Team: Team 05, Deniese Kotthoff, Christine Schuller

Aufgabenaufteilung: Die Aufgabe, sowie der Entwurf wurden gemeinsam bearbeitet

Quellenangaben:

Bearbeitungszeitraum:

Aktueller Stand:

Änderungen des Entwurfs: <Vor dem Praktikum auszufüllen: Welche Änderungen sind bzgl. des Entwurfs während der Implementierung vorgenommen worden.>

Entwurf:

Datenstruktur:

Aufgabenstellung:

Ein Binärbaum mit folgender Struktur soll realisiert werden:

{<Element>,<Höhe>,<linker Teilbaum>,<rechter Teilbaum>}

Ein leerer Baum wird als leeres Tupel **{ }** realisiert

Für alle Unterbäume unseres binären Suchbaumes gilt:

- der **linke** Unterbaum enthält nur Elemente, deren Wert **kleiner** als der Wert des Knotens ist.
- Der **rechte** Unterbaum enthält nur Elemente, deren Wert **größer** als der Wert des Knotens ist.

Der Binärbaum besteht aus folgenden Teilen:

1. **Element**: Unser Element ist unser Wurzelknoten. Dieser wird mit jedem neuen Knoten verglichen - ist der neu hinzuzufügende Knoten größer als der Wurzelknoten, wird er rechts von ihm angeordnet. Ist der neu hinzuzufügende Knoten kleiner, wird er links von ihm angeordnet.

   Das Element besteht aus einem Knoten in dem ein Integer gespeichert ist. Dieser wird mit jedem weiteren Knoten verglichen. Ein Baum entsteht erst wenn der Wurzelknoten mindestens zwei Nachfolger hat.

2. **Höhe**: Die Höhe eines Baumes entspricht der Tiefe des Knotens, welcher am weitesten von dem Wurzelknoten entfernt ist.

   Die Höhe wird als positive, ganze Zahl beschrieben.

3. **Linker Teilbaum**: sind alle Nachfolger von unserem Wurzelknoten, die kleiner seines Wertes sind.

   Jeder Blattknoten des linken Teilbaumes kann ebenfalls weitere Blattnoten haben, die rechts größer als der aktuell Zweigknoten und links kleiner als dieser sind.

4. **Rechter Teilbaum**: sind alle Nachfolger von unserem Wurzelknoten, die größer seines Wertes sind.

   Jeder Blattknoten des rechten Teilbaumes kann ebenfalls weitere Blattnoten haben, die rechts größer als der aktuell Zweigknoten und links kleiner als dieser sind.

Funktionen:

initBT: ι → btree

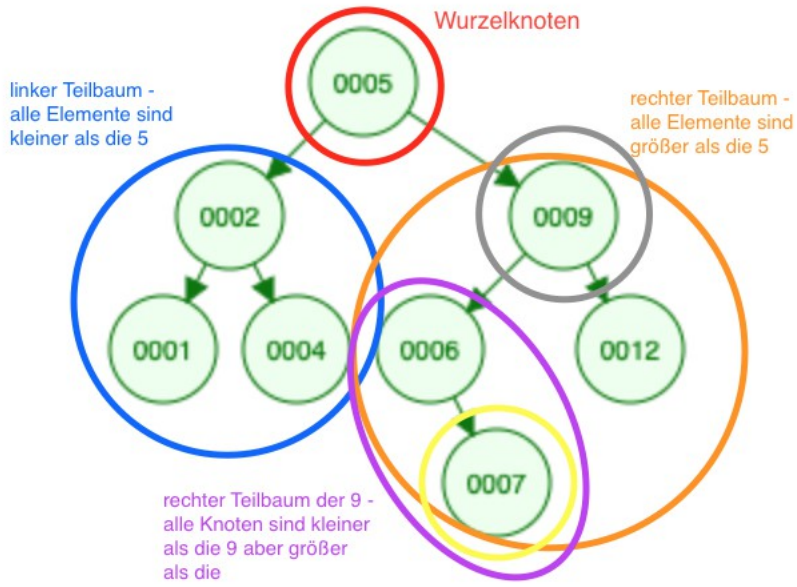
diese Operation initialisiert einen leeren Baum

isEmptyBT: btree → bool

diese Operation prüft ob der übergebene Binärbaum wirklich leer ist und gibt uns dementsprechend true oder false zurück

insertBT: btree x elem → btree

diese Operation fügt dem Baum ein Blatt hinzu. Ist der einzufügende Knoten größer als der Wurzelknoten wird er dem rechten Teilbaum hinzugefügt. Hat der rechte Teilbaum bereits Blattknoten, wird der hinzuzufügende Knoten ebenfalls mit diesen verglichen und an die jeweilige Stelle hinzugefügt. Analog dazu wird der Knoten dem linken Teilbaum hinzugefügt wenn er kleiner ist. Ist der einzufügende Knoten gleich dem Wurzelknoten, wird dieser Fall ignoriert und der Baum wird zurückgegeben.



