DAP2 – Heimübung 10

Ausgabedatum: 16. 6. 17 — Abgabedatum: Fr. 23. 6. 17 (Mo. 26. 6. für Gruppen 27–32) 12 Uhr

Schreiben Sie unbedingt immer Ihren **vollständigen Namen**, **Ihre Matrikelnummer** und **Ihre Gruppennummer** auf Ihre Abgaben!

Aufgabe 10.1 (5 Punkte): (Datenstrukturen)

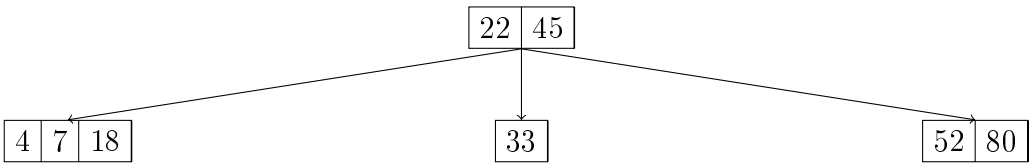
a) Ein Binärbaum mit der Schlüsselmenge $\{A, B, C, K, L, M, N, R, S, W, X\}$ ist eindeutig durch die folgenden Durchläufe gegeben:

Preorder-Durchlauf: **M B C N K X L A W S R**

Postorder-Durchlauf: **N K C X B A S R W L M**

Zeichnen Sie den gesuchten Binärbaum und geben Sie die **Inorder-Reihenfolge** der Schlüssel in diesem Binärbaum an.

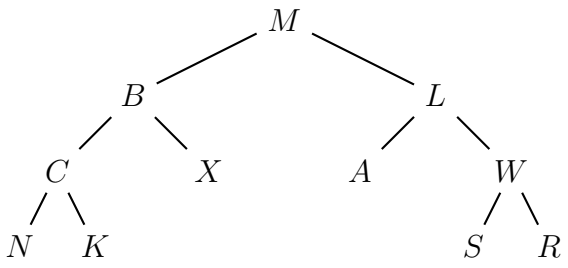
b) Gegeben sei der folgende B-Baum T mit Parameter $t = 2$:



Fügen Sie in der angegebenen Reihenfolge die Knoten $40, 42, 6, 20$ ein. Geben Sie für jede Operation die ggf. nötigen Split-Operationen sowie den resultierenden B-Baum an.

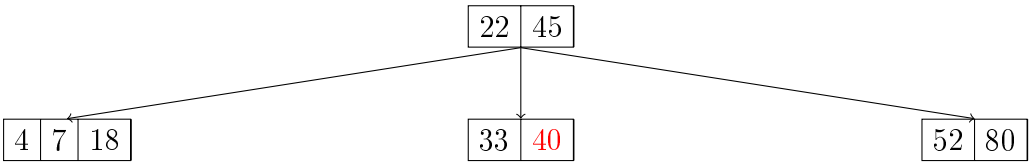
Lösung:

a) Inorder-Durchlauf lautet **N C K B X M A L S W R**. Entsprechender Baum ist

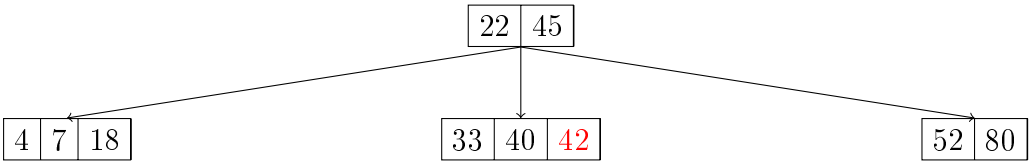


b) Vor jedem Einfügen führen wir kurz die Suche nach dem Element durch, um herauszufinden, ob ein Split nötig ist.

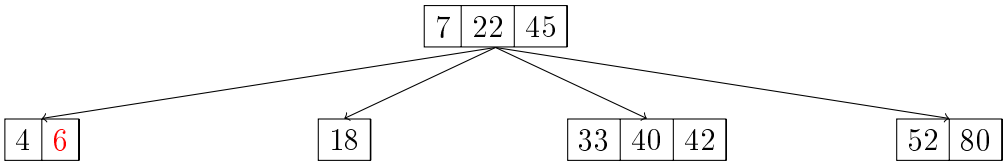
1) Auf dem Suchpfad zur 40 (Wurzel, mittleres Kind) liegt kein Knoten mit der maximalen Schlüsselzahl von $2t - 1 = 3$ Schlüsseln, es muss also kein Split durchgeführt werden. Wir fügen 40 einfach in das mittlere Blatt ein:



