Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. Fabian Kuhn
P. Bamberger, Y. Maus, P. Schneider

Informatik 2 - Sommersemester 2018
Übungsblatt 6

Abgabe: Montag, 11. Juni, 14:00 Uhr

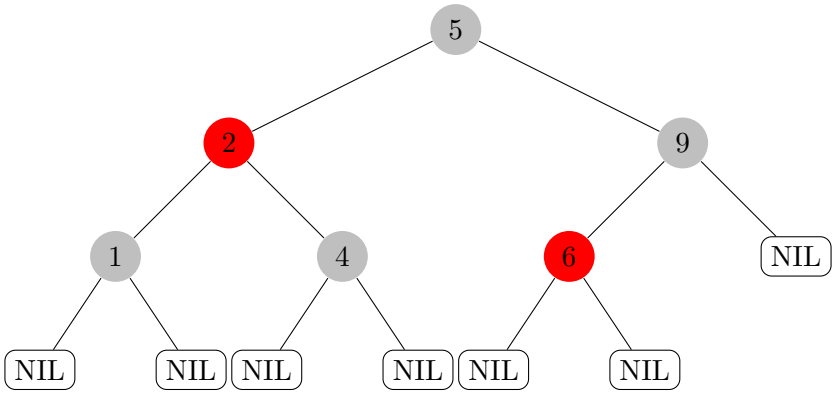
Aufgabe 1: Post-Order Traversierung *(6 Punkte)*

(a) Gegeben seien die Schlüssel $5, 7, 8, 9, 11, 12, 14, 15, 18$. Geben Sie eine Folge von `insert`(key) Operationen an, so dass ein binärer Suchbaum entsteht, bei dem man die Knoten mit der Post-Order Traversierung in der Reihenfolge $7, 5, 9, 8, 12, 15, 18, 14, 11$ besucht. *(3 Punkte)*

(b) Beschreiben Sie ein Verfahren, mit dem man aus der Post-Order gegeben als Array $A[0, \ldots, n-1]$ den zugehörigen binären Suchbaum (mit n Knoten) rekonstruieren kann. Sie können dazu Pseudocode verwenden. Gehen Sie davon aus, dass alle Schlüssel paarweise verschieden sind. *(3 Punkte)*

Aufgabe 2: Rot-Schwarz Bäume *(6 Punkte)*

Gegeben sei folgender Rot-Schwarz-Baum. Führen Sie zuerst die Operation `insert`(8) und danach `delete`(4) aus. Zeichnen Sie den resultierenden Baum. Dokumentieren Sie Ihre Zwischenschritte.



Aufgabe 3: Treaps *(8 Punkte)*

(a) Implementieren Sie einen *Treap* wie in der Vorlesung beschrieben. Sie *können* dazu die Vorlage `Treap.py` benutzen welche Ihnen bereits grundlegende Funktionen zur Verfügung stellt, wobei lediglich die Funktionen `rotate_right`, `rotate_left`, `delete` und `repair_heap` implementiert werden müssen. *(4 Punkte)*

(b) Vergleichen Sie Ihre Implementierung des binären Suchbaums vom vorherigen Übungsblatt (oder die von uns zur Verfügung gestellte `BinarySearchTree.py`) mit Ihrer Implementierung des Treap. Tun Sie dies indem Sie die Suchschlüssel $(0, 1, \ldots, 2^k)$ in dieser Reihenfolge für $k \in \{1, \ldots, 9\}$ in beide Datenstrukturen einfügen. Anschließend sollen Sie in einem Schaubild die Höhe der beiden Binärbaume über der Anzahl der eingefügten Schlüssel auftragen. Unsere Vorlage bietet dafür eine Funktion an welche Sie benutzen *können*. *(4 Punkte)*