technische universität dortmund

fakultät für informatik

Elias Kuthe
Roman Kalkreuth
Bernd Zey

Dortmund, den 23. Dezember 2019

Praktikum zur Vorlesung Einführung in die Programmierung WS 19/20

Blatt 10

Es können 28 Punkte erreicht werden.

Allgemeine Hinweise

1. Bitte lesen Sie vor der Bearbeitung **alle** Aufgaben sorgfältig durch! Dies erspart Ihnen unnötige Arbeit und somit auch Zeit!
2. Lassen Sie sich fertiggestellte Aufgaben bitte möglichst **frühzeitig** testieren. In der letzten halben Stunde vor Schluss werden bei diesem Blatt nur noch **drei** Teil-Aufgaben testiert!
3. Wir akzeptieren ein Testat nur, wenn die Lösung eigenständig auf Anhieb erklärt werden kann. Andernfalls müssen wir die entsprechende Teilaufgabe mit 0 Punkten bewerten.

Aufgabe 1: Binäre Suchbäume (28 Punkte)

In dieser Aufgabe sollen verschiedene Funktionen eines binären Suchbaums mit Hilfe von Templates und dynamischem Speicher für beliebige Elementtypen implementiert werden. Wie in der Vorlesung sollen die Elemente des Suchbaums durch einen Hilfs-Struct `Node` repräsentiert werden, welches ein Datum des Typs `T` und jeweils einen Zeiger auf die Wurzel des linken und des rechten Unterbaums hat. Der Suchbaum selbst besitzt lediglich einen Zeiger auf den Wurzelknoten. Zur Erinnerung: Ein binärer Suchbaum unterscheidet sich gegenüber einem einfachen binären Baum dadurch, dass alle Daten im linken Unterbaum kleiner und die Daten im rechten Unterbaum größer als das Datum eines Knotens sind. Erweitern Sie zur Bearbeitung dieser Aufgabe die zur Verfügung gestellte Header-Datei `bintree.h`.

a) Implementieren Sie in der Klasse `BinTree` die rekursiven Methoden `preOrder`, `inOrder` und `postOrder`, die die Knoten wie in der Vorlesung beschrieben durchlaufen und die dabei gefundenen Daten in entsprechender Reihenfolge auf der Konsole ausgeben.

_____ (3)

b) Implementieren Sie die rekursive `BinTree`-Methode `int height(Node* node)`, welche die Höhe des Suchbaums zurückgibt. Die Höhe eines Suchbaums wurde in der Vorlesung definiert und ist die Länge des längsten Pfades von der Wurzel zu einem Blatt.

_____ (3)

c) Implementieren Sie die rekursive `BinTree`-Methode `int count(Node* node)`, welche die Anzahl der Knoten des Suchbaums ermittelt und zurückgibt.

_____ (3)

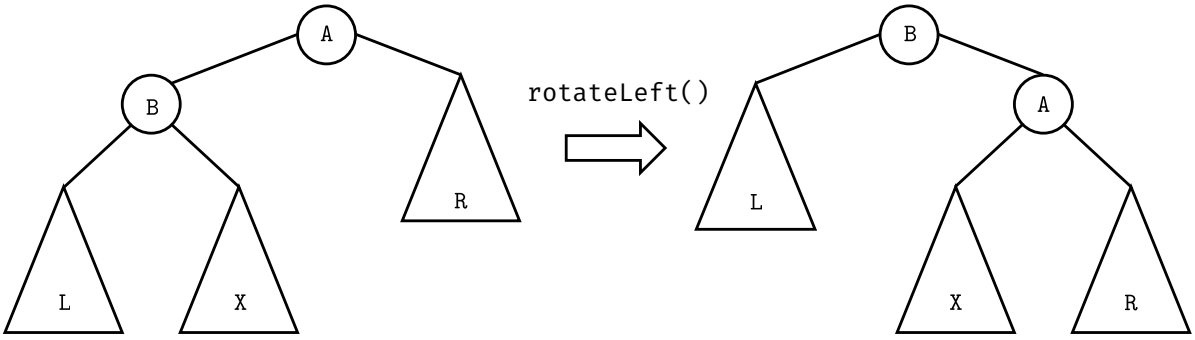
d) Schreiben Sie die `BinTree`-Methode `range`, die alle Werte des Binärbaums, welche in einem Intervall [min , max] liegen, sortiert ausgibt. Gehen Sie dazu folgendermaßen vor:
Durchlaufen Sie den Baum in Inorder-Reihenfolge und rufen Sie – abweichend zur `inOrder`-Methode – die Methode `range` nur dann rekursiv für den linken (bzw. rechten) Unterbaum auf, wenn der Wert des aktuellen Knotens größer als `min` und kleiner als `max` ist.

