# Assignment 6
# Advanced Algorithms & Data Structures PS

Christian Müller 1123410
Daniel Kocher, 0926293

May 11, 2016

**Aufgabe 11**

i.) Fügen Sie die Schlüssel $f$, $g$, $h$, $e$, $b$, $a$, $c$ in einen (anfangs leeren) Treap ein. Die Prioritäten dieser Schlüssel sind wie folgt gegeben: $a : 8$, $b : 15$, $c : 2$, $e : 3$, $f : 7$, $g : 6$, $h : 25$, $i : 22$, $j : 19$, $k : 13$.

ii.) Entfernen Sie $e$ aus dem Treap.

iii.) Fügen Sie die Schlüssel $i$, $j$, $k$ in einen anderen (anfangs leeren) Treap ein. Vereinigen Sie anschließend die zwei Treaps.

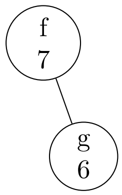
iv.) Führen Sie $\mathit{Spalte}\,(T, d, T_1, T_2)$ durch, wobei $T$ der Treap aus dem vorigen Punkt ist.

Geben Sie den Treap vor und nach jeder Rotation an.

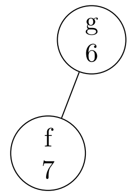
Für den Pseudocode bzw. die grundlegenden Vorgehensweise der Operationen Suchen, Einfügen, RotiereNachLinks/Rechts, Entfernen, Vereinige und Spalte, sei auf die Folien vom 14.04.2016 verwiesen.

i.) Fügen Sie die Schlüssel $f$, $g$, $h$, $e$, $b$, $a$, $c$ in einen (anfangs leeren) Treap ein. Die Prioritäten dieser Schlüssel sind wie folgt gegeben: $a : 8$, $b : 15$, $c : 2$, $e : 3$, $f : 7$, $g : 6$, $h : 25$, $i : 22$, $j : 19$, $k : 13$.

Insert $g_6$:

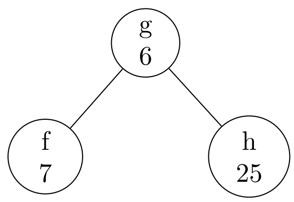



(a) Vorher



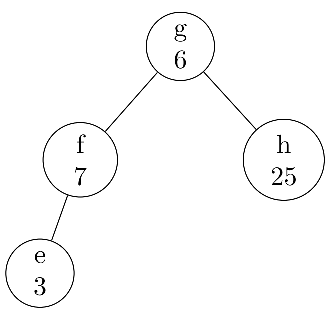

(b) Nach RotiereNachLinks($f_7$)

Insert $h_{25}$:

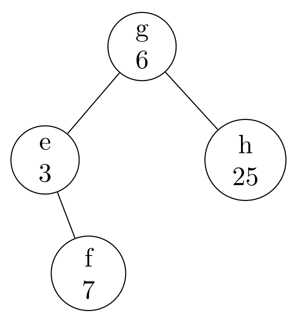



(a) Keine Rotation notwendig

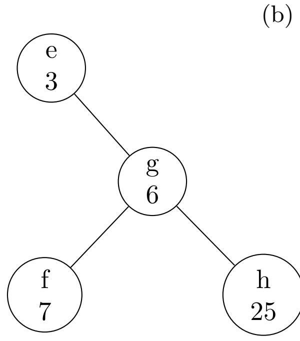
Insert $e_3$:




(a) Vorher

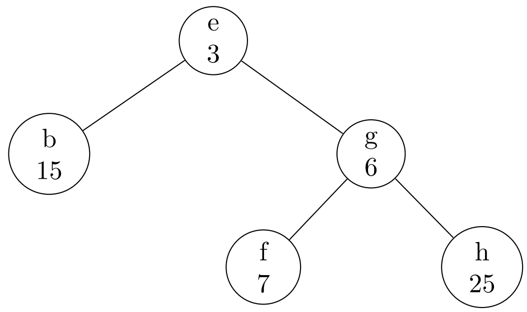



(b) Nach RotiereNachRechts($f_7$)



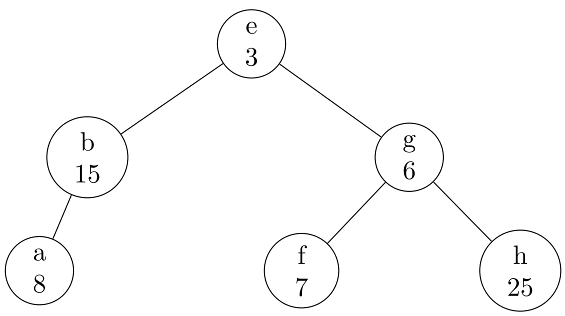

(c) Nach RotiereNachRechts($g_6$)

