Technische Universität Wien
Institut für Computergraphik und Algorithmen
Arbeitsbereich für Algorithmen und Komplexität

186.866 Algorithmen und Datenstrukturen VU

Programmieraufgabe E4

---

1 Übersicht

Ihre Aufgabe ist es die **Insert-Operation für AVL-Bäume** zu implementieren. Dazu müssen Sie eine bereits bestehende Klasse "AVLTree" um die *insert*-Methode erweitern. Sie können davon ausgehen, dass **keine** Zahlen doppelt in den AVL-Baum eingefügt werden.

Das benötigte Framework ("AlgoDat_E4.zip") steht im TUWEL zu Verfügung. Implementieren Sie in der Datei "src/exercise/E4.java" folgende Methode:

- `insert(AVLTreeNode newNode)`

2 Implementierung

Die Datei "E4.java" enthält die Klasse "MyAVLTree". Implementieren Sie in dieser Klasse die Methode `insert(AVLTreeNode newNode)`. Dabei soll der neue Knoten (**newNode**) in einen balancierten AVL-Baum eingefügt werden. Der Baum ist zu Beginn leer. Später sind bereits Knoten enthalten und Sie müssen sicherstellen, dass der Baum nach dem Einfügen des neuen Knotens noch balanciert ist.

Denken Sie daran, dass Sie sich in der Klasse befinden, die den Baum darstellt, in den der Knoten eingefügt werden soll. Sie haben Zugriff auf die folgende Methoden des Baums.

- `root()` returniert den Wurzelknoten des AVL-Baums. Dieser dient als Einstieg in den Baum.
- `setRoot(AVLTreeNode newRoot)` setzt den Knoten **newRoot** als neuen Knoten das gesamten AVL-Baums.

- `balance(AVLTreeNode node)` returniert die Balance des Unterbaums mit dem Knoten **node** als Wurzel. Die Balance *bal* eines Baumes wird, genau wie in den Folien, durch $bal = h_2 - h_1$ berechnet. Dabei ist h_1 die Höhe des linken Unterbaums und h_2 die Höhe des rechten Unterbaums.

- `isEmpty()` returniert **false** wenn der AVL-Baum mindestens einen Knoten enthält, sonst **true**.

- `height(AVLTreeNode node)` returniert die Höhe von **node** im Baum. Ein Blatt hat eine Höhe von 0. Die Höhe h_n eines Knotens n im Baum ist definiert als $h_n = \max\{h_l, h_r\} + 1$, wobei h_l, h_r die Höhe des linken, beziehungsweise rechten, Kindes von n ist. Wenn **node** gleich *NULL* ist returniert die Methode -1.

Das Argument **newNode** ist ein Objekt der AVLTreeNode-Klasse und bietet ihnen folgende, bereits implementierte, Methoden.

- `key()` returniert den, mit diesem Knoten assoziierten, Key (als `int`).

- `left()/right()/parent()` returniert den linken/rechten Nachfolger-Knoten bzw. den Elternknoten. Wenn der Knoten keinen linken/rechten Nachfolger bzw. Elternknoten hat, returniert die Methode `NULL`.

- `setLeft(AVLTreeNode newLeft)/`
  `setRight(AVLTreeNode newRight)/`
  `setParent(AVLTreeNode newParent)`
  speichert newLeft/newRight/newParent als neuen linken/rechten Nachfolger bzw. Elternknoten dieses Knotens.

- `setHeight(int newHeight)` speichert newHeight als neue Höhe dieses Knotens.

Sie haben außerdem Zugriff auf die Rotate-Operation der AVL-Bäume, die bereits in der Klasse "AVLTree" implementiert sind und von der Unterklasse "MyAVLTree" verwendet werden können. Die folgenden Methoden sind, **wie im Pseudocode der Folien**, so implementiert, dass sie einen Knoten als Argument akzeptieren, den Unterbaum mit diesem Knoten entsprechend rotieren und den neuen Wurzelknoten des rotierten Unterbaums zurückgeben.

- `rotateToRight(AVLTree.AVLTreeNode node)` rotiert den Unterbaum von **node** nach rechts.

