AVL Bäume Trainingsaufgaben

Einleitung

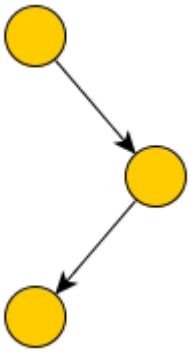
Die folgenden Trainingsaufgaben sollen mehr Sicherheit im Umgang mit AVL-Bäumen bringen. Sie sind entsprechend des Schwierigkeitsgrads sortiert. Die Lösungen befinden sich im Anhang.

1. AVL-Baum Validierung

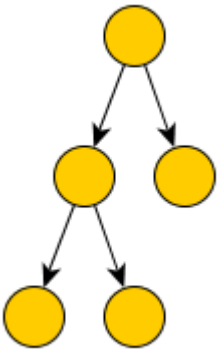
Bestimmen Sie die Balance für alle Knoten der abgebildeten Bäume und beurteilen Sie, ob es sich um einen AVL-Baum handelt oder nicht. Falls der Baum zu einem AVL-Baum ausbalanciert werden könnte geben Sie an, um welchem Knoten welche Rotation ausgeführt werden muss.

a.)

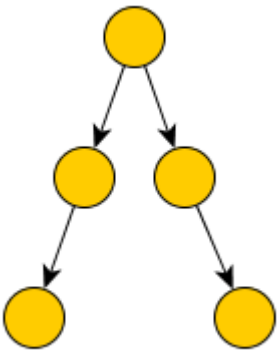
b.)



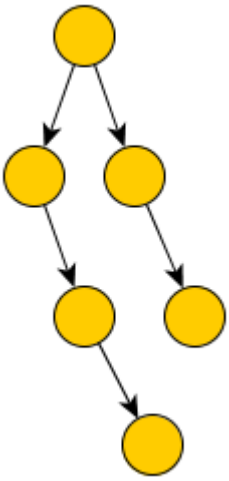
c.)



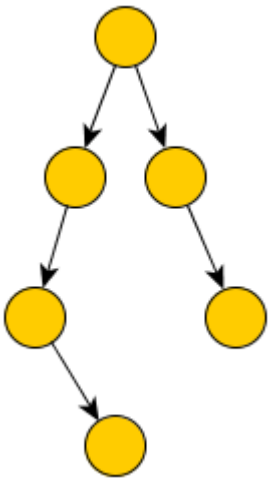
d.)



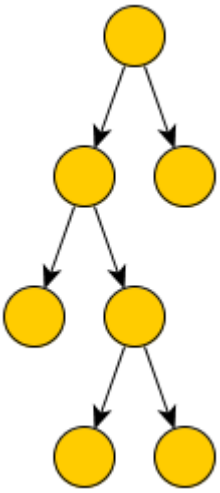
e.)



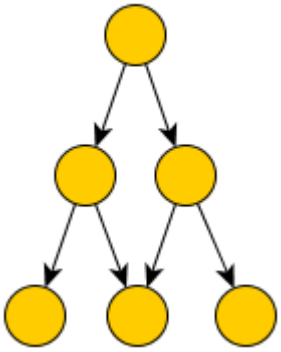
f.)



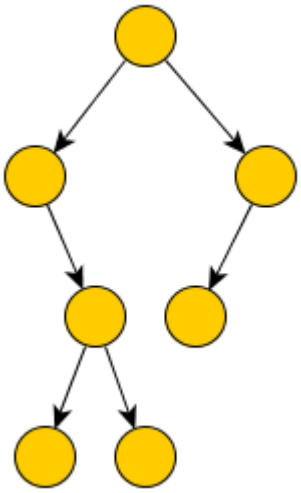
g.)



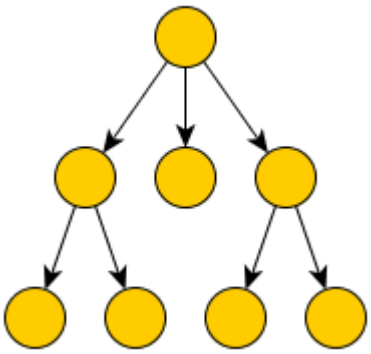
h.)



