Software-Engineering

Praktikum SS 2006

3. Test nicht-funktionaler Anforderungen

1 Lernziele

Vertiefen der in der Vorlesung vermittelten Kenntnisse über Software-Qualität sowie dynamische, white-box Testverfahren für sequentielle, objektorientierte Programme.

2 Aufgabe

Um die Zeiten zum Löschen eines Buches zu verkürzen, soll das Programm zur Verwaltung von Buchbeständen aus Aufgabe 2 jetzt einen balancierten Suchbaum zum Speichern der Bücher verwenden (z.B. AVL-Baum). Ferner soll ein Index angelegt werden, über den ein effizienter Zugriff auf alle Bücher eines Autors möglich ist (d.h. der Iterator kann entfallen)

Von der Klasse `SearchTree` aus Aufgabe 1 soll eine Unterklasse `AvlSearchTree` abgeleitet werden. Bei einem AVL-Baum sind die Teilbäume ausgeglichen, d.h. die Höhe des linken Teilbaums unterscheidet sich höchstens um Eins von der Höhe des rechten Teilbaums. Dies wird dadurch erreicht, dass nach jedem Einfügen bzw. Entfernen eines Knotens der Baum umgeordnet wird. Das hat den Vorteil, dass die Operation `find` in Zeit $O(\log n)$ ausführbar ist, wobei n die Anzahl der Knoten des Baumes ist. Bei degenerierten Suchbäumen ist die Zeit für die `find`-Operation in $O(n)$.

Damit das vollständige Durchsuchen des Baums entfallen kann, soll ein Index angelegt werden, über den ein effizienter, direkter Zugriff auf alle Bücher eines Autors möglich ist. Überschreiben Sie die Methode `getBooksFromAuthor`, so dass der Zugriff über den Index erfolgt.

Überschreiben Sie ferner die Methoden `insert` und `delete` der Klasse `SearchTree` und implementieren Sie geeignete Transformationsmethoden, um den Baum zu balancieren.

Erstellen Sie Testtreiber/-Stubs und Testfälle, um die Korrektheit Ihrer Implementierung nachzuweisen. Führen Sie dazu folgende Tests durch:

- Komponententest der Klasse `AvlSearchTree`
- Regressionstest (alle alten Testfälle müssen korrekte Ergebnisse liefern)
- Performanztest (die Operationen `find`, `insert` und `delete` auf `SearchTree` sowie `getBooksFromAuthor` auf Buchbestand müssen performant sein)

- Test auf Kompatibilität (alle alten Bücherbestände, die in Dateien gespeichert sind, müssen verarbeitet werden können)

Zum Testat müssen Sie das Programm, die Testfälle und die Testarten im Detail erklären können.

3 Literatur

- Spillner, Linz: Basiswissen Softwaretest. dpunkt.verlag
- Sneed, Winter: Testen objektorientierter Software. Hanser Verlag.
- Zeller, Krinke: Programmierwerkzeuge. dpunkt.verlag.
- Ottmann, Widmayer: Algorithmen und Datenstrukturen. BI Wissenschaftsverlag.
- Cormen, Leiserson, Rivest: Introduction to Algorithms. MIT Press.
- Aho, Hopcroft, Ullman: Data structures and algorithms. Addison-Wesley.