Insert $b_{15}$:

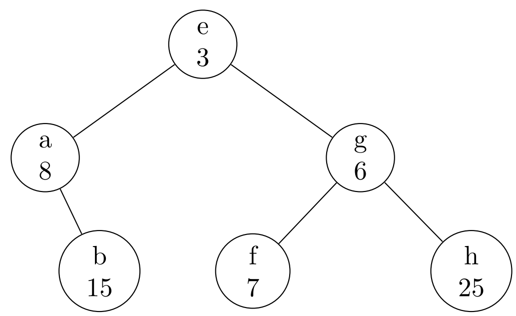



(a) Keine Rotation notwendig

Insert $a_8$:

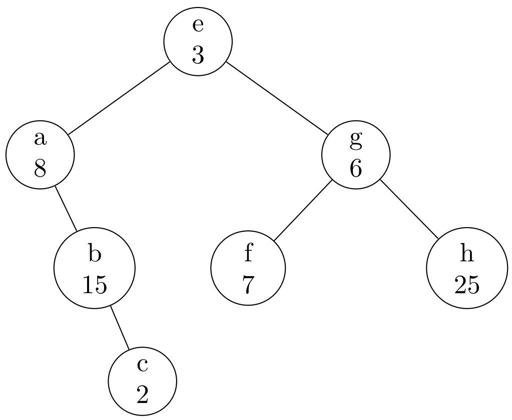



(a) Vorher

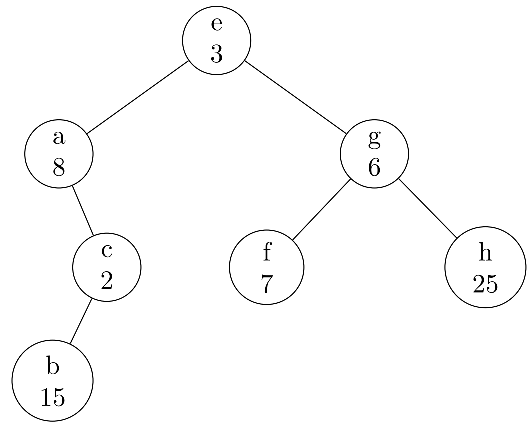



(b) Nach RotiereNachRechts($b_{15}$)

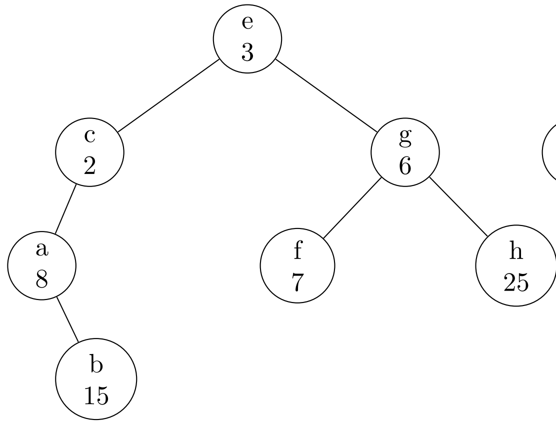
Insert $c_2$:



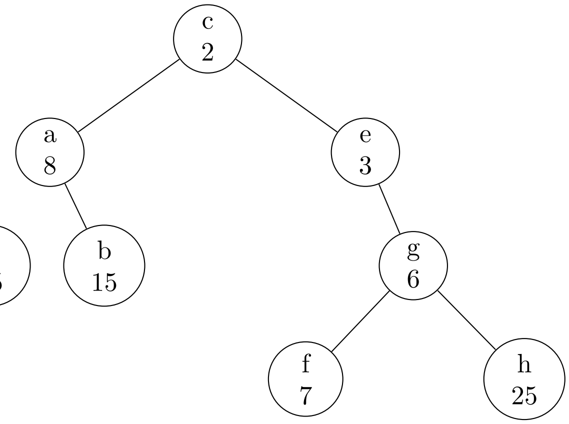

(a) Vorher




(b) Nach RotiereNachLinks($b_{15}$)

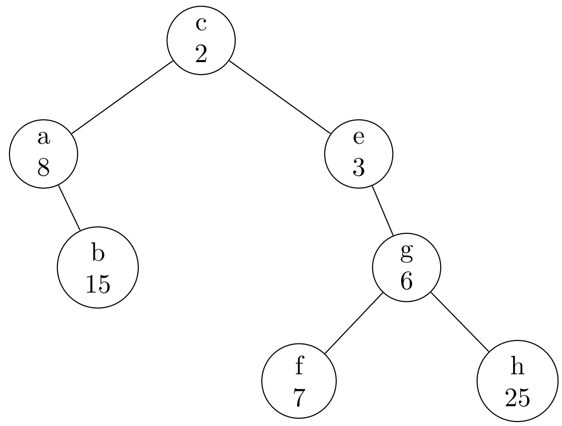



(c) Nach RotiereNachLinks($a_8$)