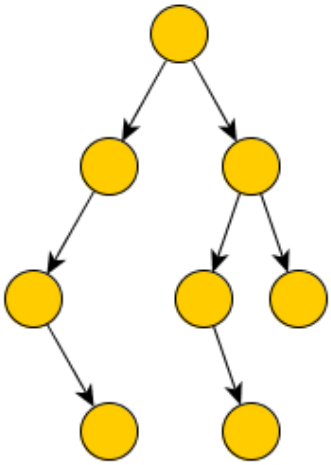
i.)



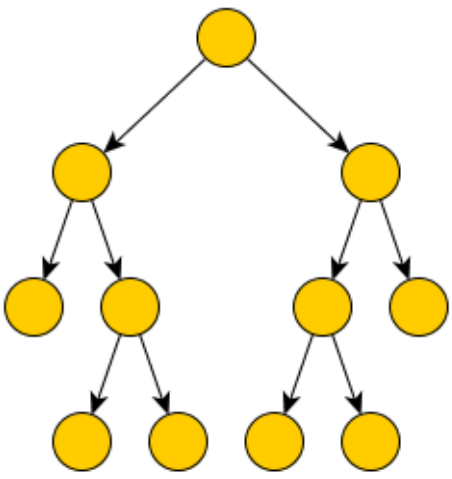
j.)



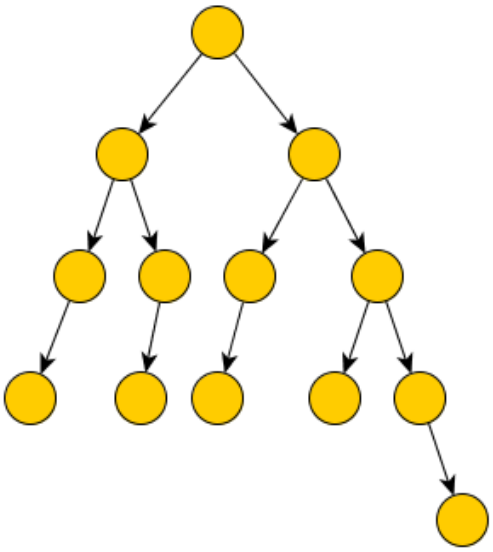
k.)



l.)



m.)



2. Hinzufügen von Knoten in einen AVL-Baum

Notieren Sie sich die durchgeführten Rotationen und um welchen Knoten die Rotation getätigt wird. Zeichnen Sie den Baum nach jeder Rotation neu.

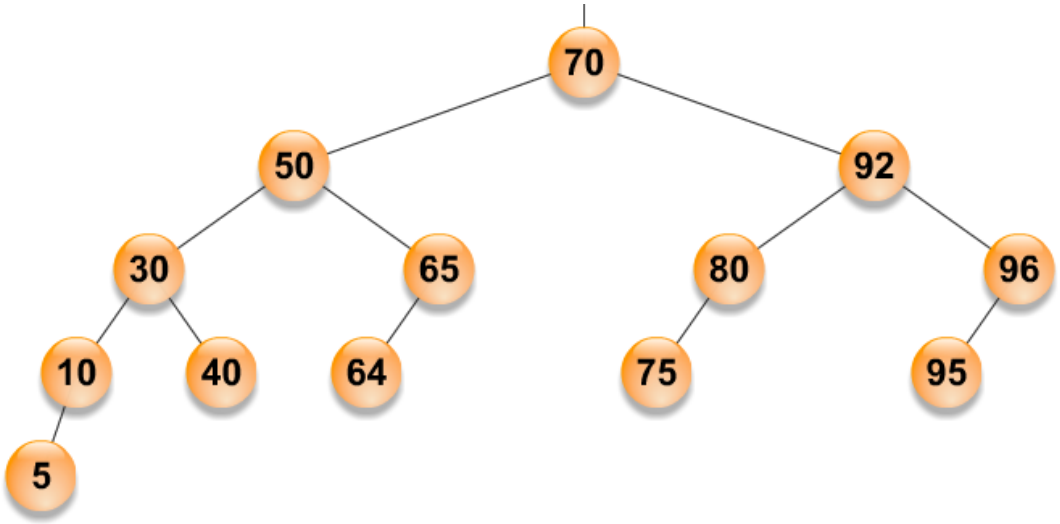
a.) Fügen Sie nacheinander die folgenden Zahlen in einen leeren AVL-Baum ein:
50, 60, 70, 40, 30, 20, 45, 55, 65, 75, 85

