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

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

Wintersemester 2008

Programmieraufgabe 1

abzugeben bis: 09. Dezember 2008, 15:00 Uhr

Organisatorisches

Im Rahmen der Lehrveranstaltung *Algorithmen und Datenstrukturen 1* gilt es, zwei voneinander unabhängige Programmieraufgaben selbständig zu lösen, die das Verständnis des im Vorlesungsteil vorgetragenen Stoffes vertiefen sollen.

Geben Sie bitte Ihr fertiges, gut getestetes und selbst geschriebenes Programm bis spätestens *Dienstag, 09. Dezember 2008, 15:00 Uhr*, mittels des eigens dafür eingerichteten Abgabesystems auf der Webseite zur Lehrveranstaltung ab. Der von Ihnen abgegebene Code wird vom System automatisch getestet und Sie erhalten per Mail eine entsprechende Abgabebestätigung.

In der Zeit von Mittwoch, den 10. Dezember 2008, bis Freitag, den 12. Dezember 2008, finden für alle LVA-Teilnehmer Abgabegespräche statt. Verwenden Sie bitte das Abgabesystem, um mit Ihrem Tutor beziehungsweise Ihrer Tutorin einen individuellen Gesprächstermin zu vereinbaren, bei dem Sie den von Ihnen eingereichten Programmcode erklären können müssen. Sobald Sie ein funktionstüchtiges Programm abgegeben haben, spätestens aber nach Ablauf der Abgabefrist am 09.12., ist für Sie die Anmeldung zu einem Abgabegespräch freigeschaltet. Falls Sie Systemroutinen 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 und dem Verständnis des benötigten Stoffes beziehungsweise Qualität des Abgabegesprächs können Sie 0 bis 5 Punkte für diese Programmieraufgabe erhalten.

Um auftretende Verständnisfragen beziehungsweise Unklarheiten bei der Angabe klären zu können, stehen Ihnen in der Woche von Montag, den 1. Dezember 2008, bis Freitag, den 5. Dezember 2008, unsere Tutorinnen und Tutoren im Informatiklabor für persönliche Gespräche zur Verfügung. Die genauen Zeiten entnehmen Sie bitte der LVA-Webseite. Bedenken Sie allerdings, dass dies nur eine Hilfestellung ist und eine selbstständige Auseinandersetzung mit der Problemstellung nicht ersetzen soll und kann – die Tutoren sind nicht als „menschliche Debugger“ anzusehen.

Bitte beachten Sie, dass wir *ausnahmslos* keine verspäteten Abgaben akzeptieren. Sollten Sie kein Programm abgegeben haben oder das von Ihnen zuletzt abgegebene Programm ist

nicht in der Lage, alle Instanzen korrekt abzuarbeiten, haben Sie die Möglichkeit, beim Abgabegespräch eine korrekte Version Ihres Codes mitzubringen, können aber dann nur noch maximal 3 Punkte erreichen. Zusätzlich liegt es in Ihrem Verantwortungsbereich, dass der Sourcecode bei diesem Gespräch verfügbar ist, im Informatiklabor kompilierbar und somit die Lauffähigkeit Ihres Programms von Ihrem Tutor entsprechend getestet werden kann.

Um eine positive Note zu erhalten, müssen Sie bei jeder der zwei Programmieraufgaben mindestens einen Punkt erreichen. Ebenso müssen Sie Ihren Termin zum Abgabegespräch einhalten – anderenfalls ist es uns nicht möglich, ein positives Zeugnis auszustellen.

Gruppenabgaben bzw. mehrfache Abgaben des selben Programms unter verschiedenen Namen werden nicht akzeptiert und führen für alle Beteiligten (auch für den eigentlichen Entwickler) dazu, dass Sie die LVA nicht positiv abschließen können!

Aufgabenstellung

Ein binärer Suchbaum heißt AVL-Baum, wenn sich für jeden seiner Teilbäume die Höhen des linken und rechten Astes höchstens um 1 unterscheiden (siehe Skriptum).

Die Operationen Suchen, Minimum, Maximum, Successor und Predecessor werden bei AVL-Bäumen genauso wie bei natürlichen binären Suchbäumen ausgeführt. Aufpassen muss man hingegen bei den Operationen Einfügen und Entfernen, da hier eventuell die Balance verloren geht.

Sie sollen nun einen effizienten Algorithmus implementieren, der die Löschung eines Knotens in einem AVL-Baum inklusive der dafür notwendigen Rebalancierungen vornimmt. Beachten Sie, dass die von Ihnen geschriebenen Rotations-Methoden im zur Verfügung gestellten Framework auch für das Einfügen von Schlüsseln in den AVL-Baum verwendet werden.

Hinweis: Beachten Sie die Rebalancierungsregeln im Skriptum!

