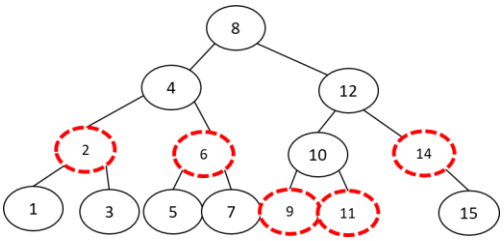
Übung 09: B-Bäume

Aufgabe 1: Rot-Schwarz Baum

Gegeben sei der folgende Rot-Schwarz-Baum. Zeichnen Sie diesen nach dem Einfügen der 13.

Anzugeben ist der Zwischenzustand nach jedem Umfärben und jeder Rotation. Geben Sie ferner an, auf welchen Knoten die Variable *z* bei jedem Zwischenzustand zeigt.



Aufgabe 2: Minimaler Grad *T* eines B-Baumes

Nehmen Sie an, dass ein gültiger B-Baum wie auf Folie 11 der Vorlesung definiert ist. *Hinweis*: Die dort angegebenen Mindest- und Höchstgrenzen für die Anzahl Schlüssel bzw. die Anzahl Kinder gelten für alle Knoten außer für die Wurzel.

a) Warum ist ein B-Baum mit minimalem Grad $T = 1$ nicht sinnvoll? Begründen Sie Ihre Antwort.

b) Zeichnen Sie alle gültigen B-Bäume mit minimalem Grad $T = 2$, die die folgenden Schlüssel enthalten: $S = \{1, 2, 3, 4, 5\}$. Annahme: Zwischendrin wird nichts gelöscht.

Aufgabe 3: Einfügen mit Preemptive Split

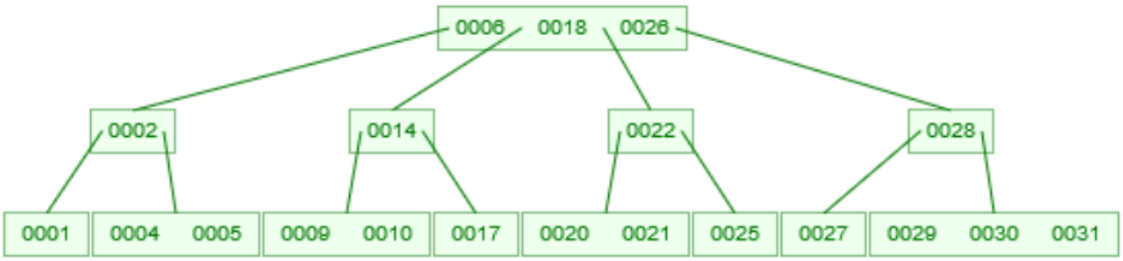
Zeichnen Sie den B-Baum für $\boldsymbol{T = 2}$, der sich ergibt, wenn die folgenden Schlüssel der Reihe nach in den anfangs leeren B-Baum eingefügt werden:

F, S, Q, K, C, L, H, T, V, W, M, R, N, P

Es genügt, wenn Sie ***jeweils*** den B-Baum zeichnen ***unmittelbar nachdem*** ein Knoten aufgeteilt werden muss („*Split*"). Zeichnen Sie ferner die finale Anordnung. Verwenden Sie exakt das Verfahren der Vorlesung mit ***Preemptive Split.***

Aufgabe 4: Maximum und Vorgänger

Im Folgenden ergänzen Sie die vorgegebene Klasse `BTree` um 2 Java-Methoden. Sie können beide Methoden mit der vorgegebenen JUnit Testklasse `BTreeTest` überprüfen. Der Test baut den abgebildeten B-Baum auf (minimaler Grad $\boldsymbol{t = 2}$).[1]



(umblättern)

[1] ***Achtung:*** Um diesen B-Baum mit der folgenden Animation nachzubauen, müssen Sie „preemptive" aktivieren und als „Max. degree" 4 einstellen.
https://www.cs.usfca.edu/~galles/visualization/BTree.html

a) `public Key maximum(BNode x)`
Die Methode findet den Knoten mit dem maximalen Schlüssel im dem Teilbaum, der Knoten `x` als Wurzel hat. Rückgabe ist dieser maximale Schlüssel.
Beispiel: Aufruf auf Wurzel sollte im gezeichneten Beispiel 31 zurückgegeben werden.

b) `public Key predecessor(Key key)`
Die Methode liefert den Vorgänger des Schlüssels `key` zurück. Rückgabe ist der Vorgängerschlüssel und `null` falls es keinen Vorgänger gibt (z.B. Schlüssel 1).

***Hinweise*:**

- Fälle unterscheiden: z.B. Vorgänger von 18, 30 bzw. 20 im konkreten Beispiel?
- De erste Teil der Methode ist bereits implementiert und zwar wird bereits der Knoten `current` gesucht an dem der Schlüssel `key` enthalten sein müsste, falls `key` überhaupt im B-Baum enthalten ist.
- Dabei wird auch in einem Stack mitgespeichert, in welcher Reihenfolge die Knoten des B-Baumes bei der Suche durchlaufen wurden. Das ist bei der Vorgängerbestimmung hilfreich, um ggfs. nach oben zu laufen (Fall: „Vorgänger der 20).
- Schauen Sie sich an, wie ein `BNode` definiert ist. Den restlichen vorgegebenen Code müssen Sie nicht nachvollziehen.

[Optional für Interessierte: So funktioniert ein B+-Baum]
https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html