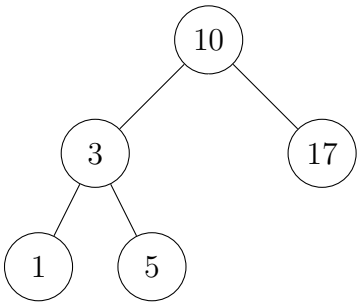


insert von 1 mit Rotation (nach rechts) am Knoten 5:

(i)

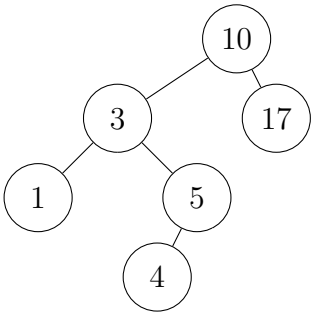


(ii)

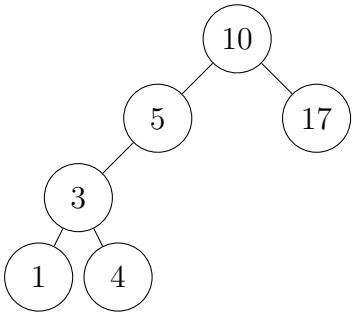


insert von 4 mit Doppelrotation am Knoten 10 (zuerst eine Linksrotation am Knoten 3, dann eine Rechtsrotation am Knoten 10):

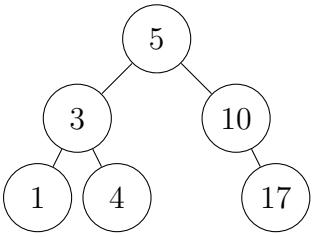
(i)



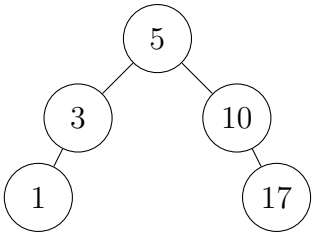
(ii)



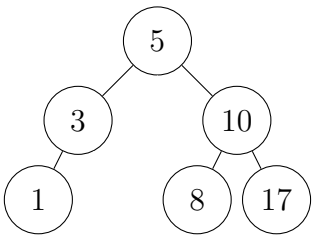
(iii)



Löschen von 4 (ohne Rotation):

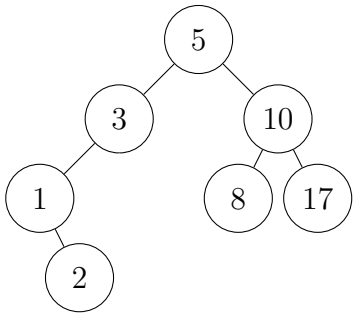


insert von 8 (ohne Rotation):

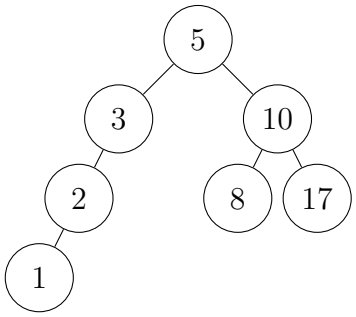


insert von 2 mit Doppelrotation am Knoten 3 (zuerst eine Linksrotation am Knoten 1, dann eine Rechtsrotation am Knoten 3):

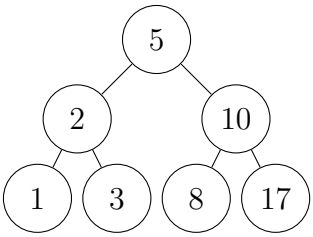
(i)



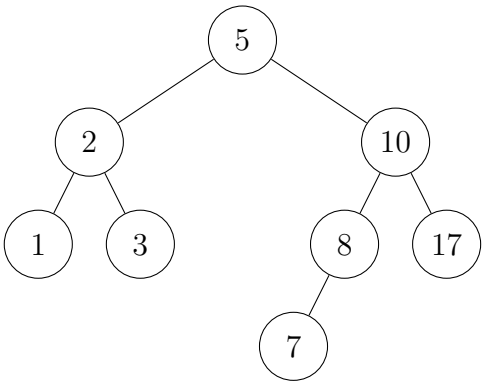
(ii)



(iii)