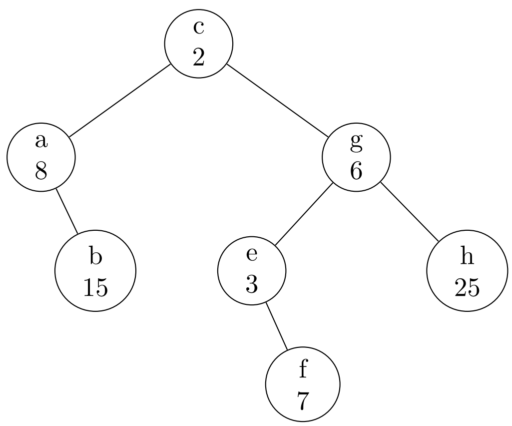



(d) Nach RotiereNachRechts($e_3$)

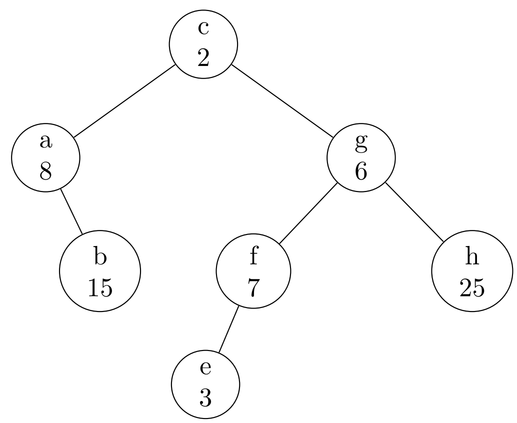
ii.) Entfernen Sie $e$ aus dem Treap.



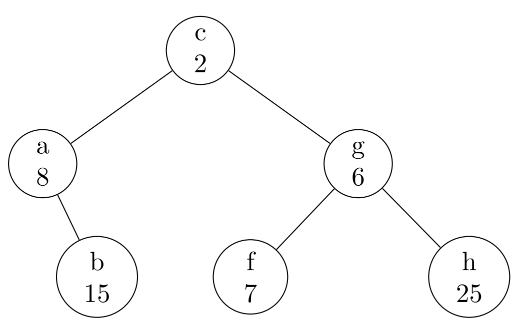

(a) Start




(a) Nach: $g_6$ ist rechtes Kind von $e_3$ $\Longrightarrow$ RotiereNachLinks($e_3$)



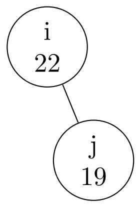

(b) Nach: $f_7$ ist rechtes Kind von $e_3$ $\Longrightarrow$ RotiereNachLinks($e_3$)



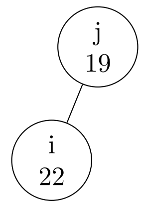

(a) Ende

iii.) Fügen Sie die Schlüssel $i_{22}$, $j_{19}$, $k_{13}$ in einen anderen (anfangs leeren) Treap ein. Vereinigen Sie anschließend die zwei Treaps.

Insert $j_{19}$:

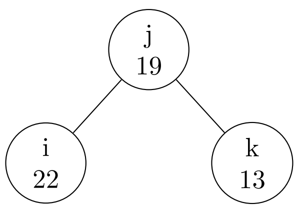



(a) Vorher

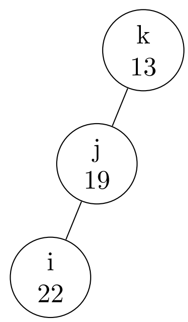



(b) Nach RotiereNachLinks($i_{22}$)

Insert $k_{13}$:

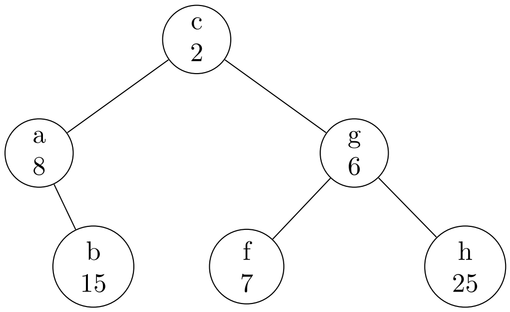



(a) Vorher

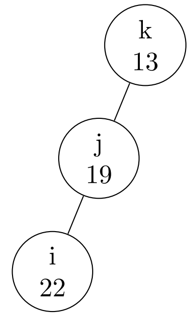



(b) Nach RotiereNachLinks($j_{19}$)

