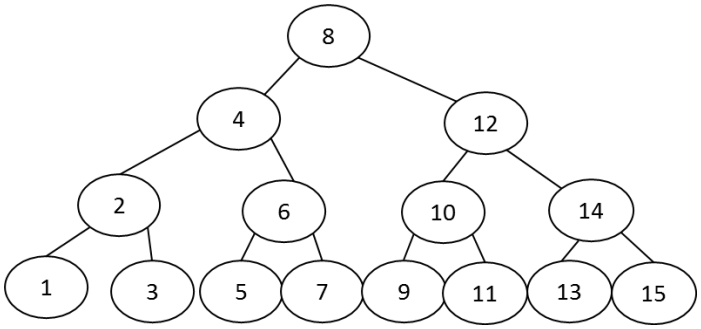
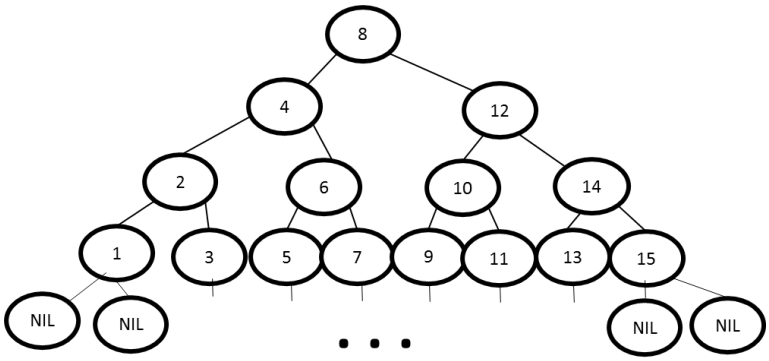
Lösung 08: Rot-Schwarz-Bäume

Aufgabe 1: Rot-Schwarz-Bäume

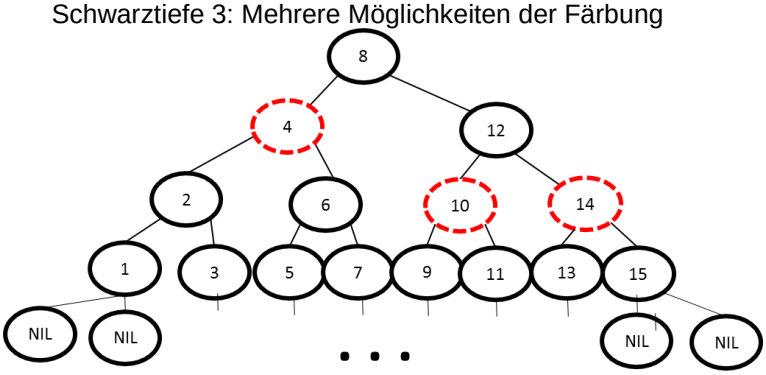
a)



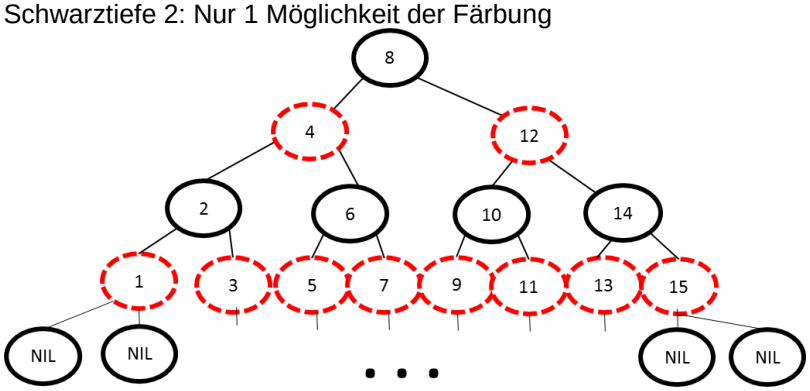
b) Hinweis: Bestimmt man die Schwarztiefe von der Wurzel, so wird die Wurzel nicht mitgezählt, dagegen die NIL-Blätter schon.
Schwarztiefe 4: Es gibt nur 1 Möglichkeit der Färbung



