Technische Universität Wien
Institut für Computergraphik und Algorithmen
Arbeitsbereich für Algorithmen und Datenstrukturen

186.813 Algorithmen und Datenstrukturen 1 VU 6.0

Sommersemester 2012

Programmieraufgabe

abzugeben bis: Montag, 30. April 2012, 15:00 Uhr

Organisatorisches

Im Rahmen der Lehrveranstaltung **Algorithmen und Datenstrukturen 1** gilt es, eine Programmieraufgabe selbständig zu lösen, die das Verständnis des im Vorlesungsteil vorgetragenen Stoffes vertiefen soll. Als Programmiersprache wird Java 6 verwendet.

Geben Sie bitte Ihr fertiges, gut getestetes und selbst geschriebenes Programm bis spätestens **Montag, 30. April 2012, 15:00 Uhr** über das Abgabesystem in TUWEL ab. Der von Ihnen abgegebene Code wird vom System automatisch getestet, und Sie erhalten eine entsprechende Rückmeldung im Abgabesystem.

Um eine positive Note zu erhalten, müssen Sie bei der Programmieraufgabe **mindestens einen Punkt erreichen** und Ihren Termin zum Abgabegespräch einhalten.

> Gruppenabgaben bzw. mehrfache Abgaben desselben Programms unter verschiedenen Namen werden nicht akzeptiert. Wenn Sie das abgegebene Programm nicht selbst programmiert haben, erhalten Sie ein negatives Zeugnis. Auch der eigentliche Entwickler kann nur mehr maximal einen Punkt auf dieses Programmierbeispiel erhalten.

Abgabefrist

Sie haben bis Montag, 30. April 2012, 15:00 Uhr die Möglichkeit ein Programm abzugeben, das alle Testinstanzen korrekt abarbeitet. Danach werden keine weiteren Abgaben akzeptiert, d.h., sollten Sie diese Frist versäumen beziehungsweise Ihr Programm nicht alle Testinstanzen korrekt abarbeiten, bekommen Sie null Punkte auf die Programmieraufgabe.

Abgabegespräche

In der Zeit von Mittwoch, den 2. Mai 2012, bis Freitag, den 4. Mai 2012, finden für alle LVA-Teilnehmer Abgabegespräche statt. Dazu vereinbaren Sie in TUWEL einen individuellen Gesprächstermin, bei dem Sie sich mit einer/m unserer TutorInnen im Informatiklabor

treffen und den von Ihnen eingereichten Programmcode erklären können müssen. Sie können sich zu diesem Abgabegespräch von 27. April 2012 bis 1. Mai 2012 15:00 Uhr in TUWEL bei einer/m TutorIn anmelden. Sollten in diesem Zeitraum keine Termine in TUWEL eingetragen oder bereits alle Termine vergeben sein, dann kontaktieren Sie bitte die Hotline `algodat1-ss12@ads.tuwien.ac.at`.

Falls Sie Systemroutinen in Ihrem Programm verwenden (z.B. das Auffinden eines Minimums), so sollten Sie auch über die Funktionsweise dieser Methoden **genau** Bescheid wissen. Je nach Funktionstüchtigkeit Ihres Programms, Innovation und Effizienz Ihrer Implementierung sowie der Qualität des Abgabegesprächs können Sie bis zu 10 Punkte für diese Programmieraufgabe erhalten. Voraussetzung für eine positive Absolvierung des Abgabegesprächs ist jedenfalls das Verständnis des zugrunde liegenden Stoffes.

Betreuung im Informatiklabor

Um auftretende Verständnisfragen beziehungsweise Unklarheiten bei der Angabe klären zu können, stehen Ihnen am Montag, den 23. April 2012, von 12:30 bis 13:30 und am Mittwoch, den 25. April 2012, von 12:30 bis 13:30 unsere Tutoren im Informatiklabor für persönliche Gespräche zur Verfügung. Bedenken Sie allerdings, dass dies nur eine Hilfestellung ist und eine selbständige Auseinandersetzung mit der Problemstellung nicht ersetzen soll und kann – die TutorInnen sind nicht als „menschliche Debugger“ anzusehen.

Aufgabenstellung

Sie haben in der Vorlesung AVL-Bäume, eine Form der balancierten Suchbäume, kennengelernt. Um diese Balance erhalten zu können erfordern die Operationen Einfügen und Löschen in vielen Fällen eine Rebalancierung des AVL-Baums.

In der Vorlesung haben Sie sich mit den Aspekten dieser Rebalancierung und den damit entstehenden Fallunterscheidungen aus theoretischer Sicht auseinandergesetzt. Im Zuge dieser Programmieraufgabe sollen Sie diese Erkenntnisse nun in die Praxis umsetzen, indem Sie das Einfügen und das Löschen von Elementen aus einem AVL-Baum inklusive aller notwendigen Rebalancierungsmaßnahmen implementieren und testen.

