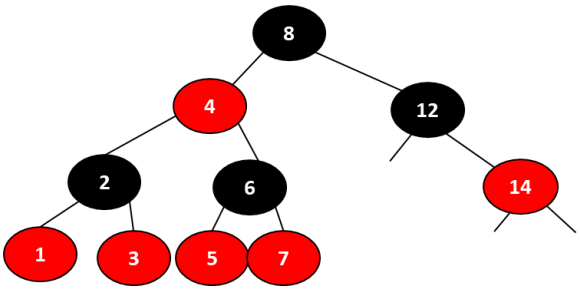
Lösung 09: B-Bäume

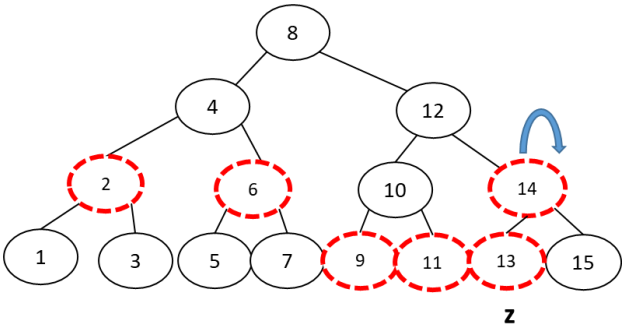
Aufgabe 1: Rot-Schwarz Baum

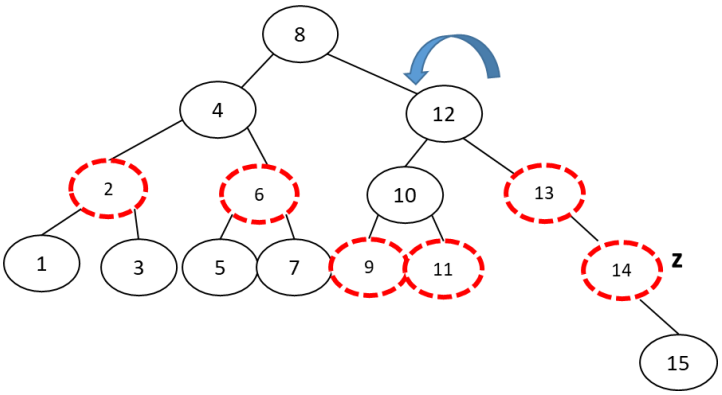
Achtung: Der in der Angabe vorgegebene Baum ist kein gültiger Rot-Schwarz-Baum. Um das zu sehen, zeichnet man am besten zumindest als linkes Kind der 14 ein schwarzes Blatt T.NULL ein und ebenso T.NULL für beide Kinder der 15. Dann sieht man, dass in den Pfaden zum linken Kind der 14 weniger schwarze Knoten liegen als zu den Kindern der 15. (Schwarztiefe verletzt).

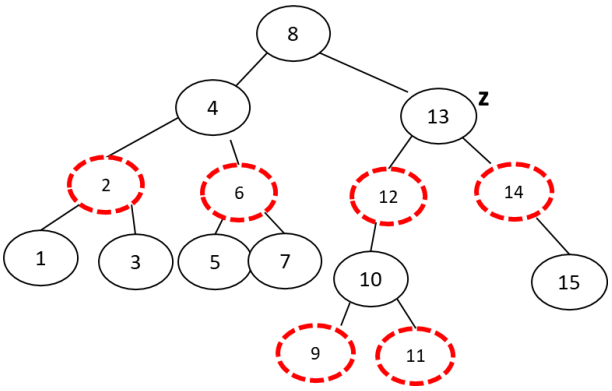
Möchte man einen gültigen Rot-Schwarz-Baum verwenden, könnte man z.B. mit dem folgenden Baum arbeiten und die 13 einfügen:



Ziel der Aufgabe ist zu zeigen, was passiert, wenn der Elterknoten von z kein linkes sondern ein rechtes Kind ist (else-Fall des Pseudocodes auf Seite 30 der Vorlesung). Egal, welchen Baum man als Ausgangsbasis nimmt (gültig oder ungültiger Rot-Schwarz-Baum), der Lerneffekt ist absolut der gleiche. Hier die Musterlösung mit dem (ungültigen) Rot-Schwarz-Baum der Angabe:







Hinweis: Hier kommt der Fall vor, dass der Elternknoten des eingefügten Knotens ein rechtes Kind und kein linkes Kind ist: `z.p != z.p.p.left`

Aufgabe 2: Minimaler Grad eines T eines B-Baumes

Hinweis: Die Mindest- und Höchstgrenzen für die Anzahl Schlüssel gelten nur für alle inneren Knoten mit Ausnahme der Wurzel.

a) Der minimale Grad $T = 1$ würde bedeuten, dass jeder Knoten (außer der Wurzel) mindestens 0 Schlüssel speichern muss. Leere Knoten wären also erlaubt.
Gleichzeitig würde das bedeuten, dass jeder innere Knoten (außer der Wurzel, kein Blatt) mindestens 1 Kind haben muss. Dass das wenig sinnvoll ist, erscheint offensichtlich.

b) $T = 2$ bedeutet, dass jeder Knoten (außer der Wurzel) x Schlüssel speichert mit $1 \leq x \leq 3$ also entweder 1, 2 oder 3 Schlüssel. Man überlegt sich, dass es nur die folgenden Möglichkeiten gibt, ASCII Art im Folgenden ☺
Um das herauszufinden, könnte man sich verschiedene Einfügereihenfolgen ansehen. Man stellt fest, dass der mittlere Wert der ersten 3 eingefügten Zahlen (Median) die Wurzel wird.

