Philipps-Universität Marburg
Fachbereich Mathematik und Informatik
Arbeitsgruppe Datenbanksysteme

Sommersemester 2021
31. 05. 2021

Prof. Dr. Bernhard Seeger
Nikolaus Glombiewski M.Sc.
Amir El-Shaikh

Übungen zur Vorlesung
Algorithmen und Datenstrukturen

Abgabe: 06. 06. 2021,
bis **spätestens** 23:59 Uhr
über die ILIAS Plattform

Übung 7

Aufgabe 7.1: Binärbaum Theorie (2+2+1+2+3) (10 Punkte)

a) Zeichnen Sie alle möglichen Binärbäume mit den Knoten a, b, c bei welchen a die Wurzel ist.

b) Ein Binärbaum kann unter Verwendung der post- und inorder Reihenfolge eindeutig rekonstruiert werden. Geben Sie einen Algorithmus in *Pseudocode* dafür an. Sie können davon ausgehen, dass Elemente im Baum nur einmal vorkommen. Die Eingabe des Algorithmus ist die In-Order und Post-Order Reihenfolge der Elemente. Als Ausgabe dient ein Baum, bei welchem Wurzel, linker Teilbaum und rechter Teilbaum spezifiziert ist.

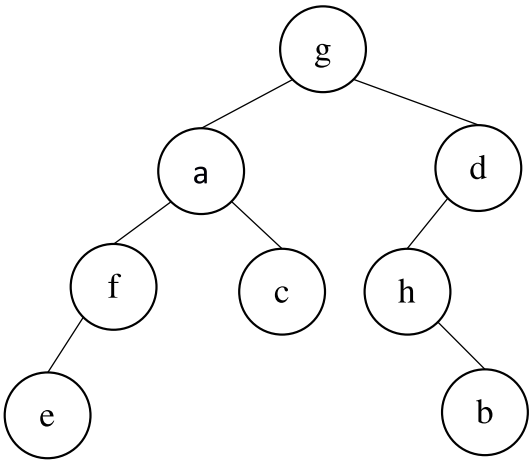
c) Widerlegen Sie mit einem Gegenbeispiel, dass ein Binärbaum unter Verwendung der post- und preorder Reihenfolge eindeutig rekonstruiert werden kann.

d) In einem Binärbaum sei b die Anzahl von Knoten mit Grad 2 und z die Anzahl von Blättern. Zeigen Sie durch Induktion, dass in einem nicht-leeren Binärbaum folgende Eigenschaft gilt: $b + 1 = z$.

e) Wir definieren L als die Menge von Blättern in einem Binärbaum. Ferner definieren wir für ein Blatt $l \in L$ mit Höhe h in einem Binärbaum $w(l) = \frac{1}{2^{h-1}}$. Zeigen Sie: $\sum_{l \in L} w(l) \leq 1$.

Aufgabe 7.2: Binärbaum Implementierung (1+2+2) (5 Punkte)

a) Verwenden Sie die Klasse `BinTree` um den unten abgebildeten Baum aufzubauen.



b) Implementieren Sie die Methoden `inOrder`, `postOrder` und `preOrder`.

c) Implementieren Sie die Methode `levelOrder`, die die Elemente des Baums Level für Level (bei der Wurzel beginnend) von links nach rechts traversiert und ausgibt.

Aufgabe 7.3: Heaps* (1+1+1+2) **(5 Punkte)**

a) Wieviele Elemente kann ein Heap mit Höhe h maximal und minimal aufnehmen? Begründen Sie Ihre Antwort.

b) Erstellen Sie eine Klasse `HeapStack`, in welcher Sie das gegebene `SimpleStack` Interface umsetzen, indem Sie einen Stack auf Basis eines Heaps implementieren. Verwenden Sie die `java.util.PriorityQueue` Klasse zur Verwaltung der Elemente in einem Heap. Beachten Sie bei Ihrer Implementierung die maximale Kapazität des Stacks.

c) Erstellen Sie eine Klasse `HeapQueue`, in welcher Sie das gegebene `SimpleQueue` Interface umsetzen, indem Sie eine FIFO (First-In First-Out) Queue auf Basis eines Heaps implementieren. Verwenden Sie die `java.util.PriorityQueue` Klasse zur Verwaltung der Elemente in einem Heap. Beachten Sie bei Ihrer Implementierung die maximale Kapazität der Queue.

d) In Experimenten, vergleichen Sie Ihre Implementierungen aus b) und c) mit den Implementierungen `java.util.Stack` (als Vergleich zum `HeapStack`) und `java.util.ArrayDeque` (als Vergleich zur `HeapQueue`). Erklären Sie Ihre Ergebnisse.

*** Aufgabe 7.3 ist für Lehramtsstudierende optional.**