Technische Universität Wien
Institut für Computergraphik und Algorithmen
Algorithms and Complexity Group

186.813 Algorithmen und Datenstrukturen 1 VU 6.0

Sommersemester 2015

Programmieraufgabe

abzugeben bis: Montag, 18. Mai 2015, 15:00

Organisatorisches

Im Rahmen der Lehrveranstaltung **Algorithmen und Datenstrukturen 1** gilt es, eine Programmieraufgabe selbstständig 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, 18. Mai 2015, 15:00** 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, 18. Mai 2015, 15:00 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 verpflichtenden Testinstanzen korrekt abarbeiten, bekommen Sie keine Punkte auf die Programmieraufgabe.

Abgabegespräche

In der Zeit von Dienstag, 19. Mai bis Freitag, 22. Mai 2015, 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 Freitag, 15. Mai, bis Montag, 18. Mai 2015, 23:59 in TUWEL bei einer/m TutorIn anmelden.

Falls Sie Datenstrukturen aus der Java-API in Ihrem Programm verwenden, so sollten Sie auch darüber **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.

Aufgabenstellung

Sie müssen für diese Programmieraufgabe einen binären Suchbaum mit zusätzlichen Informationen in den Knoten implementieren.

Beschreibung

Jeder Knoten des Suchbaumes sollte folgende Informationen enthalten:

- Verweis auf den linken Nachfolger
- Verweis auf den rechten Nachfolger
- Ein ganzzahlige Wert (Integer), der als Schlüssel verwendet wird
- Anzahl der Knoten im (Unter-)Baum, für den der jeweilige Knoten die Wurzel darstellt

Die Implementierung dieses speziellen Suchbaumes sollte folgende Methoden beinhalten:

- `boolean isEmpty()`: Liefert zurück, ob der Baum leer ist.
- `int size()`: Liefert die Größe in Form der Anzahl der Knoten des Baumes zurück.
- `boolean exists(int val)`: Liefert zurück, ob der Wert `val` im Baum abgespeichert ist.
- `int height()`: Liefert die Höhe des Baumes zurück. Die Höhe wird für diese Aufgabe folgendermaßen definiert: Ein leerer Baum hat die Höhe -1, ein Baum mit einem Knoten die Höhe 0. Mit jeder Stufe von Nachfolgern erhöht sich die Höhe um 1.
- `void insert(int val)`: Fügt den übergebenen Wert `val` in den Baum ein. Falls der Wert schon vorhanden ist, dann wird er nicht neu eingefügt. Ein Wert kann daher nur einmal im Baum vorkommen.

- `void delete(int val)`: Entfernt den Wert `val` und den dazugehörigen Knoten aus dem Baum. Als Ersatzknoten (falls benötigt) wird der Knoten mit dem kleinsten Schlüssel des rechten Unterbaumes (Successor) gewählt. Falls `val` nicht im Baum enthalten ist, ändert sich der Baum nicht.

- `int valueAtPosition(int k)`: Liefert den Wert, der sich an der `k`-ten Stelle in der Inorder-Durchmusterungsreihenfolge des Baumes befindet zurück. Das erste Element steht an Position 0. Wird eine ungültige Position (negatives oder zu großes `k`) übergeben, dann soll eine IllegalArgumentException mit entsprechender Fehlermeldung geworfen werden.

- `int position(int val)`: Liefert die Position des Wertes `val` in der Inorder-Reihenfolge aller Werte im Baum zurück. Wiederum gilt für diese Aufgabe, dass das erste Element an Position 0 steht. Falls das Element nicht im Baum vorhanden ist, dann soll jene Position zurückgeliefert werden an der es eingefügt werden würde.

- `Iterable<Integer> values(int lo, int hi)`: Liefert alle Werte zwischen `lo` (inklusive) und `hi` (inklusive) zurück. Falls `lo > hi`, dann soll die Methode auch korrekt funktionieren, d.h. es werden alle Werte größer oder gleich `lo` oder kleiner oder gleich `hi` zurückgeliefert. Nur für diese Aufgabe dürfen Sie eine zusätzliche Datenstruktur aus dem Java-Collection-Framework verwenden. Es sollte möglich sein, über diese Datenstruktur zu iterieren (z.B. Liste) und alle Werte auszugeben.

- `void simpleBalance()`: Wenn diese Methode aufgerufen wird, dann wird der gesamte Baum balanciert, sodass die Höhendifferenz zwischen allen Blättern maximal eins ist. Dazu dürfen Sie aber keine Rotationsoperationen verwenden, sondern sollen sich ein Verfahren überlegen, das die Methode `valueAtPosition` ausnutzt und damit zunächst einen zweiten Baum aufbaut. Dieser ersetzt letztlich den ursprünglichen Baum.
  Hinweis: Bedenken Sie dabei, dass die Werte im ursprünglichen Suchbaum aufsteigend gespeichert werden. Wenn man von allen Werten den Median ermittelt und diesen als Wurzel des neuen Baumes wählt, dann hat man für den linken und den rechten Teilbaum dieser Wurzel in etwa gleich viele Werte zur Verfügung.

Bitte beachten Sie noch folgende Hinweise zur Implementierung:

- Wenn nichts anderes angegeben wurde, dann sollen alle Ihre Methoden nur auf der Baumstruktur operieren. Es ist nicht erlaubt, den Inhalt des Baumes in eine zusätzliche Datenstruktur zu speichern und mit dieser die Aufgabenstellung zu dieser Methode zu lösen.