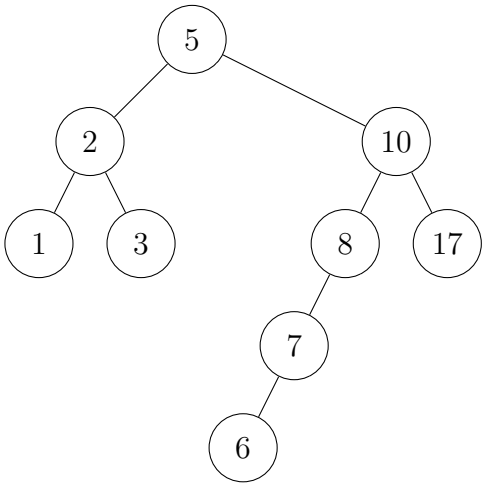


insert von 7 (ohne Rotation):

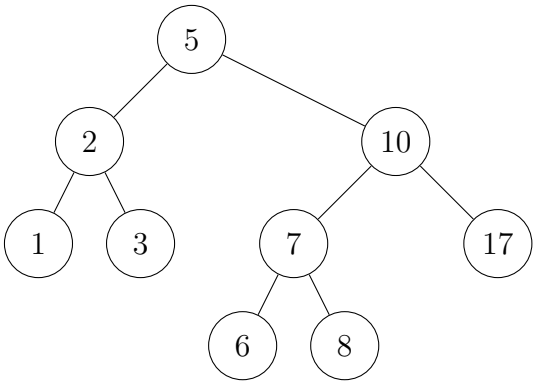


insert von 6 mit Rotation (nach rechts) am Knoten 8:

(i)

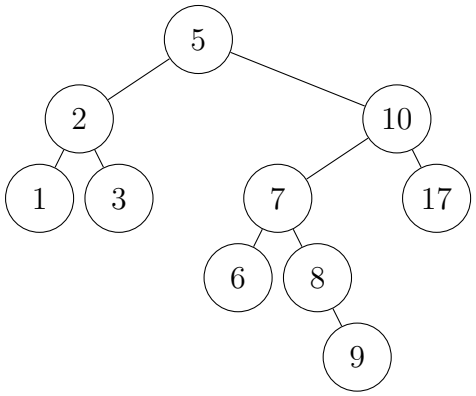


(ii)

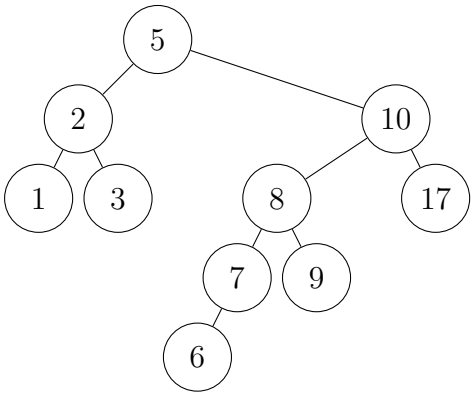


insert von 9 mit Doppelrotation am Knoten 10 (zuerst eine Linksrotation am Knoten 7, dann eine Rechtsrotation am Knoten 10):

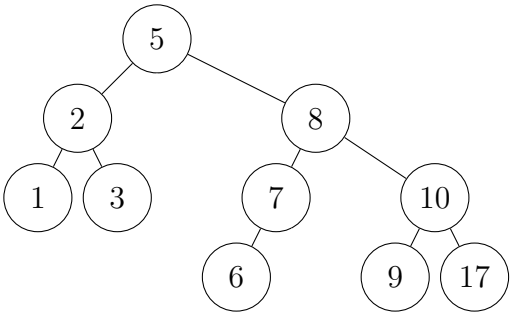
(i)



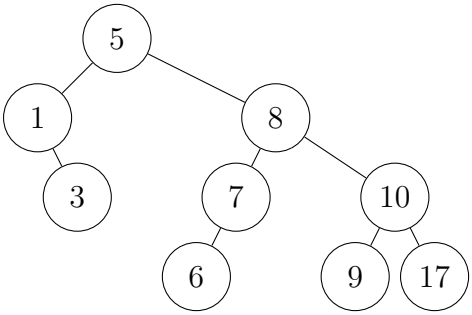
(ii)



(iii)

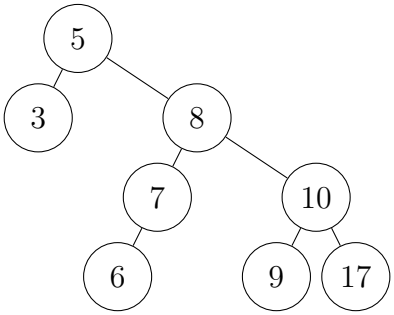


Löschen von 2 (ohne Rotation):

