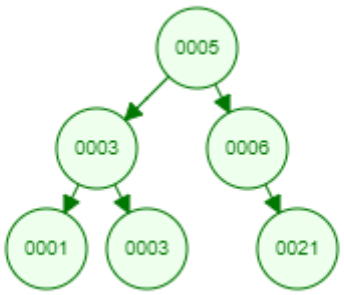
Team: 2, Florian Hußmann und Lars Kowoll

Aktueller Stand: Die Funktionen deleteBT, inOrderBT und calculateHeight sind noch nicht vollständig implementiert.

Entwurf:
Der Entwurf, als einziges Dokument für die Implementierung, beinhaltet alle Vorgaben.

Abstract:
Es ist ein ADT für einen binären Baum zu implementieren. Der Baum besteht aus Knoten, der Knoten selbst enthält eine Zähl als Information, die Höhe des Baums. Ein Knoten hat zwei Nachfolger, von denen einer oder beide NULL sein können. Der Knoten ohne Vorgänger nennt sich Wurzel. Damit es ein binärer Baum ist muss der linke Nachfolger kleiner dem rechten Nachfolger sein.



Beispiel Simulation von https://www.cs.usfca.edu/~galles/visualization/BST.html als Verdeutlichung

Funktionen:

initBT:Ø → btree
Die Funktion initBT initialisiert einen leeren Baum. Der Rückgabewert ist der leere Baum.

isEmptyBT: btree → bool
Die Funktion isEmptyBT überprüft, ob ein leerer oder ein mit Knoten gefüllter Baum übergeben wurde.

1. Prüft, ob ein leerer Baum übergeben wurde und gibt true zurück
2. Prüft, ob ein Baum mit Knoten übergeben wurde und gibt false zurück

equalBT: btree × btree → bool

Die Funktion equalBT testet zwei Bäume auf semantische Gleichheit. Semantisch bedeutet in diesem Kontext, dass die Bäume nicht gleich angeordnet sein müssen, aber die Zahlen in den Knoten gleich sind.

1. Prüft, ob zwei leere Bäume übergeben worden und gibt true zurück
2. Prüft, ob einer der beiden übergebenen Bäume leer ist und gibt false zurück
3. Prüft, ob beide Bäume semantisch gleich sind
    3.1. Die Funktion inOrderBT wird mit beiden Parametern aufgerufen und die zurückgegebenen Listen werden verglichen

insertBT: btree × elem → btree

Die Funktion insertBT fügt ein Element in einen Baum ein. Falls es ein leerer Baum ist, wird das hinzugefügte Element zum Wurzelknoten. Ist dies nicht der Fall wird der Vorgänger zum Element gesucht und mit diesem verknüpft.

1. Bei einem leeren Baum oder Knoten wird das Element mit der Höhe 1 hinzugefügt
2. Prüft, ob das Element im Knoten, größer ist als das hinzuzufügende Element
    2.1. Es wird die Funktion erneut aufgerufen, jetzt allerdings nur mit der linken Teilbaumseite und dem hinzuzufügenden Element
    2.2. Die Höhe der Knoten wird neu berechnet
    2.3. Der Baum wird zusammengesetzt
3. Prüft, ob das Element im Knoten, kleiner ist als das hinzuzufügende Element
    3.1. Es wird die Funktion erneut aufgerufen, jetzt allerdings nur mit der rechten Teilbaumseite und dem hinzuzufügenden Element
    3.2. Die Höhe der Knoten wird neu berechnet
    3.3. Der Baum wird zusammengesetzt

deleteBT: btree × elem → btree

Die Funktion deleteBT entfernt ein Knoten aus einem Baum. Hierfür muss der Knoten der zu löschende Element hält erstmal gefunden werden. Handelt es sich um die Wurzel, ohne Nachfolger wird ein leerer Baum zurückgegeben. Wird ein Knoten mit Nachfolgern gelöscht, muss die Höhe neu berechnet werden. Hat der Knoten nur ein Nachfolger muss die Höhe nicht neu berechnet werden, weil der Teilpfad unverändert bleibt.