Vereinige($T_1$, $T_2$):

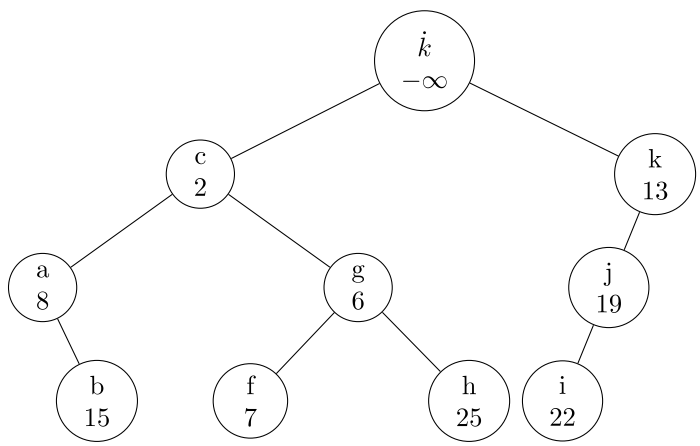



(a) $T_1$



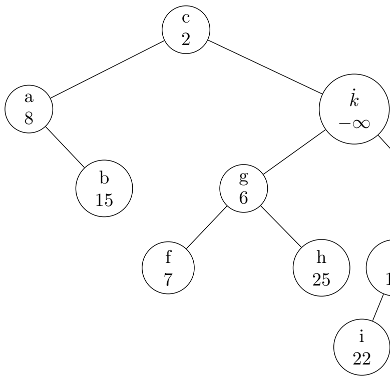

(b) $T_2$

Sei $\dot{k}$ ein Schlüssel mit $\text{key}(x_1) < \dot{k} < \text{key}(x_2)$ für alle $x_1 \in T_1$ und $x_2 \in T_2$. Ein echter Buchstabe kann hier nicht verwendet werden, da ein solcher im deutschen Alphabet (natürliche Ordnung) nicht existiert. Es gilt also: $a < b < c < ... < \dot{k} < i < j < k < ... < z$.

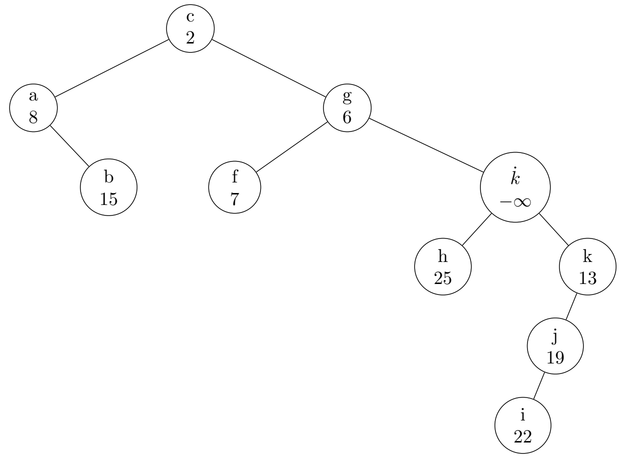



(a) Neuer Knoten mit Schlüssel $\dot{k}$ als Wurzel.

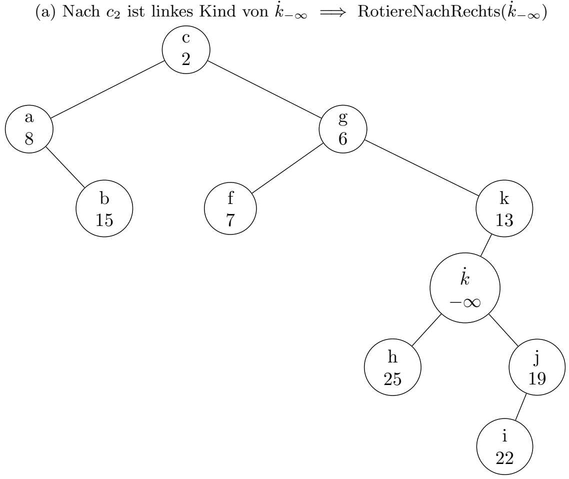
Entferne Wurzel aus Treap:



