Algorithmen und Datenstrukturen I

Blatt 6

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 09.12.2019 16:28

Aufgabe 1 — Binärbaum

Das Repository für diese Aufgabe bekommen Sie unter https://classroom.github.com/a/mz3RSg_r.

Implementieren Sie den Binärbaum wie im Repository durch die Header-Datei vorgegeben ist:

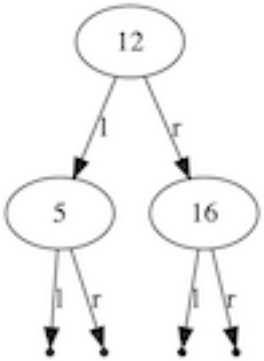
- Die Methoden von `BinTree` stellen das Interface nach außen dar und behandeln im wesentlichen den Fall, dass `root == nullptr` ist. Die Algorithmen zum Suchen, Einfügen und Löschen sollen in den `Node`-Methoden implementiert werden.
- Die Idee bei der Methode `Node::remove` ist, dass als Ergebnis der Teilbaum **nach** dem Löschen zurück gegeben wird und so als entsprechender Nachfolger neu gesetzt werden kann (selbst wenn sich nichts verändert hat). Denken Sie insbesondere daran den `Node` beim Löschen eines Schlüssels tatsächlich auch zu Löschen (`delete`).
- Die Methoden `preorder`, `inorder` und `postorder` zur Traversierung eines Baumes sind bereits implementiert. Sie können diese gut in den Tests verwenden, da die Traversierungen **zusammen** (es reichen genau genommen 2, wenn `inorder` dabei ist) die Struktur des Baumes genau wiedergeben.
- Der Operator `<<` ist für den Baum schon überladen. Er wird genutzt um einen Baum in dot-Syntax (siehe unten) zu repräsentieren.
- Ändern Sie nichts am öffentlichen Interface der Klasse `BinTree`. Private Komponenten und Friends dürfen Sie hinzufügen.

- Die mitgelieferten Tests sind relativ rudimentär. Erweitern Sie diese. Die vorhandenen Tests gehen davon aus, dass Sie beim Löschen den symmetrischen **Nachfolger** und nicht den symmetrischen Vorgänger verwenden.

Im Repository finden Sie außerdem eine `main.cpp` mit einer interaktiven `main`-Funktion zum Arbeiten mit dem Binärbaum. Die Funktion schreibt den Baum in der dot-Syntax in eine Datei `tree.dot`, z.B.

```
digraph tree {
    12 -> 5 [label="l"];
    null5[shape=point];
    5 -> null5 [label="l"];
    null6[shape=point];
    5 -> null6 [label="r"];
    12 -> 16 [label="r"];
    null7[shape=point];
    16 -> null7 [label="l"];
    null8[shape=point];
    16 -> null8 [label="r"];
}
```

Wenn Sie GraphViz installiert haben, wird gleich eine `png`-Datei erzeugt, z.B.



Sie können den Inhalt der `dot`-Datei auch unter `http://www.webgraphviz.com/` nutzen.