Konkret müssen Sie dazu die folgenden Methoden der Klasse `AvlTreeImpl` implementieren:

- `rotateLeft` und `rotateRight`:
  - **Eingabe:** Der Knoten, um den rotiert werden soll (einfache Rotation).
  - **Rückgabewert:** Der neue Wurzelknoten des rotierten Teilbaums.
- `remove`:
  - **Eingabe:** Der Schlüssel, der gelöscht werden soll.
  - **Rückgabewert:** Keiner.

Beim Abgabegespräch müssen Sie auch Ihre Überlegungen zu folgenden Punkten präsentieren können:

- Wodurch stellen Sie sicher, dass ihr Algorithmus beim Löschen immer einen gültigen AVL-Baum erzeugt?
- Erkären Sie kurz die vier verschiedenen Rotationstypen. Kann es vorkommen, dass zur Rebalancierung unterschiedliche Vorgehensweisen zu einem gültigen Ergebnis führen oder ist die Wahl der zu verwendenden Rotationsmethode(n) immer eindeutig?
- Was müssen Sie bei iterativer bzw. rekursiver Implementierung beachten?
- Welche Laufzeit ergibt sich bei ihrem Algorithmus für u Löschoperationen bei einem AVL-Baum mit anfangs n Knoten, wobei der Aufwand in Abhängigkeit von u und n in Θ-Notation angegeben werden soll?
- Welche Laufzeit ergibt sich beim Einfügen von n Elementen in einen anfangs leeren AVL-Baum in Abhängigkeit von n in Θ-Notation?

Das über die LVA-Webseite zur Verfügung gestellte Codegerüst stellt Ihnen alle notwendigen Methoden zur Verfügung, um auf den AVL-Baum zugreifen zu können.

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]. Dadurch sind allerdings auch Sie dazu angehalten, bestimmte Regeln zu befolgen, um ein funktionierendes Programm (im Sinne der automatischen Überprüfung) zu schreiben. Neben diesen Einschränkungen nehmen wir für dieses Programmierbeispiel an, dass die Eingabedaten den Spezifikationen (siehe weiter unten) entsprechen.

Das Codegerüst besteht aus fünf Klassen, wobei Sie Ihren Code in die Datei `AvlTreeImpl.java` einfügen müssen. Um Debugging-Informationen auszugeben steht die Methode `printDebug()` zur Verfügung. Vergessen Sie nicht, diese Aufrufe vor der Abgabe zu entfernen! Weitere Details, die das Codegerüst betreffen, entnehmen Sie bitte der LVA-Webseite[2]. Dort befindet sich auch der Link zur Javadoc des Codegerüstes.

Hinweis: Nicht jede Methode oder Membervariable, die vom Codegerüst zur Verfügung gestellt wird, ist zwingend für Ihre Implementierung notwendig. Allerdings muss in jedem Fall die Höhe eines Knotens, die in der Variable `height` abgespeichtert wird, korrekt sein.

[1] `http://www.ads.tuwien.ac.at/teaching/lva/186172.html#Programmieraufgaben`
[2] `http://www.ads.tuwien.ac.at/teaching/lva/186172.html`

Folgende Dinge, die unter Umständen Einfluss auf die Bewertung haben könnten, seien aber noch erwähnt:

- Die Effizienz des Algorithmus,
- Speicherverbrauch,
- Rechtfertigung des Lösungsweges,
- Ihre Antworten auf die theoretischen Fragen der Aufgabenstellung.

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 durchaus mit zusätzlichen, selbst erstellten Daten. Eine Eingabedatei enthält zumindest eine Steuersequenz `#insert`, gefolgt – jeweils in einer eigenen Zeile – von den Knotenschlüsseln, die in den AVL-Baum eingefügt werden sollen. Wenn Knoten aus dem AVL-Baum gelöscht werden sollen, geschieht dies mit der Steuersequenz `#remove`. In jeder Zeile einer Eingabedatei steht jeweils entweder eines der Schlüsselwörter `#insert` oder `#remove`, oder ein Schlüssel, der eingefügt oder entfernt werden soll.

Zusätzlich zu den Testdaten wird auch ein Visualisierungstool angeboten, das die vom Codegerüst optional erzeugten Debugging-Informationen visualisieren kann.

Abgabe

Die Abgabe erfolgt über das eigens eingerichtete Abgabesystem auf der Webseite zur LVA. Bedenken Sie bitte, dass Sie maximal 10 Mal abgeben können und dass 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. Verwenden Sie dazu neben den von uns zur Verfügung gestellten Testdaten (inklusive Lösung) auch selbst erstellte. Sollte die Abgabe dennoch nicht erfolgreich sein – Sie erhalten eine entsprechende Mail von uns – testen und debuggen Sie vor einer neuerlichen Abgabe bitte nochmals ausführlich.