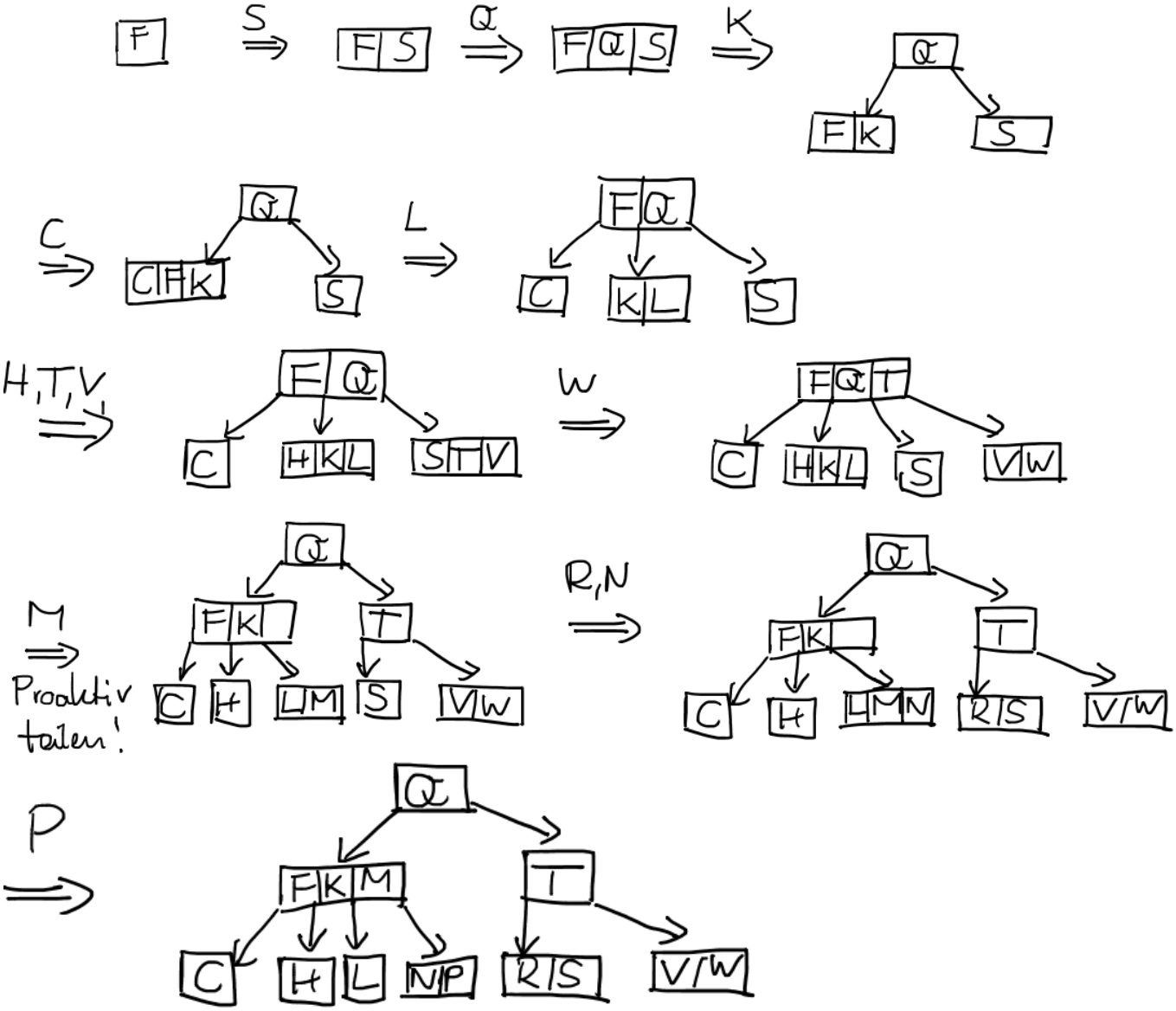
<u>Möglichkeit 1:</u>
2
1 3 4 5

<u>Möglichkeit 2:</u>
4
1 2 3 5

<u>Möglichkeit 3:</u>
3
1 2 4 5

Aufgabe 3: Einfügen mit Preemptive Split

Im Folgenden sind einige Zwischenschritte skizziert. .



Aufgabe 4: Maximum und Vorgänger

a) Das Suche des Maximums ist ähnlich wie im binären Suchbaum. Man muss einfach den rechtesten Schlüssel im rechtesten Blatt finden.
Lösung, siehe `BTree.java`

b) Am Beispiel lernt man, dass man 3 Fälle unterscheiden muss:
- Beispiel 18: Hier sucht man einfach das Maximum im rechten Teilbaum.
- Beispiel 30: Hier gibt man einfach den Schlüssel, der links von 30 gespeichert ist zurück.
- Beispiel 20: Man muss nach oben wandern und bestimmt vom wievielten Kind `j` man zum nächst höheren Knoten gelangt ist. Im Schlüssel mit Index `j-1`, also links des durchlaufenen

Verweises steht der Vorgänger. Gibt es links des Verweises keinen Schlüssel, muss man weiter nach oben gehen.

Code, siehe `BTree.java` im Verzeichnis:

https://inf-git.fh-rosenheim.de/muwo522/ad_wise_2019/tree/master/src/de/th_rosenheim/ad/uebung09