- `rotateToLeft(AVLTree.AVLTreeNode node)` rotiert den Unterbaum von **node** nach links.
- `doubleRotateLeftRight(AVLTree.AVLTreeNode node)` führt eine links-rechts Rotation an **node** aus.
- `doubleRotateRightLeft(AVLTree.AVLTreeNode node)` führt eine rechts-links Rotation an **node** aus.

Hinweis: Sie müssen bei ihrer Implementierung dafür sorgen, dass nach Verschiebung von Knoten im Baum, an den involvierten Knoten die richtige Höhe gespeichert ist.

3 Testen

Wenn sie die Datei "AlgoDat_E4.java" ausführen und eine der drei Instanzen auswählen, wird ihre Implementierung auf die folgenden Fehler überprüft.

- **Korrekte Balance.** In der Klasse "MyAVLTree" finden Sie eine Konstante namens *THRESHOLD_BALANCE* mit dem Wert 2. Wenn ihre Balance gleich diesem Wert, oder größer ist, wird eine dementsprechende Nachricht ausgegeben. Sie können den Wert für Testzwecke verändern. Beachten Sie aber, dass eine korrekte Überprüfung für AVL-Bäume nur durchgeführt wird, wenn der **Wert gleich 2 ist**.
- **Anzahl der Knoten.** Sollten Sie fälschlicherweise beim Einfügen eines neuen Knotens, Knoten löschen und damit die Anzahl der Knoten im Baum nicht stimmen, wird eine dementsprechende Nachricht ausgegeben.

Beachten Sie, dass alle anderen Fehler (z.B. inkorrekte Ordnung der Knoten in einem balancierten Baum) **nicht überprüft werden**. Eine falsche Lösung die keinen der oben beschriebenen Fehler aufweißt **wird nicht als falsch erkannt und somit als Lösung gespeichert**. Stellen Sie sicher, dass der Baum in ihrer Abgabe, ein korrekter balancierter AVL-Baum ist. Sollte ein Fehler ausgegeben werden, wird die Lösung inklusive des letzten, falschen Baumes, gespeichert. Sie können diese im Browser visualisieren und die Ausgabe zur Fehlerfindung benutzen.

4 Auswertung

Wenn Sie die Datei "AlgoDat_E4.java" ausführen, können Sie, wie im Tutorial beschrieben, in der Konsole eine Testinstanz auswählen. Es wird eine *.csv-Datei* im *solutions* Ordner erstellt, welche Sie mit der Datei "E4/Evaluation/plot.html" auswerten können.

Hier wird ein AVL-Baum dargestellt, in den der Reihe nach die Elemente der gelösten Instanz eingefügt werden. Sie können über die Steuerelemente das Einfügen der Elemente Schritt für Schritt nachvollziehen. Für die Abgabe relevant ist nur die Instanz abgabe.csv. Der Baum wird als .svg-Datei erzeugt und kann daher nicht wie gewohnt gespeichert werden. Erstellen sie einen **Screenshot des vollständigen balancierten AVL-Baums (Step 19)** der Instanz **abgabe.csv**.

Anmerkung: Alle Blätter des AVL-Baumes repräsentieren die Nullpointer (mit Label **N**), die anzeigen dass die Knoten, auf dieser Seite, keine Nachfolgerknotenhaben.

Beantworten Sie folgende Fragen:

- Beschreiben Sie Ihre Implementierung der *insert* Methode. Welche verschiedenen Fälle mussten Sie beachten?
- Gibt es in ihrem vollständigen AVL-Baum (Step 19) für die Instanz "abgabe.csv" Blattknoten mit einem Höhenunterschied von mehr als 1?
- Wenn "Ja" ist ihre Lösung immer noch ein balancierter AVL-Baum? Begründen Sie.

Hinweis: Bevor Sie beginnen, lesen Sie sich unbedingt das Tutorial zu den Programmieraufgaben durch („Tutorial.pdf"). Dieses finden Sie im ZIP-Archiv des Programmier-Frameworks im TUWEL. Diese bietet eine kurze Einführung in das verwendete Framework.