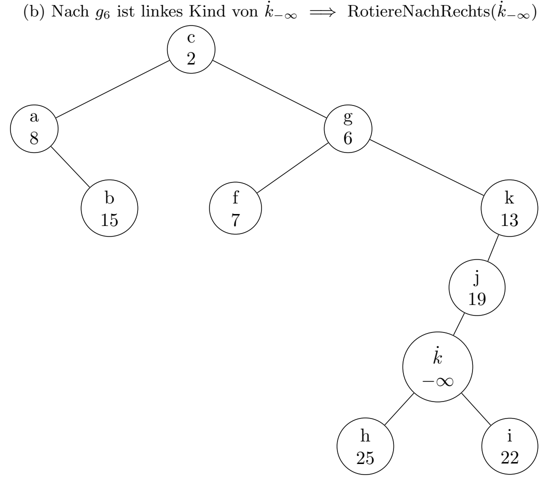

(a) Nach $c_2$ ist linkes Kind von $\dot{k}_{-\infty}$ $\implies$ RotiereNachRechts($\dot{k}_{-\infty}$)



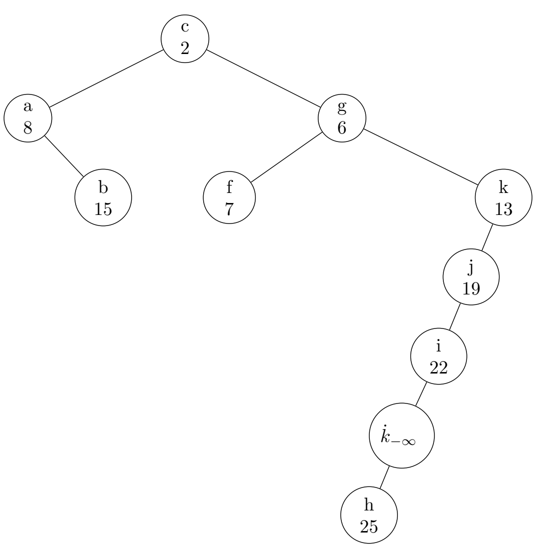

(b) Nach $g_6$ ist linkes Kind von $\dot{k}_{-\infty}$ $\implies$ RotiereNachRechts($\dot{k}_{-\infty}$)



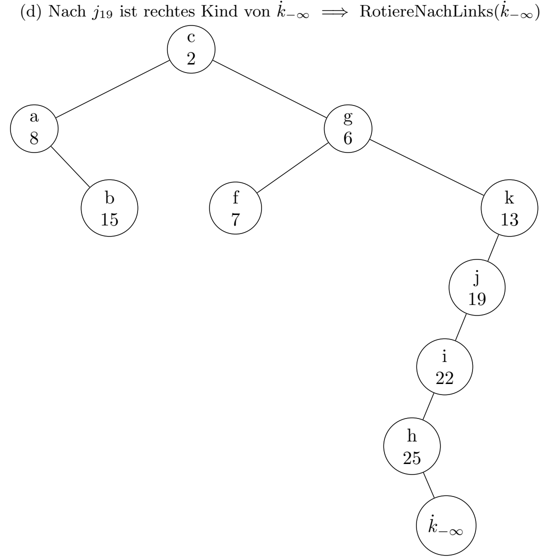

(c) Nach $k_{13}$ ist rechtes Kind von $\dot{k}_{-\infty}$ $\implies$ RotiereNachLinks($\dot{k}_{-\infty}$)




(d) Nach $j_{19}$ ist rechtes Kind von $\dot{k}_{-\infty}$ $\implies$ RotiereNachLinks($\dot{k}_{-\infty}$)



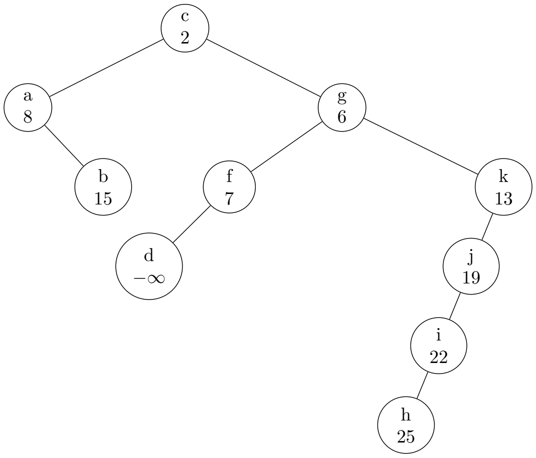

(e) Nach $i_{22}$ ist rechtes Kind von $\dot{k}_{-\infty}$ $\implies$ RotiereNachLinks($\dot{k}_{-\infty}$)