Geben Sie alle von Ihnen geschriebenen Sourcedateien ab. Erstellen Sie eine ZIP-Datei (.zip, kein .gz, .tgz, tar.gz, etc.), die den gesamten von Ihnen geschriebenen Code beinhaltet – auch wenn Ihr gesamter Code in nur einer Datei enthalten ist. Alle von Ihnen geschriebenen Klassen müssen im Paket `ads1ws08.pa1` liegen. Nehmen Sie keine Änderungen an den Dateien `Main.java`, `Scanner.java`, `AbstractAvlTree.java` und `AvlNode.java` vor und geben Sie diese daher auch *nicht* ab, da sie vom Abgabesystem ignoriert werden. Vergessen Sie nicht, dass Ihr abgegebenes Programm keine Debugging-Information ausgeben darf! Stellen Sie weiters bitte sicher, dass Ihr Programmarchiv nur `.java` Dateien enthält – alle anderen Dateien werden vom System automatisch gelöscht.

Hinweis: Verwenden Sie bei ihrer Implementation unter keinen Umständen Umlaute (ä,ö,ü,Ä,Ö,Ü) sowie das Zeichen „ß“ außerhalb von Kommentaren. Dies kann sonst – bei unterschiedlichen Zeichensätzen am Entwicklungs- und Abgabesystem – zu unvorhersehbaren Problemen und Fehlern führen, was schlussendlich auch als fehlerhafter Abgabeversuch gewertet wird. Sollten Sie verspätet abgeben und ihre Abgabedatei im Informatiklabor aus den oben genannten Gründen nicht kompilierbar sein, so erhalten Sie automatisch keine Punkte auf das Programmierbeispiel.

An alle Benutzer von Betriebssystemen aus dem Hause Apple: Bitte beachten Sie, dass die systeminterne `zip`-Funktion keine herkömmlichen Zip-Dateien erzeugt. Es werden standardmäßig zusätzlich zu den zu packenden Dateien weitere in das Archiv hinzugefügt, deren Dateinamen nur aus Sonderzeichen – aber mit der Endung `.java` – bestehen. Stellen Sie bitte sicher, dass diese Dateien nicht in Ihrem Zip-Archiv enthalten sind. Anderfalls kommt es zu einer negativen Bewertung Ihrer Programmieraufgabe, da diese zusätzlichen Dateien vom Abgabesystem nicht automatisch als ungültig erkannt werden können, aber zu Compilerfehlern führen.

Die Überprüfung Ihres abgegeben 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 Ihre Lösung noch mit Ihnen nicht bekannten, aber den Spezifikationen entsprechenden Eingabedaten ausgeführt. Die hier verwendeten Testdaten werden nach der Deadline zur Programmabgabe auf der Webseite zur LVA veröffentlicht.

Nach Beendigung der Tests, die direkt nach Ihrer Abgabe gestartet werden, erhalten Sie von uns eine e-Mail an die uns bekannte Adresse (siehe dazu auch Webseite) mit entsprechenden kurzen Erfolgs- bzw. Fehlermeldungen. Aufgrund der großen Hörerzahl kann es zu Verzögerungen beim Zusenden von Nachrichten kommen. Haben Sie daher bitte ein wenig Geduld beziehungsweise geben Sie 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 keine nicht selbst implementierte Lösung ab!

Testumgebung

Unser Testsystem ruft Ihr Programm mit folgendem Kommandozeilenbefehl auf:

```
java ads1ws08.pa1.Main < input > output
```

wobei `input` eine Eingabedatei darstellt. Die Ausgabe wird in die Datei `output` gespeichert. Um die Funktionsweise zu testen, wird die von Ihrem Code erzeugte Ausgabedatei mittels des Unix-Kommandos `diff` mit den Musterlösungen auf Gleichheit verglichen. Bedenken Sie bitte daher, sämtliche Debuggingausgaben vor der Programmabgabe zu entfernen.

Zusätzliche Informationen

Lesen Sie bitte auch die auf der Webseite zu dieser LVA veröffentlichten Hinweise. Sie finden im Skriptum zur Vorlesung – und natürlich auch in der Vorlesung selbst – eine Einführung zu AVL-Bäumen.

Es besteht selbstverständlich auch die Möglichkeit, mit anderen Studenten über die Aufgabenstellung zu diskutieren beziehungsweise sich gegenseitig zu helfen. Bedenken Sie allerdings, dass eine Abgabe des Programmcodes eines Kollegen als nicht ausreichende Leistung erachtet wird. Auf alle Fälle sollten Sie das von Ihnen abgegebene Programm sehr gut erklären können.

Im Falle von Fragen oder weiteren Problemen, helfen Ihnen auch Ihr Tutor bzw. Ihre Tutorin, die Studienassistenten und die Mitarbeiter unseres Instituts – in dieser Reihenfolge – gerne weiter. Sie können sich auch an die AlgoDat1-Hotline unter `algodat1-ws08@ads.tuwien.ac.at` wenden.