Technische Hochschule Rosenheim
Technical University of Applied Sciences

Übung 07: Binäre Suchbäume

Aufgabe 1: Binäre Suchbäume

a) Gegeben sei die Menge der Schlüssel $\{1, 4, 5, 10, 16, 17, 21\}$. Zeichnen Sie sowohl einen binären Suchbaum der Höhe 2 als auch der Höhe 6, der diese Schlüssel enthält.

b) Zeichnen Sie einen *MinHeap*, der alle diese Schlüssel enthält!

c) In welcher Reihenfolge besucht man die Knoten des ***binären Suchbaums*** der Höhe 2 von ***a)***, falls *Preorder* bzw. *Inorder* bzw. *Postorder* verwendet wird.

d) Zeichnen Sie den binären Suchbaum, der sich ergibt wenn man der Reihe nach die folgenden Schlüssel einfügt: `E A S Y Q U T I O`
Hinweis: Gehen Sie davon aus, dass z.B. der Buchstabe `E` größer ist als der Buchstabe `A`.

e) Löschen Sie die Schlüssel aus dem Ergebnis von a) in der Reihenfolge des Einfügens, also:
`E A S Y Q U T I O`
Zeichen Sie den binären Suchbaum nach ***jedem*** Löschvorgang und geben Sie an ob es sich um den Fall 1, 2a, 2b, 3a oder 3b handelt, siehe Vorlesung!

f) Bei der Preorder-Traversierung eines binären Suchbaums ergibt sich die folgende Besuchsreihenfolge: `20, 10, 5, 1, 7, 15, 30, 25, 35, 32, 40`
Zeichnen Sie den dazugehörigen binären Suchbaum und erklären Sie knapp wie Sie auf das Ergebnis kommen.

Aufgabe 2: Löschen und Höhe

Die Klasse `BST.java` und sowie eine JUnit Testklasse ist vorgegeben[1].

a) Implementieren Sie die folgende Methode `public void delete(Key k),` die den Schlüssel *k* aus dem binären Suchbaum löscht:
Hinweise:
- Zuerst den Knoten `z` suchen, der den Schlüssel `k` speichert.
- Falls der zu löschendes Schlüssel nicht enthalten ist, bleibt der Baum unverändert.
- Orientieren Sie sich unbedingt am Pseudocode der Vorlesung, berücksichtigen Sie alle Fälle!
- Ändern Sie nur Referenzen wie `left`, `right` und `parent` eines Knotens. Es darf **kein neuer Knoten mit `new(.)`** erzeugt werden.
- Verwenden Sie die vorgegebene Methoden `transplant(.)` und `min(.)`.

b) Die bereitgestellte JUnit-Testklasse fügt der Reihe nach die folgenden Schlüssel in den Baum ein: <7, 6, 10, 13, 12, 16, 23, 18, 20, 15, 5, 3>. Zeichnen Sie den sich ergebenden Baum.

c) Testen Sie Ihre Implementierung aus a) mit JUnit!
Wie sieht der Baum aus b) aus, falls man die 3, die 10 bzw. die 16 jeweils aus dem ***Ausgangsbaum*** löscht. Die JUnit-Klasse gibt die Knoten in Level-Order Reihenfolge aus.

d) Implementieren Sie die folgende ***rekursive*** Methode:
`public int height(Node x) {`
Die Methode berechnet die Höhe des Teilbaumes, der den Knoten *x* als Wurzel hat. Unter der Höhe versteht man die Anzahl der Kanten auf dem längsten Pfad, der vom Knoten x zu einem Blatt des Teilbaumes führt. Testen Sie mit dem JUnit!

[1] Im GitLab gibt es sowohl ein komplettes IntelliJ Projekt als auch die Einzeldateien