_____ (4)

e) Schreiben Sie die `BinTree`-Methode `void rotateLeft()`, welche die Wurzel des linken Unterbaums zur neuen Wurzel des Baums macht. Wenn der Baum oder der linke Unterbaum leer sind, ist der Baum unverändert. Ansonsten wird, um die Suchbaum-Eigenschaft zu erhalten, der Baum wie folgt modifiziert:

- Die Wurzel des linken Unterbaums wird die neue Wurzel; die alte Wurzel wird Wurzel des rechten Unterbaums.
- Der linke Unterbaum der neuen Wurzel bleibt unverändert.
- Die alte Wurzel mit ihrem rechten Unterbaum wird rechter Unterbaum der neuen Wurzel.
- Der rechte Unterbaum der neuen Wurzel wird linker Unterbaum der alten Wurzel.

Orientieren Sie sich dabei an dem folgenden Schema.



HINWEIS: Dieses Verfahren ist Teil vom sogenannten *Ausgleichen* von Bäumen.

_____ (4)

f) Implementieren implementieren Sie eine Methode `void save(BinTree<int> &tree)` zum Speichern eines BinTree-Objekts (nur `int`-Ausprägung) in eine Datei. Gehen Sie dabei folgendermaßen vor:

- Überlegen Sie sich, wie der Baum sinnvoll serialisiert werden kann: finden Sie eine Reihenfolge, in der Sie die Daten in einer Datei speichern um später den ursprünglichen Baum wiederherstellen zu können. Welche der oben genannten Strategien zum Durchlaufen eines Baums könnten hierfür geeignet sein?
- Schreiben Sie diese Reihenfolge der Integer-Werte in eine ASCII-Datei. *Achtung*: Wenn Sie die Werte einfach hintereinander in eine Datei schreiben, dann können beim Lesen die Integer-Werte nicht mehr sinnvoll getrennt werden. Verwenden Sie daher ein Komma als Trennzeichen um die einzelnen Zahlen voneinander zu trennen.

- *Tipp:* Es kann sinnvoll sein die Reihenfolge der Daten beim Durchlauf zuerst in eine Liste einzufügen um die Daten anschließend in die Datei zu schreiben. Hierfür können Sie die zur Verfügung gestellte Liste in `eidpliste.h` verwenden.

_____ (5)

g) Implementieren Sie die Funktion `void load(BinTree<int> &tree)` zum Laden eines BinTree-Objekts (nur `int`-Ausprägung) aus einer Datei. Gehen Sie dabei folgendermaßen vor:

- Lesen Sie die Datei in einen String ein.
- Trennen Sie die einzelnen Zahlen, entfernen Sie das Trennzeichen und konvertieren Sie die Teil-Strings vom Datentyp `string` in den Datentyp `int`.
  - Die in der Vorlesung genannten String-Methoden `find`[1] und `substr`[2] können verwendet werden um Teil-Strings zu suchen und einen String zu trennen.
  - Ein C-String (`char`-Array) kann mit Hilfe der Funktion `atoi`[3] in einen Integer konvertiert werden. Der String `myString` kann daher folgendermaßen konvertiert werden: `int value = atoi(myString.c_str());`

_____ (5)

h) Testen Sie ihre Implementierung mit der bereitgestellten Testumgebung.

_____ (1)

[1] `http://www.cplusplus.com/reference/string/string/find/`
[2] `http://www.cplusplus.com/reference/string/string/substr/`
[3] `http://www.cplusplus.com/reference/cstdlib/atoi/`