Protokoll TREECHECK

Matthias Kemmer, Li Wen Wang

Das Programm wurde in Python implementiert und erlaubt das Einlesen von Textdateien, mit dem Ziel AVL Bäume aufzubauen. Die Funktionalitäten sind:

- Prüfung, ob Resultat ein balancierter AVL Baum ist
- Ausgabe von Balance-Faktor für jeden Knoten
- Ausgabe von Baum-Statistiken
- Suche nach einem einzelnen Wert
- Suche nach einem Teilbaum

Aufwand Prüfung AVL
Im durchschnittlicher Fall oder wenn der Baum ausbalanciert ist, dauert die Prüfung $O(\log_2 n)$. Wenn ein Baum vollständig unbalanciert ist, dann kann der Aufwand der Prüfung im worst case $O(n)$ sein.

Die Prüfung erfolgt durch Berechnung des Balance Faktors für jeden Knoten. Bei einem balancierten AVL Baum hat kein Knoten einen höheren Balance-Faktor als ±1. Da die Höhe bereits in den Knoten gespeichert wird, ist die Traversierung durch den Baum der aufwendigste Schritt.

Balance(k) = Höhe(rechter Teilbaum) - Höhe(linker Teilbaum)

Aufwand Knoten einfügen
Im durchschnittlicher Fall oder wenn der Baum ausbalanciert ist, dauert das Einfügen $O(\log_2 n)$. Wenn ein Baum vollständig unbalanciert ist, dann kann der Aufwand im worst case $O(n)$ sein. Für das Einfügen eines Knotens ist die Traversierung durch den Baum der aufwändigste Schritt.

Aufwand suche Knoten
Im durchschnittlicher Fall oder wenn der Baum ausbalanciert ist, dauert das Suchen eines Knotens $O(\log_2 n)$. Wenn ein Baum vollständig unbalanciert ist, dann kann der Aufwand im worst case $O(n)$ sein. Für die Suche nach einem Knoten ist die Traversierung durch den Baum der aufwendigste Schritt.

Aufwand suche Teilbaum
Im durchschnittlicher Fall oder wenn der Baum und Teilbaum ausbalanciert sind, dauert das Suchen eines Teilbaums $O(m \log_2 n)$ bzw. $O(n \log_2 n)$. Wenn der Baum und Teilbaum vollständig unbalanciert sind, dann kann der Aufwand im worst case $O(n^2)$ bzw. $O(m\ n)$ sein.

Für die Suche nach Teilbaum ist die Traversierung durch den Baum der aufwendigste Schritt und dieser Schritt muss für jeden Knoten des Teilbaums wiederholt werden.

Rekursion
Die Abbruchbedingungen für Rekursion waren (1) derzeitiger Knoten = NULL-Wert und (2) derzeitiger Knoten ist gesuchter Schlüssel. Ansonsten wurde der Schlüssel und Knoten vergleichen und durch erneuten Aufruf im passenden Teilbaum fortgeführt.

Traversierung durch den Baum bei Ausgabe der Knoten erfolgte in "reverse post-oder", also (1) rechter Teilbaum, (2) linker Teilbaum und (3) derzeitiger Knoten.