Algorithmen und Datenstrukturen
Wintersemester 20/21
Prof. Dr. Georg Schied

Aufgabenblatt 7

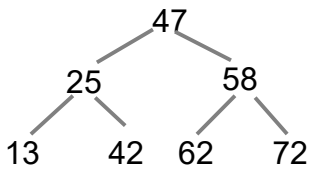
Abgabetermin: Do. 26. November 2020, 23:59 Uhr

Zum Bestehen müssen 10 von 20 Punkten erreicht werden.

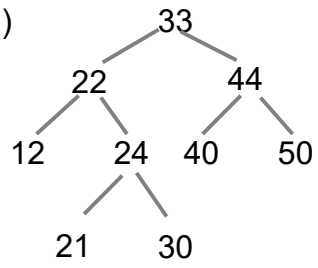
Aufgabe 7.1

Welche der folgenden Bäume sind Suchbäume?

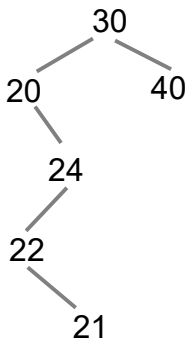
(1)



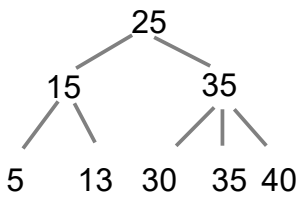
(2)



(3)



(4)



Aufgabe 7.2 - Scheinaufgabe (5 P)

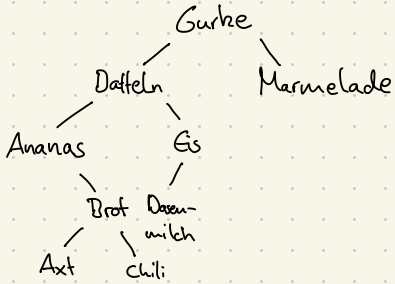
Binäre Suchbäume sollen dazu eingesetzt werden, um Artikelbezeichnungen zu speichern. Als Ordnungskriterium wird die übliche lexikographische Sortierung verwendet.

a) Fügen Sie nacheinander die Bezeichnungen `Gurke`, `Datteln`, `Ananas`, `Brot`, `Marmelade`, `Eis`, `Dosenmilch`, `Axt`, `Ente` und `Chili` in einen am Anfang leeren Suchbaum ein.

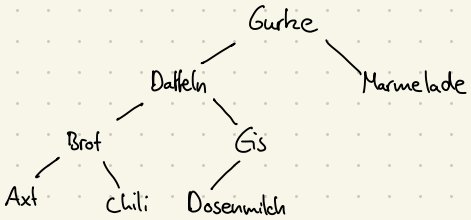
b) Löschen Sie aus dem Baum, der bei a) entstanden ist, nacheinander die Einträge `Ananas`, `Datteln` und `Gurke`.

Aufgabe 7.2

a)

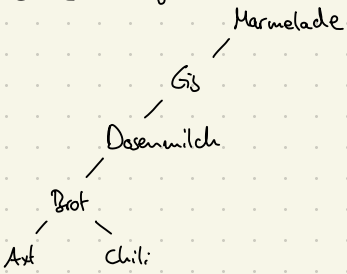


b.) Ananas wird gelöscht:



Datteln wird gelöscht:

Gurke wird gelöscht:



Aufgabe 7.3 - Scheinaufgabe (15 P)

In Moodle finden Sie die Klassen `SearchTree` und `TreeNode` mit der Implementierung eines binären Suchbaums für Werte vom Typ `int`.

a) Erweitern Sie die Klasse `SearchTree` um folgende Methoden:

`public int sum()`

Berechnet die Summe aller Werte im Baum.

`public int leaves()`

Bestimmt die Anzahl der Blätter des Baums

`public boolean insertIter(double v)`

Fügt den Wert `v` *iterativ* in den Suchbaum ein. Liefert `true` als Ergebnis, falls der Wert noch nicht enthalten war und `false`, falls er vorher schon eingetragen war.

`public int extractMin()`

Entfernt den Knoten mit dem kleinsten Wert aus dem Baum und gibt dessen Wert zurück. Liefert eine `RuntimeException`, falls der Baum leer ist.

`public ArrayList<Integer> toSortedList()`

Liefert die im Baum gespeicherten Werte *aufsteigen sortiert* als `ArrayList` (Paket `java.util`) zurück.

`public boolean equals(SearchTree other)`

Prüft, ob der Baum `other` genau die gleichen Werte enthält, unabhängig von der Struktur des Baums. Die Prüfung sollte effizient sein.

b) Ein Suchbaum mit den Operation `insertIter(v)` und `extractMin()` kann auch als Implementierung für Prioritätswarteschlangen verwendet werden. Welche Größenordnung haben die Laufzeiten für die Methoden `insertIter(v)` und `extractMin()` im mittleren und im schlechtesten Fall, abhängig von der Anzahl *n* der Einträge im Baum?

c) Welche Laufzeit hat Ihre Operation `equals` im mittleren und schlechtesten Fall? Gehen Sie dabei davon aus, dass beide Bäume jeweils *n* Werte enthalten.

In Moodle finden Sie dazu Programmvorlagen und eine JUnit-Testklasse `JuTestSearchTree`. In Klasse `SearchTreeDemo` finden Sie weitere Verwendungsbeispiele sowie Methoden, um die Laufzeiten für b) und c) mit Zufallswerten messen zu können.