Schwarztiefe 3: Mehrere Möglichkeiten der Färbung



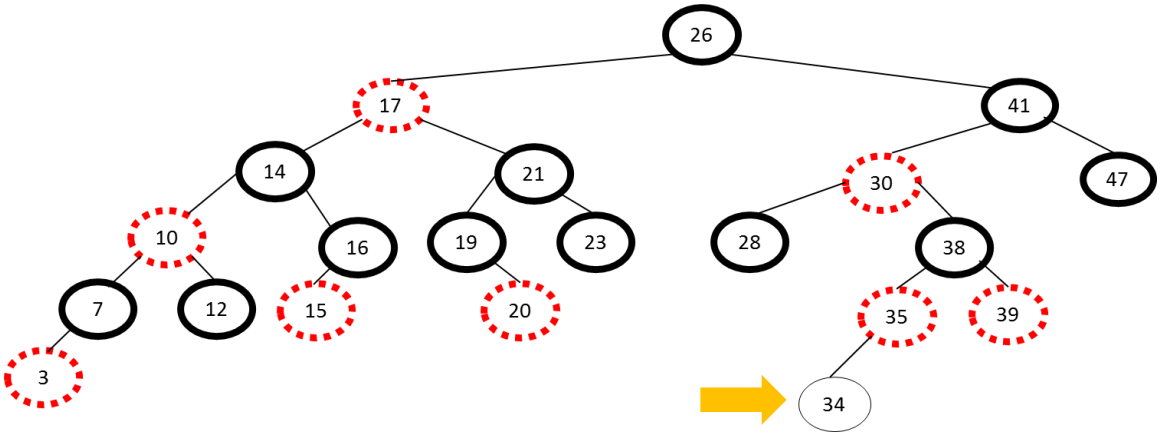
Schwarztiefe 2: Nur 1 Möglichkeit der Färbung



c) Die Abbildung zeigt die Einfügeposition des Knotens 34.