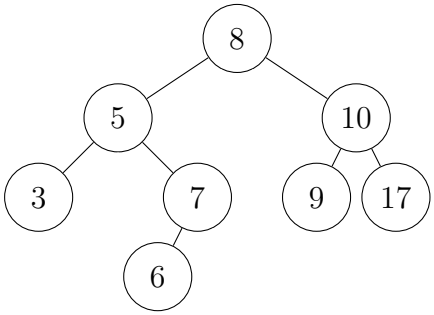


Löschen von 1 mit Rotation (nach links) am Knoten 5:

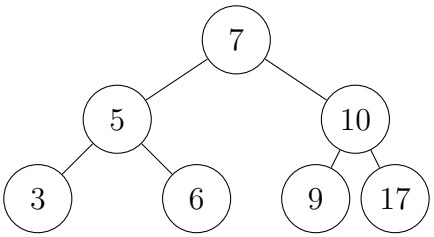
(i)



(ii)



Löschen von 8 (ohne Rotation):



Aufgabe 8.2 (P) Fast vollständige Birnbäume

In der Vorlesung wurde die folgende Aussage verwendet: Jeder fast vollständige Binärbaum mit $n \geq 1$ Knoten und Baumtiefe $t \geq 0$ erfüllt $2^t \leq n \leq 2^{t+1} - 1$.

Beweisen Sie diese Aussage.

Lösungsvorschlag 8.2

Wir machen zwei Beobachtungen:

- Ein vollständiger Binärbaum mit Baumtiefe t besitzt stets mindestens so viele Knoten wie ein fast vollständiger Binärbaum mit Baumtiefe t.
- Ein vollständiger Binärbaum mit Baumtiefe $t-1$ besitzt stets weniger Knoten als ein fast vollständiger Binärbaum mit Baumtiefe t.

Ein vollständiger Binärbaum mit Baumtiefe t hat genau $\sum_{i=0}^{t} 2^i = 2^{t+1} - 1$ Knoten, da es auf Level 0 genau einen Knoten gibt und sich die Zahl der Knoten mit jedem Level verdoppelt. Somit ergibt sich $2^t - 1 < n \leq 2^{t+1} - 1$, was zu beweisen war.

Aufgabe 8.3 (P) Zusatzaufgabe

Wir betrachten die Anzahl der Blocktransfers, die beim externen Sortieren auftreten. Die Komplexität ist in der Vorlesung mit $\mathcal{O}(\frac{n}{B} \log_{M/B} \frac{n}{M})$ hergeleitet worden. Hierbei ist M die Größe des internen Speichers, B die Größe eines Blocks und n die Zahl der zu sortierenden Elemente im externen Speicher. Man kann für externes Sortieren zeigen, dass im Worst-Case $\Omega(\frac{n}{B} \log_{M/B} \frac{n}{B})$ Blocktransfers benötigt werden. Ist damit der in der in der Vorlesung gezeigte Algorithmus asymptotisch optimal?

Lösungsvorschlag 8.3

Wir zeigen, dass der Algorithmus asymptotisch optimal ist. Es gilt:

$$\begin{aligned}\frac{n}{B} \log_{M/B} \frac{n}{M} &= \frac{n}{B}(\log_{M/B} \frac{nB}{BM}) = \frac{n}{B}(\log_{M/B} \frac{n}{B} + \log_{M/B} \frac{B}{M}) \\ &= \frac{n}{B}(\log_{M/B} \frac{n}{B} - 1) \leq \frac{n}{B}(\log_{M/B} \frac{n}{B})\end{aligned}$$

Somit ist also $\frac{n}{B} \log_{M/B} \frac{n}{M} \in \mathcal{O}(\frac{n}{B} \log_{M/B} \frac{n}{B})$. Dies bedeutet, dass die obere Schranke für die Worst-Case-Laufzeit des Algorithmus asymptotisch höchstens so schnell wächst wie die untere Schranke. Langsamer wachsen kann sie aber auch nicht, da $\frac{n}{B} \log_{M/B} \frac{n}{B}$ sonst keine untere Schranke wäre. Deshalb können sich untere und obere Schranke nur maximal um einen konstanten Faktor unterscheiden. Daher ist die Worst-Case-Laufzeit des Algorithmus

asymptotisch auch nur maximal um einen konstanten Faktor größer als die untere Schranke. Der Algorithmus ist damit asymptotisch optimal.