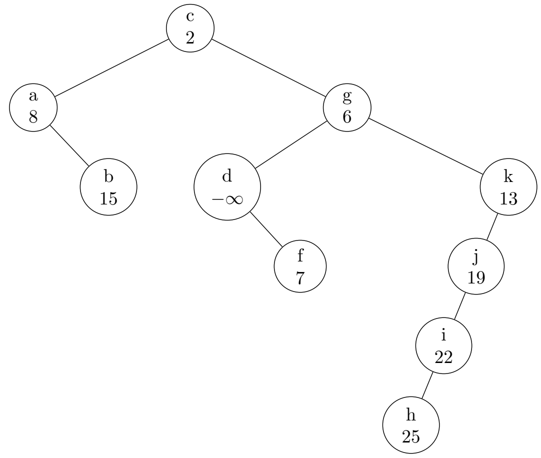

(f) Nach $h_{25}$ ist linkes Kind von $\dot{k}_{-\infty}$ $\implies$ RotiereNachRechts($\dot{k}_{-\infty}$)
Der Hilfsknoten $\dot{k}_{-\infty}$ ist jetzt ein Blatt und kann einfach entfernt werden.

iv.) Führen Sie $Spalte\,(T, d, T_1, T_2)$ durch, wobei $T$ der Treap aus dem vorigen Punkt ist.

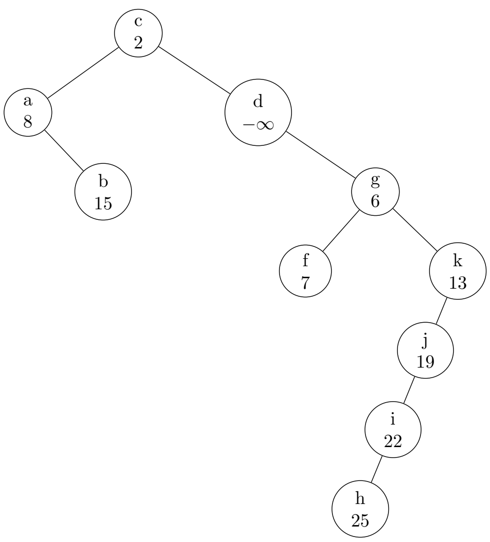
Füge Knoten $d_{-\infty}$ in T ein:



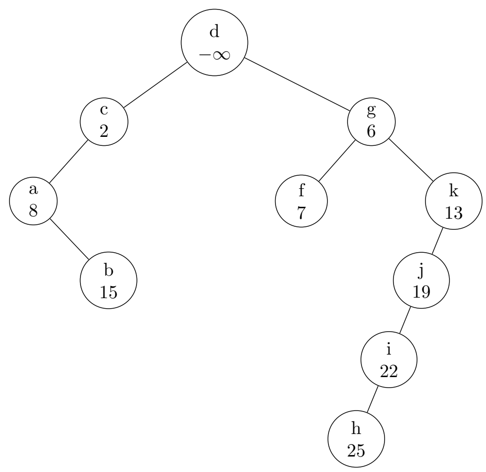

(a) Vorher



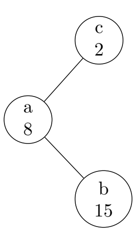

(b) Nach $d_{-\infty}$ ist linkes Kind von $f_7 \implies$ RotiereNachRechts($f_7$)



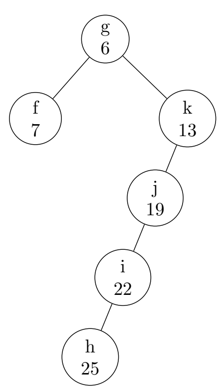

(c) Nach $d_{-\infty}$ ist linkes Kind von $g_6 \implies$ RotiereNachRechts($g_6$)




(d) Nach $d_{-\infty}$ ist rechtes Kind von $c_2 \implies$ RotiereNachLinks($c_2$)




(e) $T_1$




(f) $T_2$

**Aufgabe 12**

Der *linke Rand* in einem binären Suchbaum $T$ ist der Pfad von der Wurzel zum Knoten mit dem kleinsten Schlüssel. Der *rechte Rand* in einem binären Suchbaum $T$ ist der Pfad von der Wurzel zum Knoten mit dem größten Schlüssel. Betrachten Sie einen Treap $T$ direkt nach dem Einfügen eines Objektes $x$. Sei $C$ die Länge des rechten Randes des linken Unterbaums des Knotens mit dem Element $x$ und sei $D$ die Länge des linken Randes des rechten Unterbaums des Knotens mit dem Element $x$. Zeigen Sie, dass die Anzahl der Rotationen, die während des Einfügens von $x$ durchgeführt wurden, $C + D$ ist.

Linker Rand eines binären Suchbaums: von der Wurzel zum linkesten Kind, d.h. verfolge ausgehend von der Wurzel immer die Kante zum linken Kind.
Rechter Rand eines binären Suchbaums: analog (verfolge immer die Kante zum rechten Kind).

