Algorithmen und Datenstrukturen
Sommersemester 2022
Blatt 5

Kevin Angele, Tobias Dick, Oskar Neuhuber,
Andrea Portscher, Monika Steidl, Laurin Wischounig

Abgabe bis 26.04.2022 23:59
Besprechung im PS am 28.04.2022

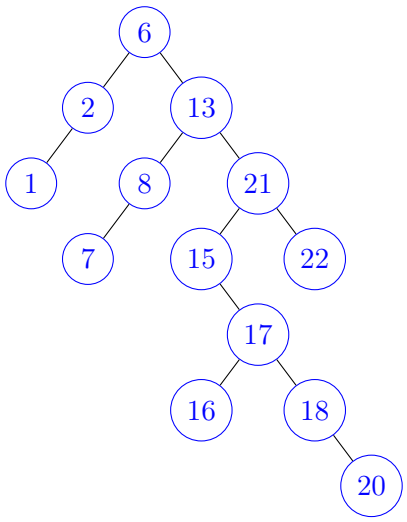
Aufgabe 1 (2 Punkte): Binärer Suchbaum

Zeichnen Sie einen binären Suchbaum, indem Sie die folgenden Schlüssel von links nach rechts in einem anfangs leeren Baum einfügen:

6	13	2	21	8	15	22	7	1	17	16	18	20

Benutzen Sie dafür die in der Vorlesung besprochenen Methoden `addRoot`, `addLeft` und `addRight`. Welche Laufzeitkomplexität hat die Suche nach einem bestimmten Schlüssel, wenn die Suche beim Wurzelknoten beginnt? Zeigen Sie den Baum anschließend in Array-Notation.

Lösung:



Wir brauchen ein Array der Länge 127:

6	2	13	1		8	21					7		15	22					
								17											
																	16	18	
																		20	

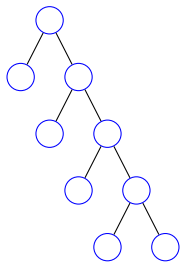
Die Laufzeitkomplexität für die Suche in einem balancierten Binärbaum ist $\mathcal{O}(\log n)$. Der hier entstandene Baum ist allerdingt nicht balanciert. Daher beträgt die Laufzeitkomplexität hier $\mathcal{O}(n)$.

Aufgabe 2 (4 Punkte): Höhe eines Baumes

1. Der in der Vorlesung besprochene Algorithmus zur Bestimmung der Höhe eines Baumes hat im schlechtesten Fall eine Laufzeit von $\Omega(n^2)$. Überlegen Sie sich, wie ein Baum aussehen könnte, auf den dieser schlechteste Fall zutrifft.
2. Zeigen Sie, dass die folgende Aussage wahr ist: In einem Binärbaum mit N Knoten ist die kleinste mögliche Höhe $\log_2(N+1) - 1$.
3. Zeigen Sie, dass die folgende Aussage wahr ist: Ein Binärbaum mit B Blättern hat mindestens $\log_2 B$ Zeilen exklusive der Wurzel.

Lösung:

1)



2)
Diese Aussage kann direkt von Beispiel 2 auf Seite 7 der Vorlesungsnotizen abgeleitet werden:

> Beweis der 2. Ungleichung (ein Binärbaum der Höhe h fasst maximal $n = 2^{h+1} - 1$ Knoten): $n = \sum_{i=0}^{h} 2^h = 2^{h+1} - 1$; siehe die *Endliche geometrische Reihe* in im Kapitel *Rekursion*.

$N \leq 2^{(h+1)} - 1$ kann so umgeformt werden, dass der o.g. Term $h \geq \log_2(N+1) - 1$ entsteht - und dass dieser wahr ist, wird bereits in den Vorlesungsnotizen bewiesen.

3)
Ein Binärbaum erreicht seine maximale Anzahl an Blättern (und minimale Anzahl an Zeilen), wenn alle Zeilen komplett befüllt sind. Gehen wir davon aus, dass sich für einen solchen Baum alle Blätter auf der letzten Zeile l befinden (zur Erinnerung: die Wurzel befindet sich auf Zeile 0). Dann ist die folgende Aussage wahr für L Blätter:

$$L = 2^l \tag{1}$$
$$\log_2 L = l \tag{2}$$

Die Entfernung von Knoten führt natürlicherweise zur Verminderung der Anzahl an Blättern. Daher ist der oben beschriebene Baum der Baum mit der geringsten Höhe bei L Blättern, also $\log_2 L \leq l$.

Aufgabe 3 (2 Punkte): Beiten-Traversierung

Im angehängten File `BreadthFirstIterator.java` finden Sie das Grundgerüst eines Iterators über einen Binärbaum. Vervollständigen Sie den Code an den mit `TODO` gekennzeichneten Stellen, sodass die Knoten in der Reihenfolge der Breiten-Traversierung ausgegeben werden. Verwenden Sie für die Implementierung, wie in der Vorlesung besprochen, eine Warteschlange.
Lösung:
Siehe `BreadthFirstIteratorSolution.java`

Aufgabe 4 (2 Punkte): Prä- und Postorder-Traversierung

Zeichnen Sie einen Binärbaum, der zugleich die folgenden drei Eigenschaften erfüllt:

- Jeder interne Knoten des Baumes soll genau einen Buchstaben enthalten.
- Die Präorder-Traversierung des Baumes ergibt **`OSTERFERIEN`**.
- Die Postorder-Traversierung des Baumes ergibt **`ERTFSIRNEEO`**.

Lösung: