**Aufgabenblatt 5** **Abgabe: 27.05.06, 24.00 Uhr**

1. **Optimale Suchbäume** (mittel) (schriftlich, 6 Punkte)
Optimale Suchbäume: Ein ausgeglichener Suchbaum hat eine maximale Tiefe von $1 \cdot \log(n) + O(1)$. Somit ist für solch einen Suchbaum auch die mittlere Suchdauer durch $1 \cdot \log(n) + O(1)$ beschränkt und wir schließen, dass ein Optimaler Suchbaum stets eine mittlere Suchdauer von höchstens $1 \cdot \log(n) + O(1)$ hat.

   a) (leicht, 1 Punkt) Geben Sie ein Beispiel für einen Optimalen Suchbaum mit $n$ Knoten, dessen mittlere Suchdauer $1 \cdot \log(n) + O(1)$ beträgt.

   b) (mittel, 2 Punkte) Auch Optimale Suchbäume können zu Listen entarten. Entwerfen Sie zunächst ein Beispiel mit 4 Knoten, bei dem der Optimale Suchbaum zu einer Liste entartet. Wie groß ist die mittlere Suchdauer?

   (schwer, 3 Punkte) Verallgemeinern Sie das Beispiel auf $n$ Knoten und berechnen Sie auch hier die mittlere Suchdauer.

2. **Einfügen in AVL-Bäume** (leicht) (schriftlich, 3 Punkte)
AVL-Bäume sind spezielle Suchbäume.

   a) Wir haben bereits Suchbäume betrachtet (um effizient Daten zu verwalten) und haben Optimale Suchbäume betrachtet, um im Mittel die Suchzeit zu minimieren. Begründen Sie, wozu nun noch AVL-Bäume als weitere Variante von Suchbäumen betrachtet werden, finden Sie einige Vor- und Nachteile.

   b) Fügen Sie die Elemente „Claus“, „Lewandowski“, „Schmid“, „Burkovski“, „Griepentrog“, „Draskoczy“, „Grigas“, „Gruber“ und „Fischer“ in dieser Reihenfolge in einen anfangs leeren AVL-Baum ein. Welche Rotationen müssen Sie durchführen? Geben Sie zusätzlich nach jeder Einfügeoperation die Preorder-Ausgabe des Baumes im eClaus-System ab.

3. **Löschen in AVL-Bäumen I** (mittel) (schriftlich, 4 Punkte)
Programmieren (bzw. ergänzen Sie den Programmcode aus dem Skript) Sie in Pseudocode die *Delete*-Operation für AVL-Bäume. Achten Sie darauf, dass Sie wirklich alle Fälle berücksichtigen.

4. **Löschen in AVL-Bäumen II** (mittel–schwer) (schriftlich, 5 Punkte)
Das Löschen in AVL-Bäumen kann bis zu $O(\log n)$ Rotationen benötigen.
Ein Kommillitone hat in der Mittagspause ein wenig mit AVL-Bäumen herumgespielt und erzählt Ihnen nun seine Entdeckung: Wenn man in einen beliebigen AVL-Baum einen Wert $x$ einträgt, so muss danach (wie im Skript bewiesen wurde) höchstens 1 (Doppel-)Rotation durchgeführt werden, um die AVL-Eigenschaft wiederherzustellen. Löscht man dann diesen Knoten sofort wieder aus dem Baum, so bleibt die AVL-Eigenschaft stets erhalten, so dass keine Rotation notwendig ist. Dies gilt auch, wenn der Wert $x$ nach dem Einfügen (und ggf. Rotieren) kein Blatt war.

a) (1 Punkt) In welcher Situation ist ein eingefügter Knoten nach Wiederherstellung der AVL-Eigenschaft kein Blatt?

b) (4 Punkte) Beweisen Sie, dass die Vermutung Ihres Kommillitonen richtig ist oder finden Sie ein Gegenbeispiel.

**5. Löschen im Worst-Case-Fall** (mittel) (schriftlich, 4 Punkte)

Finden Sie ein Worst-Case-Beispiel (maximale Anzahl von Rotationen) für das Löschen eines Knotens in einem AVL-Baum mit 7 Knoten.

Geben Sie im eClaus-System die Preorder-Ausgabe des Baumes, die Anzahl der Rotationen und deren Typ und eine kurze Erläuterung ab.

**6. Anzahl AVL-Bäume $C_t^{\mathbf{AVL}}$** (mittel) (schriftlich, 5 Punkte)

Die Fibonacci-Bäume sind die dünnsten AVL-Bäume. Dieser schlechteste Fall kommt allerdings nur selten vor, da es im Verhältnis zu der Anzahl der AVL-Bäume nur sehr wenige davon gibt!

Aber wieviele sind es genau und wie groß bzw. klein ist das Verhältnis zu der Anzahl der AVL-Bäume mit der Tiefe $t$?

a) Geben Sie eine rekursive Definition für die Anzahl der Fibonacci-Bäume $C_t^{\mathrm{AVL_{Fib}}}$ an und berechnen Sie diese für jede Tiefe $t \in \{1, 2, 3, 4, 5, 6\}$.

b) Geben Sie eine rekursive Definition für die Anzahl der AVL-Bäume $C_t^{\mathrm{AVL}}$ an und berechnen Sie diese ebenfalls für jede Tiefe $t \in \{1, 2, 3, 4, 5, 6\}$.

c) Berechnen Sie nun das Verhältnis $V_t^{\mathrm{AVL_{Fib}}} = \frac{C_t^{\mathrm{AVL_{Fib}}}}{C_t^{\mathrm{AVL}}}$ für jede Tiefe $t \in \{1, 2, 3, 4, 5, 6\}$.