**treecheck – Protokoll**

**Einfügen von Werten:**

Funktioniert simpel und mit einem Aufwand von O(log(n)) beim Average Case (Best Case: O(1), Worst Case: O(n))

Man beginnt beim Root-Element und sieht nach, ob der einzufügende Wert kleiner oder größer ist und ruft dann jeweils das addNode(…) für den jeweiligen Knoten auf, sofern dieser existiert. Existiert der jeweilige Folgeknoten irgendwann nicht, so wird der Wert dort eingefügt.

Falls der Wert weder größer noch kleiner sein sollte, befindet sich der Wert logischerweise an dieser Stelle und die Funktion wird abgebrochen.

**Höhe bestimmen:**

Hat den Aufwand von O(log(n)) – andere Cases wie oben – und man führt das getHeight(…) für beide Folgeknoten aus. Von den beiden erhaltenen Werten nimmt man das Maximum, addiert 1 (Höhe des eigenen Knotens) dazu und gibt das Ergebnis zurück.

**AVL testen:**

Hat den Aufwand von O(n\*log(n)), da wir für jeden Knoten (=n) ein getHeight(…) (=log(n)) durchführen müssen. Wir rufen auch diese Funktion für beide Folgeknoten auf, wobei wir deren booleans die zurückgegeben werden mit einem AND verknüpfen, wodurch ein einzelnes false ausreicht um den ganzen Baum als Nicht-AVL-Baum zu erkennen.

**Minimum/Maximum:**

Hier liegt der Aufwand wieder bei O(log(n)). Man geht solange in eine Richtung bis man beim Blattknoten ankommt und gibt dessen Wert zurück.

**Durchschnitt:**

Dieser setzt sich sowohl aus der Zählung als auch der Summe aller Knoten zusammen, die beide genau gleich funktionieren:

Man sieht nach welche Folgeknoten existieren und ruft bei denen die Funktion auf. Zurückgegeben wird die Summe beider Werte plus dem eigenen Knoten-Wert (bei count() natürlich 1).

Am Ende rechnet man Summe/Zählung und erhält den (arithmetischen) Durchschnitt.