b.) Fügen Sie nacheinander die folgenden Zahlen in einen leeren AVL-Baum ein:
50, 30, 80, 10, 40, 70, 90, 65, 64, 75, 95, 92, 96

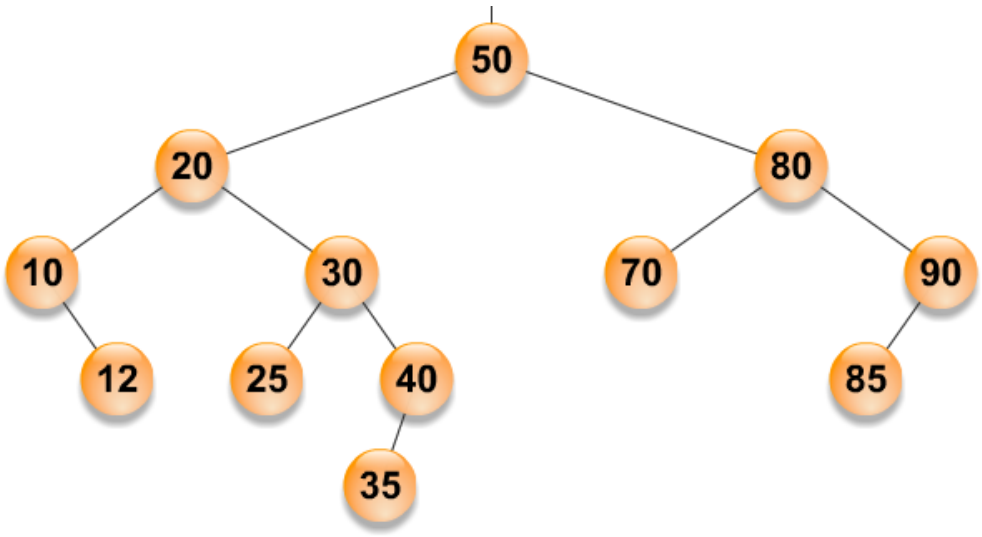
3. Löschen von Knoten aus einem AVL-Baum

Zeichnen Sie den Baum nach jeder Rotation neu, auch wenn dies zu Beginn ein wenig Aufwand bedeutet. So bleibt es übersichtlich und nachvollziehbar, wie Sie vorgegangen sind.

a.) Gegeben sei der folgende AVL-Baum. Löschen Sie die folgenden Knoten in gegebener Reihenfolge: 50, 80, 70, 92, 95, 96, 30



b.) Löschen Sie im folgenden Baum den Wurzel-Knoten:



4. Programmieraufgaben

Nehmen Sie für die Programmieraufgaben an, dass folgende Klassen vorhanden sind.

```
class Tree<K extends Comparable<? super K>> {
      private Node<K> root = null;

}


class Node<K extends Comparable<? super K>> {
             K key;

             Node<K> left, right;

}
```

Ergänzen Sie die Klasse `Tree` um die beiden folgenden Methoden:

a.) Die Bäume seien gemäss der in Suchbäumen üblichen Ordnungsregel aufgebaut. Programmieren Sie eine Methode namens `printDescendingOrder()`, welche die Elemente eines AVL-Baums in absteigender Grösse sortiert ausgibt.

b.) Programmieren Sie eine Methode namens `validateOrder()` die überprüft, ob sich im linken Teilbaum eines Knotens nur kleinere und im rechten Teilbaum eines Knotens nur grössere Keys befinden. Es soll eine `IllegalStateException` geworfen werden, falls diese Ordnungsrelation verletzt ist.