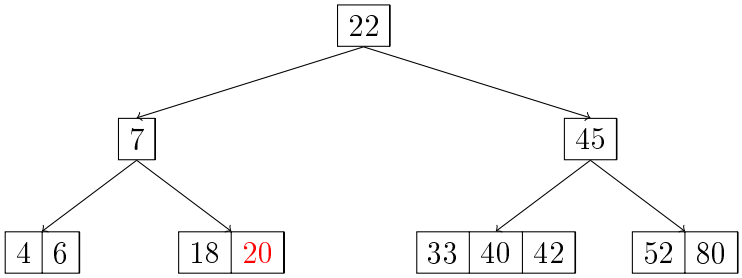
2) Auf dem Suchpfad zur 42 (Wurzel, mittleres Kind) liegt kein Knoten mit der maximalen Schlüsselzahl von $2t - 1 = 3$ Schlüsseln, es muss also kein Split durchgeführt werden. Wir fügen 42 einfach in das mittlere Blatt ein:



3) Auf dem Suchpfad zur 6 liegt die Wurzel mit zwei und das mittlere Blatt mit drei Schlüsseln, das Blatt muss also gesplittet werden. Der Median der drei Schlüssel ist 7, dieser Schlüssel wird in die Wurzel geschoben, die beiden weiteren Elemente werden zu Blättern. Anschließend wird der Schlüssel 6 zum Blatt mit dem Schlüssel 4 hinzugefügt:



4) Auf dem Suchpfad zum Schlüssel 20 finden wir diesmal die Wurzel mit drei Schlüsseln, diese muss also gesplittet werden. Anschließend wird die 20 zum Blatt mit dem Schlüssel 18 hinzugefügt:



□

Aufgabe 10.2 (5 Punkte): (Datenstrukturen)

Entwerfen Sie eine Datenstruktur, die alle folgenden Operationen in $\mathcal{O}(1)$ unterstützt. In dieser Aufgabe sind alle Elemente aus dem Wertebereich $\{1, \ldots, k\}$, **und k ist eine Konstante**.

- `Einfügen(x)`: Fügt ein Element x in die Datenstruktur ein.
- `LöscheNeuestes`: Löscht unter allen Elementen in der Datenstruktur das, was zuletzt eingefügt wurde.
- `LöscheÄltestes`: Löscht unter allen Elementen in der Datenstruktur das, was am frühesten eingefügt wurde.
- `Zähle(x)`: Gibt zurück, wie oft x aktuell in der Datenstruktur ist.

Beispiel: Nach der Sequenz: `Einfügen(7)`, `Einfügen(7)`, `Einfügen(2)`, `Einfügen(7)`, `Einfügen(4)`, `LöscheÄltestes`, `Zähle(7)` soll die Datenstruktur bei der letzten Operation 2 zurückliefern, da die 7 noch zweimal in der Datenstruktur vorhanden ist.

Beschreiben Sie in wenigen kurzen Sätzen, wie Ihre Datenstruktur aufgebaut ist und wie die angegeben Operationen realisiert werden. Hierbei ist kein Pseudocode gefordert. Machen Sie deutlich, dass die Datenstruktur korrekt arbeitet und alle Operationen Worst-Case-Laufzeit O(1) haben. Für die volle Punktzahl wird erwartet, dass **alle Operationen eine Worst-Case Laufzeit von $\mathcal{O}(1)$ haben.**

Lösung:

Die gesuchte Datenstruktur ist von eine doppel verkettete Liste A mit Zeigern `Head` und `Tail` auf dem ersten bzw. letzten Element; sowie ein Array $B[1..k]$ gebaut. Da k eine Konstante ist, ist das möglich zu definieren. Im Feld $B[x]$ des Arrays B wird die Anzahl der Auftretens von x in A gespeichert. Jedes Element von A hat ein Schlüssel (Wert), und zwei Zeigern `prev` und `next`. Am Anfang ist die Liste A leer (`head`=`tail`=`null`) und $B[i] = 0$ für alle $1 \leq i \leq k$.

Jetzt beschreiben wir, wie die verlangten Operationen in der Laufzeit $O(1)$ ausgeführt werden:

- `Einfügen(x)`: Ein Element x wird in die Liste in $O(1)$ eingefügt, so dass der Zeiger `tail` jetzt auf x zeigt ($O(1)$), und $B[x]$ wird um 1 erhöht ($O(1)$).
- `LöscheNeuestes`: Wir löschen das letzten Element `tail` mit dem Schlüssel x =Wert(`tail`) ($O(1)$). Der Zeiger `tail` wird nach `tail`→`prev` aktualisiert ($O(1)$) und $B[x]$ wird um 1 verringert ($O(1)$).
- `LöscheÄltestes`: Wie für `LöscheNeuestes`, löschen wir das ersten Element `head` mit dem Schlüssel x = Wert(`head`) ($O(1)$). Der Zeiger `head` wird nach `head`→`next` aktualisiert ($O(1)$) und $B[x]$ wird um 1 verringert ($O(1)$).
- `Zähle(x)`: Da $B[x]$ die Anzahl von x in A enthält, geben wir $B[x]$ zurück ($O(1)$).

Es ist klar, dass alle angegebene Operationen eine Worst-Case Laufzeit von $O(1)$ haben.