$C$ ... Länge des rechten Randes des linken Unterbaums des Knotens mit dem Element $x$.
$D$ ... Länge des linken Randes des rechten Unterbaums des Knotens mit dem Element $x$.
$R$ ... # der Rotationen, die während des Einfügens von $x$ durchgeführt wurden.

Zu zeigen: $C + D = R$

*Proof.* Ausgehend von der Annahme, dass $C + D = R$ vor der ersten Rotation gilt, zeigen wir nun, dass $C + D = R$ auch nach der darauffolgenden Rotation gilt.

$C + D = R$ gilt für $N$ Rotation (auch für $C = D = 0$ trivialerweise).

Seien $C$, $D$, $C_{NR}$ und $D_{NR}$ wie folgt:

$C$ ... Länge des rechten Randes des linken Unterbaums des Knotens mit dem Element $x$ *vor der Rotation.*
$D$ ... Länge des linken Randes des rechten Unterbaums des Knotens mit dem Element $x$ *vor der Rotation.*
$C_{NR}$ ... Länge des rechten Randes des linken Unterbaums des Knotens mit dem Element $x$ *nach der Rotation.*
$D_{NR}$ ... Länge des linken Randes des rechten Unterbaums des Knotens mit dem Element $x$ *nach der Rotation.*

Es gibt nun zwei Fälle zu unterscheiden.

Fall 1: Linksrotation bzgl. Knoten $x$

Vor der Rotation hat der linke Rand des rechten Unterbaums von $x$ die Länge $D$. Analog hat der rechte Rand des linken Unterbaums von $x$ die Länge $C$.

Bei einer Linksrotation passiert Folgendes. Sei $y$ der Elternknoten von $x$ vor der Linksrotation.

- $x$ wird auf die Position von $y$ geschoben und $y$ wird linkes Kind von $x$.
- Der linke Unterbaum von $x$ wird zum rechten Unterbaum von $y$.
- Der rechte Unterbaum von $x$ bleibt dessen rechter Unterbaum.
- Der linke Unterbaum von $y$ bleibt dessen linker Unterbaum.

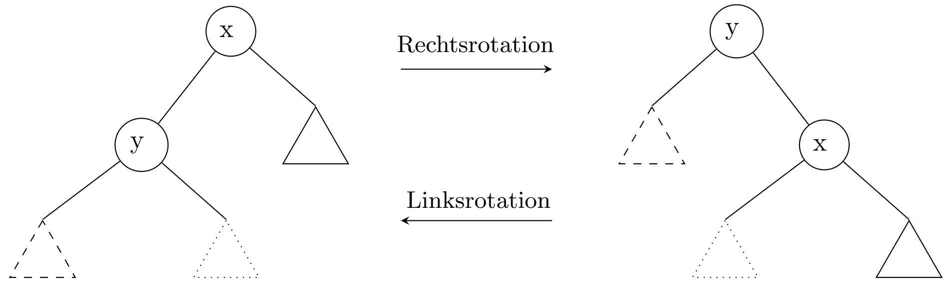



Figure 16: Links- und Rechtsrotation bzgl. $x$

Durch die "Nach-Unten-Verschiebung" des linken Unterbaums von $x$ um eine Ebene, wird der rechte Rand des linken Unterbaums des Knotens mit dem Element $x$ ebenfalls um 1 länger, d.h. $C_{NR} = C + 1$. Die Länge des linken Randes des rechten Unterbaums des Knotens mit dem Element $x$ bleibt hingegen unverändert, d.h. $D_{NR} = D$.

Für diesen Fall gilt also für $N$ Rotationen, dass $C$ $N$-mal um 1 länger wird, d.h. $C_{NR} = C + \sum_{i=1}^{N} 1 = C + N = N$. $D_{NR}$ bleibt hingegen immer gleich bleibt, d.h. $D_{NR} = D = 0$.

Daraus folgt: $C_{NR} + D_{NR} = N = N$ für $N$ Rotationen. $\square$

Fall 2: Rechtsrotation bzgl. Knoten $x$

Die selbe Argumentation kann für die Rechtsrotation verwendet werden. Der einzige Unterschied liegt in der Veränderung des Baumes. Bei einer Rechtsrotation gilt Folgendes. Sei $y$ das linke Kind von $x$ vor der Rechtsrotation.

- $y$ wird der neue Elternknoten von $x$.
- Der linke Unterbaum von $y$ bleibt dessen linker Unterbaum.
- Der rechte Unterbaum von $y$ wird linker Unterbaum von $x$.
- $x$ wird das rechte Kind von $y$.

In diesem Fall wir der Pfad $D$ um 1 länger und $C$ bleibt unverändert, d.h. $D_{NR} = D + 1$ und $C_{NR} = C$.