1. Prüft, ob der Knoten keine Nachfolger hat und das übergebene Element mit dem Element im Knoten übereinstimmt und gibt einen leeren Knoten zurück
2. Prüft, ob der Knoten einen Nachfolger hat und das übergebene Element mit dem Element im Knoten übereinstimmt. Der Nachfolger wird auf die Position des gelöschten Knoten verschoben. Die Höhe muss nicht neu berechnet werden
3. Prüft, ob der Knoten zwei Nachfolger hat und das übergebene Element mit dem Element im Knoten übereinstimmt
    3.1. Geht in den linken Nachfolger
        3.1.1. Prüft, ob der Knoten keine Nachfolger hat und ersetzt die Position mit dem zu löschenden Knoten
        3.1.2. Prüft, ob der Knoten keinen rechten Nachfolger hat und ersetzt die Position mit dem zu löschenden Knoten
        3.1.3. Prüft, ob der Knoten einen rechten Nachfolger hat und geht den rechten Pfad bis zum letzten Knoten und ersetzt die Position mit dem zu löschenden Knoten
    3.2. Die Höhe der Knoten wird neu berechnet
    3.3. Der Baum wird zusammengesetzt
4. Prüft, ob der Knoten zwei Nachfolger hat und, ob das Element im Knoten größer ist als das übergebene Element
    4.1. Es wird die Funktion erneut aufgerufen, jetzt allerdings nur mit der linken Teilbaumseite und dem hinzuzufügenden Element
    4.2. Die Höhe der Knoten wird neu berechnet
    4.3. Der Baum wird zusammengesetzt
5. Prüft, ob der Knoten zwei Nachfolger hat und ob, das Element im Knoten kleiner ist als das übergebene Element
    5.1. Es wird die Funktion erneut aufgerufen, jetzt allerdings nur mit der rechten Teilbaumseite und dem hinzuzufügenden Element
    5.2. Die Höhe der Knoten wird neu berechnet
    5.3. Der Baum wird zusammengesetzt

findBT: btree × elem → integer

Die Funktion findBT sucht in einem Baum nach einem Element und gibt die Höhe des Knotens zurück. Hierfür muss Element im Baum erstmal gesucht werden, es sei denn es ist im Wurzelknoten.

1. Wird ein leerer Baum übergeben oder ein Element nicht gefunden wird 0 zurückgegeben
2. Wenn das Element im Knoten mit dem als Parameter übergebenen Element übereinstimmt, wird die Höhe des Knoten zurückgegeben
3. Prüft, ob das Element im Knoten, größer ist als das hinzuzufügende Element
    3.1. Es wird die Funktion erneut aufgerufen, jetzt allerdings nur mit der linken Teilbaumseite und dem zu findenden Element
4. Prüft, ob das Element im Knoten, kleiner ist als das hinzuzufügende Element
    4.1. Es wird die Funktion erneut aufgerufen, jetzt allerdings nur mit der rechten Teilbaumseite und dem hinzufügenden Element

inOrderBT: btree → list

Die Funktion inOrderBT fügt alle Elemente eines Baums in eine Liste. Die Liste ist der Größe nach sortiert, wenn der Baum richtig implementiert wurde.

1. Wird ein leerer Knoten übergeben, wird eine leere Liste zurückgegeben
2. Prüft, ob ein Knoten mit nur einem rechten Nachfolger übergeben wurde
    2.1. Fügt das Element in die Liste ein und verschiebt den Index um einen nach rechts und ruft die Funktion mit dem rechten Teilbaum auf
3. Prüft, ob ein Baum mit nur einem linken Nachfolger übergeben wurde
    3.1. Fügt das Element in die Liste ein und verschiebt den Index um einen nach links und ruft die Funktion mit dem linken Teilbaum auf
4. Prüft, ob ein Baum mit zwei Nachfolgern übergeben wurde
    4.1. Schiebt den Index nach links und ruft die Funktion mit dem linken Teilbaum auf, fügt das aktuelle Element in die Liste ein und verschiebt den Index nach rechts und ruft die Funktion mit dem rechten Teilbaum auf.

Interne Funktion:

calculateHeight:btree → integer

Die Funktion calculateHeight berechnet die Höhe eines Knoten. Die Höhe der Teilbäume wird verglichen, aus der höheren Zahl + 1 erfolgt die neue Höhe für den Knoten.

1. Wird ein leerer Baum übergeben wird eine 0 zurückgegeben
2. Die Funktion max wird aufgerufen und die beiden Teilbäume übergeben. Auf den Rückgabewert von max wird 1 addiert

max: integer × integer → integer

Die Funktion max erhält zwei Zahlen und gibt die größere Zahl zurück. Die Zahlen werden durch A und B dargestellt.

1. Prüft, ob A größer ist als B und gibt A zurück
2. Gibt B zurück