TINF24ITA-Laborprojekt: Bäume

Algorithmen und Komplexität

Das Laborprojekt ist teil der Prüfungsleistung der Vorlesung „Algorithmen und Komplexität“ (Modul T2INF1003). Eine erfolgreiche Bearbeitung ist für die Teilnahme an der Klausur erforderlich. Das Projekt wird in Teams von jeweils zwei Personen bearbeitet.

Alle Quelldateien (keine Kompilate!) sind bis zum Abgabetermin am **Montag, den 07.07.2025** als Archiv mit den Nachnamen der Teilnehmer <Name1>_<Name2>.zip per E-Mail einzureichen an: fritzsch@lehre.dhbw-stuttgart.de

In dieser Projektarbeit sollen zunächst einige hilfreiche Algorithmen für Binärbäume implementiert werden. Des Weiteren soll die durchschnittliche Höhe zufälliger Suchbäume experimentell ermittelt werden.

Ihre Umsetzung kann auf der vorhandenen Java-Implementierung basieren oder wahlweise in C++ erfolgen. Die gegebenen Java-Quelldateien finden Sie im Archiv `TreeTools.zip`.

a) Erweitern Sie die gegebene Klasse `TreeTools` um folgende Methoden:

```
public static int treeHeight(Tree b)
```
liefert für den übergebenen Baum die Höhe zurück

```
public static int anzahlKnoten(Tree b)
```
liefert für den übergebenen Baum die Anzahl der Knoten zurück

```
public static void printTreeInorderWithParenthesis(Tree b)
```
gibt den übergebenen Baum in Inorder-Traversierung mit Klammerung aus

```
public static void printTreeLevelorder(Tree b)
```
gibt den übergebenen Baum in Levelorder-Traversierung aus

```
public static int[] searchTreeSort(int[] zahlen)
```
sortiert eine übergebene Zahlenfolge mittels einer Instanz der Klasse SearchTree und liefert diese zurück. Tragen Sie dazu die gegebene Folge in einen SuchBaum ein. Die Sortierung soll mit Hilfe einer Methode `private static void tree2SortedStack(Tree b, Stack k)` *realisiert werden, welche die Zahlen rekursiv per geeigneter Traversierung sortiert auf einen Stack legt.*

Die Klasse `TreeTools` enthält bereits einige Methoden zur Ausgabe eines Baums auf der Konsole, die Sie zur Visualisierung Ihrer Zwischenergebnisse nutzen können. Die Methode `treeHeight()` muss dafür allerdings bereits korrekt implementiert worden sein.

b) Ein zufällig erzeugter binärer Suchbaum mit n Knoten hat eine Höhe von $c \cdot \log_2 n$. Sie sollen diese Aussage experimentell bestätigen. Erzeugen Sie dazu große binäre Suchbäume mit n Elementen und bestimmen deren Höhe. Um die Suchbäume aufzubauen, erzeugen Sie vor dem Einfügen eine zufällige Anordnung der Elemente, z.B. mit Hilfe von `StdRandom.shuffle()` aus der Bibliothek `stdlib.jar` (http://introcs.cs.princeton.edu/java/stdlib/javadoc/StdRandom.html).

Erweitern Sie die Klasse `TreeToolsTest` um einen Test „Höhe zufälliger Suchbäume“.
Vom Benutzer soll die Anzahl der Knoten für einen Testbaum eingelesen werden, sowie die Anzahl der Durchläufe. Gibt der Benutzer „1000“ und „10“ ein, so sollen nacheinander 10 Suchbäume mit je 1000 Knoten erstellt werden. Für jeden Durchlauf ist die Höhe des Baums zu bestimmen. Zum

Schluss wird die durchschnittliche Höhe der Bäume aller Durchläufe berechnet und daraus die Konstante c ermittelt (runden Sie diese auf 2 Nachkommastellen).
Ihre Ausgabe sollte wie folgt aussehen und ist als Textdatei Ihrer Abgabe hinzuzufügen:

```
Erzeuge 10 Suchbäume mit je 1000 Knoten

Höhe Suchbaum 1: xxx
Höhe Suchbaum 2: xxx
...

Durchschnittliche Höhe: xxx (entspricht xx.xx * log2 n)
```