Codegerüst

Der Algorithmus ist in Java 6 zu implementieren. Um Ihnen einerseits das Lösen der Aufgabenstellung zu erleichtern und andererseits automatisches Testen mit entsprechender Rückmeldung an Sie zu ermöglichen, stellen wir Ihnen ein Codegerüst zur Verfügung[1].

Das Codegerüst besteht aus mehreren Klassen, wobei Sie Ihren Code in die Datei `AvlTree.java` einfügen müssen.

Konkret müssen Sie zumindest die folgenden Methoden der Klasse `AvlTree` implementieren:

- `insert(int k)`: Hier soll der Schlüssel `k` in den AVL-Baum eingefügt und alle nötigen Operationen durchgeführt werden, damit der resultierende Baum wieder ein gültiger AVL-Baum ist. Bereits vorhandene Schlüssel dürfen nicht erneut eingefügt werden.
- `remove(int k)`: Hier soll der Schlüssel `k` aus dem AVL-Baum entfernt werden und alle nötigen Operationen durchgeführt werden, damit der resultierende Baum wieder ein gültiger AVL-Baum ist. Sofern ein Ersatzknoten für den zu löschenden Knoten benötigt wird, soll wie im Skriptum der Successor verwendet werden. Nicht existente Schlüssel ändern den AVL-Baum nicht.
- `AvlNode rotateLeft(AvlNode n)`: Diese Methode soll Ihren Code für eine Links-Rotation enthalten.
- `AvlNode rotateRight(AvlNode n)`: Diese Methode soll Ihren Code für eine Rechts-Rotation enthalten.

Die Klasse `AvlNode` kapselt die Knoten des AVL-Baums und enthält die Felder `left,right,parent,key,balance`. Das Feld `balance` wird vom Framework nicht überprüft, soll jedoch die Balance des Knotens speichern. Die Felder `left` bzw. `right` sollen, wie der Name vermuten lässt, auf das linke bzw. rechte Kind zeigen. Das Feld `parent`

[1] `http://www.ads.tuwien.ac.at/teaching/lva/186813.html#Programmieraufgaben`

soll auf den Elternknoten zeigen. Hat ein Knoten kein linkes bzw. rechtes Kind, ist das entsprechende Feld auf `null` zu setzen.

Weitere Details, die das Codegerüst betreffen, entnehmen Sie bitte der LVA-Webseite[2]. Dort befindet sich auch der Link zur Dokumentation (Javadoc) des Codegerüstes.

Hinweise

`Main.printDebug(String msg)` kann verwendet werden um Debuginformationen auszugeben. Die Ausgabe erfolgt nur, wenn beim Aufrufen das Debugflag (`-d`) gesetzt wurde. Sie müssen diese Ausgaben vor der Abgabe **nicht** entfernen, da das Testsystem beim Kontrollieren Ihrer Implementierung dieses Flag nicht setzt.

Das Framework nutzt Methoden, die das Ausführen von *sicherheitsbedenklichem* Code auf dem Abgabesystem verhindern soll. Werden trotzdem solche Programmteile abgegeben oder sollte versucht werden, das Sicherheitssystem zu umgehen, wird das als Betrugsversuch gewertet. Die minimale Konsequenz dafür ist ein negatives Zeugnis auf diese Lehrveranstaltung.

Rückgabe des Frameworks

Nachdem alle Einfüge- und Lösch-Operationen durchgeführt wurden, gibt das Framework die `parent`-Werte der einzelnen Knoten gemäß der Inorder-Durchmusterungsreihenfolge ihres AVL-Baums aus. Diese muss mit der Ausgabe der Musterlösung übereinstimmen. Wenn während der Abarbeitung Ihres Programms ein Fehler auftritt, wird stattdessen eine entsprechende Fehlermeldung ausgegeben.

Testdaten

Auf der Webseite zur LVA sind auch Testdaten veröffentlicht, die es Ihnen erleichtern sollen, Ihre Implementierung zu testen. Verarbeitet Ihr Programm diese Daten korrekt, heißt das aber nicht zwangsläufig, dass Ihr Programm alle Eingaben korrekt behandelt. Testen Sie daher auch mit zusätzlichen, selbst erstellten Daten.

Die Testinstanzen sind folgendermaßen aufgebaut: Jede Zeile enthält entweder ein Schlüsselwort (#insert für Einfügen oder #remove für Löschen) oder einen Schlüssel. Ein Schlüsselwort ändert den Zustand des Input-Scanners. Das heißt, alle Schlüssel nach einem #insert Befehl werden bis zum nächsten #remove Befehl eingefügt und umgekehrt.

Sie können davon ausgehen, dass **alle** von uns erstellten Testinstanzen dieser Spezifikation entsprechen, und müssen selbst keine Fehlerprüfung durchführen.