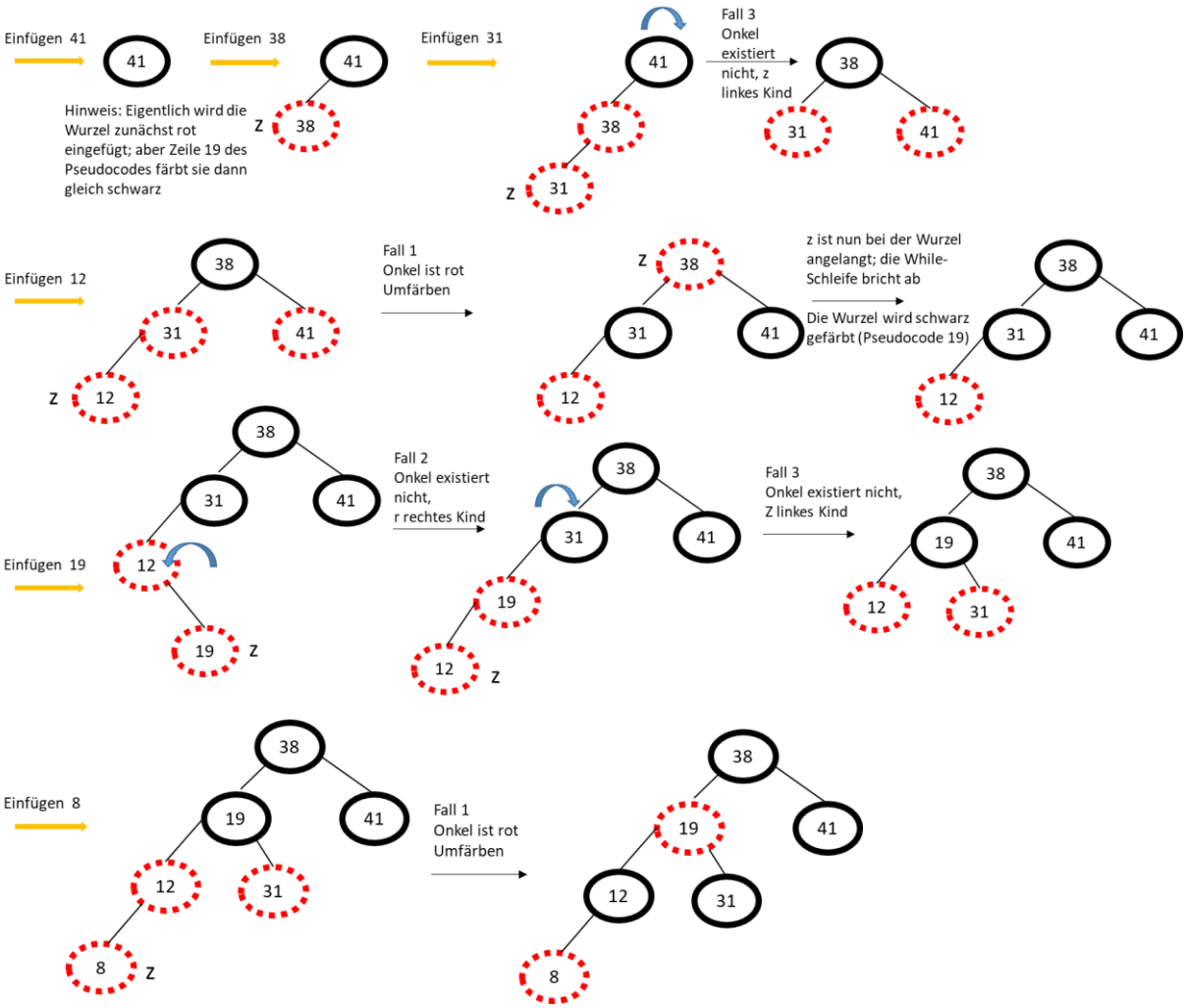
- Färbt man Knoten 34 rot: Dann ist die Bedingung 4 verletzt. Da 35 bereits rot ist, darf 34 nicht auch rot sein.
- Färbt man Knoten 34 schwarz: Dann ist die Bedingung 5 verletzt. Im linken Teilbaum der Wurzel gäbe es auf jedem Pfad jeweils 3 schwarze Knoten (inklusive der nicht gezeichneten schwarzen Blätter), im rechten Teilbaum 4 schwarz Knoten. Die Schwarztiefe wäre damit nicht wohldefiniert.

Hinweis: Da das erste Problem (Bedingung 4) leichter zu beheben ist, färbt der Algorithmus jeden Knoten zunächst rot.



Aufgabe 2: Einfügen in Rot-Schwarz-Bäume

Die folgenden Zeichnungen zeigen welcher Rot-Schwarz-Baum sich nach den jeweiligen Einfüge-Operationen ergibt. Man beachte, dass der finale Baum kein perfekt ausbalancierter binärer Suchbaum ist. Dennoch erfüllt er die Eigenschaften eines Rot-Schwarz-Baumes und ist ausreichend gut balanciert.



Den Ablauf kann man auch sehr gut mit der folgenden Animation nachvollziehen:

https://www.cs.usfca.edu/~galles/visualization/RedBlack.html

Aufgabe 3: Iteratives Preorder in binären Suchbäumen

Man verwendet einen Stack. Aufpassen muss man nur, dass man zuerst den rechten, dann den linken Knoten auf den Stack legt.

Lösung, siehe Quelltest im GitLab:

https://inf-git.fh-rosenheim.de/muwo522/ad_wise_2019/tree/master/src/de/th_rosenheim/ad/uebung08