ALGODAT

Protokoll für die zweite Programmieraufgabe

Zwei Klassen definiert:

1. Die Klasse Tree die einen Pointer auf den ersten Knoten des Binärsuchbaums speichert, und Methoden aufrufen kann die gewisse Operationen am Baum durchführen.
2. Die Klasse Node beschreibt die Eigenschaften der jeweiligen Knoten im Baum. Sie hat drei Member Variablen namens key,left,right. Key speichert den Wert des Knotens, wobei left und right, Pointer auf weitere Knoten sind.

Methoden:

1. **treeRead**: Ist eine übergeladene Funktion, die entweder eine oder zwei Dateien einliest. Im ersten Fall werden der Balancefaktor, statistische Daten und Information darüber, ob es sich beim eingelesenen Tree um einen AVL Baum handelt, ausgegeben. Falls zwei Dateien eingelesen werden, wird der Suchbaum auf eine Subtreestruktur untersucht.
2. **fillTree**: Erzeugt Knotenobjekte und erstellt einen Baum. **O(n)** wobei n = Anzahl der Knoten.
3. **getMin**: Returned den kleinsten Knotenwert im Baum. O(h) wobei h = Höhe des Baums. Falls der Baum zu einer Liste entartet, muss man mit O(n) Aufwand rechnen ansonsten ist **O(logn)** zu erwarten.
4. **getMax**: Returned den größten Knotenwert im Baum. Aufwand: Analog zu getMin
5. **getAverage**: Returned den Mittelwert der Knoten. Aufwand: **O(n)** n = Knotenanzahl
6. **getAVL**: Gibt auf der Konsole die Balancefaktoren für alle Knoten aus und ob der Baum ein AVL Baum ist oder nicht.
7. **easySearch**: Der Baum wird nach einem Keywert durchsucht und der Pfad wird in einem Vektor notiert, im Falle einer erfolgreichen Suche wird der Pfad ausgegeben. Aufwand: **O(logn)**
8. **subTree**: Der Baum wird nach einer Subtreestruktur untersucht dazu wird eine rekursive Funktion verwendet, die überprüft ob die Knoten sowohl im Subtree als auch im Haupttree in derselben Reihenfolge enthalten sind. Eine weitere rekursive Funktion isSub die am Anfang jedes subtreeCheck Stackframes aufgerufen wird, stellt fest, ob ein Key überhaupt erreichbar ist (von einem spezifischen Startknoten im Haupttree). Aufwand: Die äußere Funktion O(m) wobei m die Anzahl der Knoten im Subtree ist. Die innere Funktion O(n) wobei n die Anzahl der Knoten eines Baumes der ein Unterbaum des Hauptbaumes ist. Daraus folgt O(mn) = **O(n^2)**