Analog zu Fall 1 ergibt sich daraus für $N$ Rotationen: $C_{NR} + D_{NR} = N = N$. $\square$

Das heißt, dass $C + D = R$ auch für die Kombination von Links- und Rechtsrotationen gilt, da nur entweder $D$ oder $C$ um 1 länger werden. $\square$

**Aufgabe 13**

Sei $U = \{0, \ldots, N-1\}$, wobei $N$ eine Primzahl ist und sei $m = 4$. Seien $a_i = 40i$ und $b_i = 60i$. Wir definieren folgende Klasse von Hashfunktionen:

$$H = \Big\{h_i(k) = \big((a_i k + b_i) \mod N-1\big) \mod m\Big\} \text{ für } i \in \big\{1, \ldots, N(N-1)\big\} \tag{1}$$

Ist $H$ universell? Warum? Falls $H$ nicht universell ist, so modifizieren Sie $h_i$, $a_i$ und $b_i$, sodass Sie eine universelle Klasse erhalten.

$H$ ist nicht universell.

*Proof.* Da $N$ prim ist, wissen wir, dass (1) $(N-1)$ keine Primzahl ist (außer $N=3$) und (2) $(N-1)$ eine gerade Zahl ist (da alle Primzahlen außer 2 ungerade sind).

Weiters wissen wir, dass $a_i = 40i$ und $b_i = 60i$ beides gerade Zahlen sind, da eine Multiplikation mit einer geraden Zahl immer eine gerade Zahl ergibt.

Dies hat zur Folge, dass $H = \left\{h_i(k) = (40ik + 60i) \mod N-1 \mod 4\right\}$ immer eine gerade Zahl ergibt (gerade Zahl mod gerader Zahl ergibt immer gerade Zahl). Dadurch wird nur der halbe Bereich von $m$ ausgenutzt (nämlich nur jede zweite - gerade - Zahl), d.h. nur $\frac{m}{2}$.

Damit $H$ universell ist, muss nun Folgendes für $(x, y)$ mit $x \neq y$ erfüllt sein:

$$\frac{|\left\{h \in H : h(x) = h(y)\right\}|}{|H|} \leq \frac{1}{m} \Longleftrightarrow |\left\{h \in H : h(x) = h(y)\right\}| \leq \frac{|H|}{m} \tag{2}$$

Da $i \in \left\{1, \ldots, N(N-1)\right\}$, ist $|H| = N(N-1)$. Weiters ist $m = 4$. Da nur jede zweite - gerade - Zahl aus dem Wertebereich von $m$ genutzt wird, ist $|\left\{h \in H : h(x) = h(y)\right\}| = \frac{|H|}{2} = \frac{N(N-1)}{2}$. Eingesetzt in (2) ergibt sich dadurch

$$\frac{N(N-1)}{2} \leq \frac{N(N-1)}{4} \Longleftrightarrow \frac{1}{2} \leq \frac{1}{4} \tag{3}$$

(3) ist nicht erfüllt $\Rightarrow$ $H$ ist nicht universell. □

Damit $H$ universell ist, halten wir uns nun an den Satz von Folie 20:

$$H = \left\{h_{a,b}(x) \,|\, 1 \leq a < N \wedge 0 \leq b < N\right\} \tag{4}$$

ist eine universelle Klasse von Hash-Funktionen.

Das heißt, wir müssen $a_i$, $b_i$ und $h_i$ so anpassen, dass Folgendes gilt:

$$a_i \in \{1, \ldots, N-1\}, b_i \in \{0, \ldots, N-1\}, h_i(x) = \left((a_i k + b_i) \mod N\right) \mod m \tag{5}$$

Es wurde in der Vorlesung bewiesen, dass diese Klasse von Hash-Funktionen universell ist.

$$a_i = f\left(i, N\right), f\left(i, N\right) \in \{1, \ldots, N-1\} \text{ für } i \in \Big\{1, \ldots, N\left(N-1\right)\Big\} \tag{6}$$

$$b_i = g\left(i, N\right), g\left(i, N\right) \in \{0, \ldots, N-1\} \text{ für } i \in \Big\{1, \ldots, N\left(N-1\right)\Big\} \tag{7}$$

Die folgenden Funktionen $f\left(i, N\right)$ und $g\left(i, N\right)$ erfüllen diese Bedingungen:

$$a_i = 1 + \Big(\left(i-1\right) \mod N - 1\Big) \tag{8}$$

$$b_i = \left\lceil \frac{i-1}{N} \right\rceil \tag{9}$$

Daraus ergibt sich

$$H = \Big\{h_i\left(x\right) = \Big(\left(a_i k + b_i\right) \mod N\Big) \mod m\Big\} \tag{10}$$

was eine universelle Klasse von Hash-Funktionen ist.