[2] `http://www.ads.tuwien.ac.at/teaching/lva/186813.html`

Abgabe

Die Abgabe Ihrer Implementierung der Klasse `AvlTree` erfolgt über TUWEL. Bedenken Sie bitte, dass Sie maximal 30 Mal abgeben können und ausschließlich die jeweils letzte Abgabe bewertet wird. Prinzipiell sollte es nicht nötig sein, mehr als eine Abgabe zu tätigen, wenn Sie Ihr Programm entsprechend getestet haben.

Hinweis

Verwenden Sie bei Ihrer Implementierung unter keinen Umständen Sonderzeichen, wie zum Beispiel Umlaute (ä, ö, ü, Ä, Ö, Ü) oder das Zeichen „ß“. Dies kann sonst – bei unterschiedlichen Zeichensätzen am Entwicklungs- und Abgabesystem – zu unvorhersehbaren Problemen und Fehlern führen, was schlussendlich auch als fehlerhafter Abgabeversuch gewertet wird.

Die Überprüfung Ihres abgegebenen Codes erfolgt automatisch, wobei in drei Schritten getestet wird:

Kompilation Es wird der **Bytecode** erzeugt, der beim anschließenden Testen verwendet wird.

veröffentlichte Testdaten Bei diesem Schritt wird Ihr Programm mit den auf der Webseite veröffentlichten Daten getestet.

unveröffentlichte Testdaten Abschließend wird Ihr Programm noch mit Ihnen nicht bekannten aber den Spezifikationen entsprechenden Eingabedaten ausgeführt.

Nach Beendigung der Tests, die direkt nach Ihrer Abgabe gestartet werden, erhalten Sie eine Rückmeldung über das Abgabesystem in TUWEL mit entsprechenden kurzen Erfolgs- bzw. Fehlermeldungen. Aufgrund der großen Hörerzahl kann es zu Verzögerungen beim Verarbeiten Ihrer Abgaben kommen.

Geben Sie daher Ihre Lösung nicht erst in den letzten Stunden ab, sondern versuchen Sie, rechtzeitig die Aufgabenstellung zu lösen. Beachten Sie bitte auch, dass wir Ihren Code mit den von Ihren Kollegen abgegebenen Programmen automatisch vergleichen werden, um Plagiate zu erkennen. Geben Sie daher nur selbst implementierte Lösungen ab!

Theoriefragen

Beim Abgabegespräch müssen Sie u.a. Ihre Überlegungen zu folgenden Punkten präsentieren können:

- Was sind höhenbalancierte Bäume (Definition und Beispiele)?
- Was sind die Eigenschaften eines AVL-Baums? Was bedeutet balanciert, bzw Höhe/Tiefe von Knoten/Bäumen in einem AVL-Baum?

- Wie ist die Höhe eines AVL-Baums in Θ-Notation in Abhängigkeit der Knotenanzahl des Baumes?
- Ist der AVL-Baum für eine Menge von Schlüsseln eindeutig?
- Lässt sich ein AVL-Baum nur aufgrund vorgegebener Inorder- und Preorder-Durchmusterungsreihenfolge eindeutig rekonstruieren? Wenn ja, wie?
- Kann man jeden gültigen AVL-Baum nur durch Einfüge-Operationen konstruieren, oder gibt es Bäume die nur mit Hilfe von Lösch-Operationen entstehen können?
- Welche Konsequenzen hätte es für AVL-Bäume, würde man es erlauben, gleiche Schlüssel mehrfach einzufügen?

Testumgebung

Unser Testsystem mit Intel Xeon X5650 CPU ruft Ihr Programm mit folgendem Kommandozeilenbefehl auf:

```
java ads1ss12.pa.Main input > output
```

wobei `input` eine Eingabedatei darstellt. Die Ausgabe wird in die Datei `output` gespeichert. Pro Testinstanz darf eine maximale Ausführungszeit von 15 Sekunden nicht überschritten werden, anderenfalls wird die Ausführung abgebrochen.

Bewertung

Abschließend seien nochmals die wesentlichen Punkte zusammengefasst, die in die Bewertung Ihrer Abgabe einfließen und schließlich über die Anzahl der Punkte für diese Programmieraufgabe entscheiden:

- Korrektheit des Algorithmus
- Laufzeiteffizienz
- Speicherverbrauch
- Rechtfertigung des Lösungsweges
- Antworten auf die theoretischen Fragen der Aufgabenstellung

Voraussetzung für eine positive Absolvierung des Abgabegesprächs ist jedenfalls grundsätzliches Verständnis der Problemstellung.

Zusätzliche Informationen

Lesen Sie bitte auch die auf der Webseite zu dieser LVA veröffentlichten Hinweise. Wenn Sie Fragen oder Probleme haben, wenden Sie sich rechtzeitig an an die AlgoDat1-Hotline unter `algodat1-ss12@ads.tuwien.ac.at`.