Übung 3: Datenstrukturen

Aufgabe 3-1:

Implementieren Sie eine einfach verkettete Liste zum sortierten Speichern von Integerwerten. Im Moodle-Kurs steht eine Header-Datei und ein Rahmenprogramm zum Download bereit.

Aufgabe 3-2:

Ergänzen Sie die aus der Vorlesung bekannte doppelt verkettete Liste um eine Umordnung der Elemente: Platziere die Elemente, auf die häufig zugegriffen wird, weit vorne in der Liste. Wir gehen davon aus, dass die Zugriffshäufigkeiten nicht im voraus bekannt sind.

Implementieren Sie eine solche selbstanordnende, lineare Liste mit der Move-To-Front-Regel. Wenn die letzte Suchanfrage auf das Element e zugegriffen hat, dann stellt die MF-Regel das Element e an den Anfang der Liste.

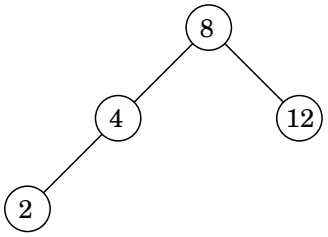
Schreiben Sie ein Programm, dass $2^{15}, 2^{16}, \ldots, 2^{25}$ viele Werte in die Liste einfügt und dann 80% der Suchanfragen auf nur 20% der Daten zugreifen. Stellen Sie Unterschiede bei den Laufzeiten zwischen der „normalen“ Liste und der selbstanordnenden Liste fest?

Aufgabe 3-3:

Geben Sie einen rekursiven Algorithmus an, der die Inhalte einer Liste in die umgekehrte Reihenfolge bringt. (Siehe Anforderungen der GI.)

Aufgabe 3-4:

Geben Sie den resultierenden Baum an, der aus dem folgenden Suchbaum entsteht, wenn der Wert 3 eingefügt wird. Um den resultierenden Baum zu balancieren, muss zunächst eine Links-Rotation bei 2 und anschließend eine Rechts-Rotation bei 4 durchgeführt werden. Zeichnen Sie jeweils den resultierenden Suchbaum.



Aufgabe 3-5:

Fügen Sie nacheinander die Schlüssel 4,2,6,1,3,5,7 in dieser Reihenfolge in einen anfangs leeren Splay-Baum ein. Führen Sie anschließend eine Suche nach dem Schlüssel 3 aus und löschen Sie schließlich den Wert 5 aus dem Baum. Geben Sie nach jedem Schritt den resultierenden Splay-Baum an.

Aufgabe 3-6:

Implementieren Sie einen Splay-Baum. Welche Laufzeiten ergeben sich beim Einfügen von $2^{15}, 2^{16}, \ldots, 2^{25}$ zufälligen Werten? Ändert sich das Laufzeitverhalten, wenn aufsteigend sortierte Werte eingefügt werden? Im Moodle-Kurs steht ein Rahmenprogramm zum Download bereit.

Aufgabe 3-7:

Bei Hash-Verfahren ist es wichtig, eine gute Hash-Funktion zu verwenden. Die multiplikative Methode verwendet die Funktion

$$h(k) = (a \cdot k \bmod 2^w) \div 2^{w-r}$$

wobei $\div$ die ganzzahlige Division ist, also $a \div b = \lfloor a/b \rfloor$. Bei welchem Index würde der Schlüssel $k = 19$ mit den Parametern $w = 7$, $r = 4$ und $a = 97$ der Hash-Funktion abgespeichert werden?

Aufgabe 3-8:

Betrachten Sie die Hashfunktion $h(k) = k \bmod m$ mit $m = 13$. Fügen Sie die Werte

$$16, 33, 47, 50, 42, 35, 59, 31, 7$$

in eine anfangs leere Hash-Tabelle ein. Auftretende Kollisionen sollen durch

(a) verkettete Listen,
(b) lineares Sondieren,
(c) quadratisches Sondieren bzw.
(d) double hashing mit $h_2(k) = 1 + k \bmod 11$

aufgelöst werden. Geben Sie die Einträge nach jedem Einfügen eines Wertes an.