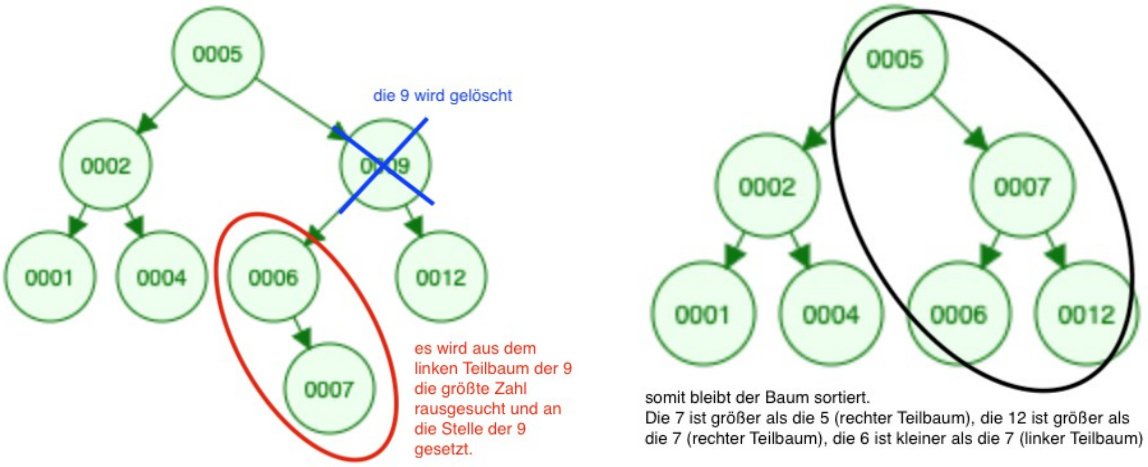
Unser Wurzelknoten ist die 5 (rot). Wir wollen die 7 (gelb) hinzufügen. Es wird geprüft: ist die 7 größer und kleiner als die 5? Sie ist größer → rechter Teilbaum. Jetzt wird die 7 mit dem nächsten Knoten in dem rechten Teilbaum verglichen. Ist die 7 größer oder kleiner als die 9 (grau)? Die 9 ist somit der „neue Wurzelknoten“ mit dem die 7 verglichen wird. Sie ist kleiner als die 9 → linker Teilbaum des Knoten 9 (lila). Nun wird die 7 noch mit der 6 verglichen, sie ist größer als die 6 und wird rechts an den Knoten hinzugefügt. Die Höhe des Baumes wird um 1 addiert.

equalBT: btree x btree → bool

diese Operation prüft zwei Bäume auf semantische Gleichheit. Sie sind semantisch gleich wenn beide Bäume leer sind → es wird ein true zurückgegeben. Ebenfalls semantisch gleich sind sie wenn beide Bäume nicht leer sind, in diesem Fall werden sie in Order gebracht und ein ein Prädikat wird zurückgegeben. Sobald einer der beiden Bäume nicht leer ist wird false zurückgegeben.

deleteBT: btree x elem → btree

diese Operation löscht ein Element aus dem Baum und bringt ihn wieder in die richtige Sortierung. Wenn der Baum leer ist wird dieser zurückgegeben. Ist der Baum nicht leer wird am Wurzelknoten begonnen. Wenn der linke Teilbaum leer ist wird der rechte Teilbaum zurückgegeben. Wenn der rechte Teilbaum leer ist wird der linke Teilbaum zurückgegeben.



Wenn das zu löschende Element die Wurzel ist, wird nach dem größten Element im linken Teilbaum gesucht und dieses als neue Wurzel verwendet. In unserem Beispielbaum ist die 4 das größte Element im rechten Teilbaum, Es wird in die Wurzel geschrieben und der Baum bleibt in Order. Analog dazu können wir auch nach dem kleinsten Element im rechten Teilbaum suchen.

findBT: btree x elem → integer

diese Operation sucht nach einem übergebenen Element im Baum und gibt uns die die Höhe innerhalb des Baumen bis zu diesem Element zurück. Wir laufen bis zum übergebenen Element und gehen den Weg dann zurück bis zum Wurzelknoten wobei auf diesem Weg die Höhe gezählt wird. Wenn das zu suchende Element der Wurzelknoten ist, wird die Höhe der aktuellen Wurzel zurückgegeben. Wenn das zu suchende Element kleiner unserer Wurzel ist dann suchen wir im linken Teilbaum nach ihm, wenn es größer ist dann im rechten Teilbaum. Wenn unser Baum leer ist geben wir 0 zurück, da das gesuchte Element dann nicht existiert.

inOrderBT: btree → liste

diese Operation bekommt einen Baum übergeben welche diesen als sortierte Liste zurück gibt. Wenn beide Teilbäume leer sind wird der Wert der Wurzel in eine Liste geschrieben. Wenn der linke Teilbaum leer ist, wird der Wert der Wurzel in eine Liste gepackt und über einen rekursiven Aufruf über jedes weitere Element aus dem rechten Teilbaum iteriert und jeweils nach dem kleinsten gesucht und in die Liste gepackt. Analog dazu wenn der rechte Teilbaum leer ist, bloß dann iterieren wir über den linken Teilbaum und suchen nach dem größten Element. Wenn beide Teilbäume nicht leer sind wird der Wert der Wurzel in eine Liste geschrieben und die Werte des linken Teilbaumes links des Wurzelknotens (weil sie kleiner der Wurzel sind) und die Werte des rechten Teilbaumes rechts vom Wurzelknoten (weil sie größer der Wurzel sind) in die Liste geschrieben. Jeweils durch den oben beschriebenen rekursiven Aufruf.