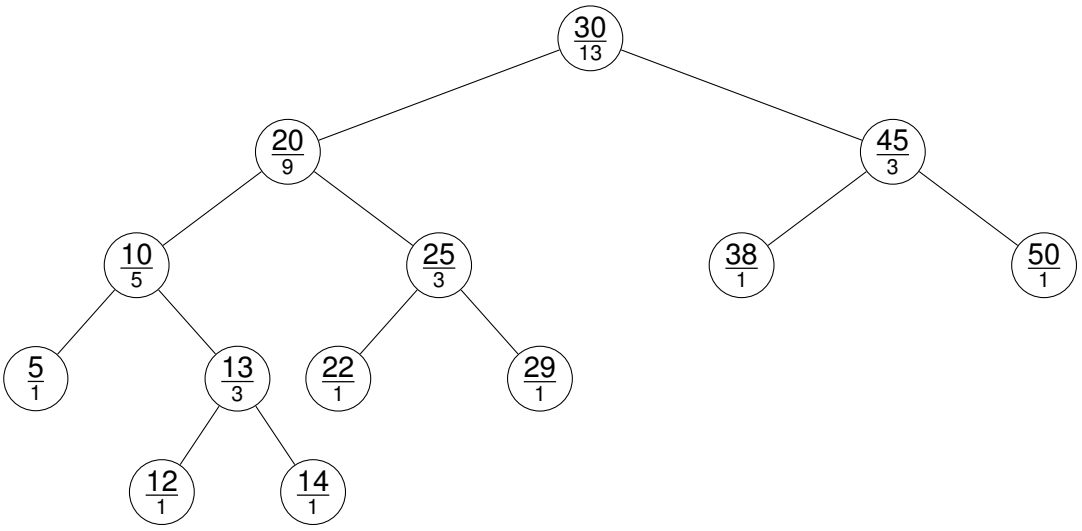
- Sie dürfen weitere Hilfmethoden implementieren, der Testcode wird aber nur die oben genannten Methoden aufrufen und auf Korrektheit testen.

- Sie dürfen selbst entscheiden, ob Sie Ihre Methoden rekursiv oder iterativ implementieren.

- Achten Sie generell auf eine möglichst effiziente Implementierung (bezüglich Laufzeit und zusätzlich erforderlichen Speicherplatzes).

Beispiel

Das folgende Beispiel gibt Ihnen einen Überblick über die Ergebnisse der einzelnen Methoden. Die obere Zahl im Knoten gibt den Schlüssel an, die Untere die Größe des Unterbaumes. Es sei folgender Baum gegeben:

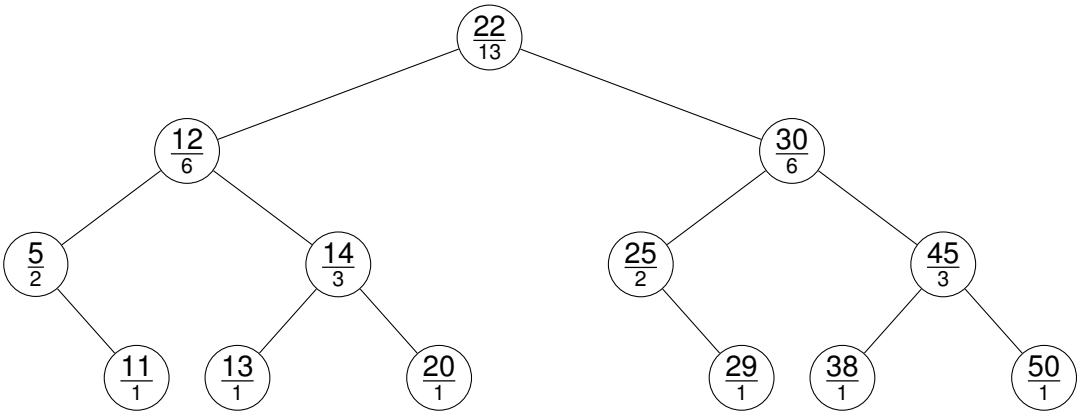


Es ergeben sich dann für die einzelnen Methodenaufrufe die folgenden Rückgabewerte:

- `isEmpty():false`
- `size():13`
- `height():4`
- `exists(20):true`
- `exists(23):false`
- `valueAtPosition(3):13`
- `valueAtPosition(0):5`
- `position(10):1`
- `position(29):8`
- `position(31):10`
- `values(14, 38):14 20 22 25 29 30 38`
- `values(38, 14):5 10 12 13 14 38 45 50`
- `insert(11):`
- `delete(10):`

- `valueAtPosition(1): 11`

Wenn danach `simpleBalance()` aufgerufen wird, dann sieht der Baum beispielsweise so aus:



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 in TUWEL zur Verfügung.

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

Weiters beachten Sie bitte, dass alle Änderungen im Code außerhalb von `Tree.java` nicht von unserem Abgabesystem berücksichtigt werden.

Hinweise
`Tester.printDebug(String msg)` kann verwendet werden um Debuginformationen auszugeben. Die Ausgabe erfolgt nur, wenn beim Aufrufen das Debugflag (`-d`) gesetzt wurde. Weitere Erläuterungen zur Ausgabe finden Sie im Abschnitt Testdaten. Sie müssen diese Ausgaben vor der Abgabe **nicht** entfernen, da das Testsystem beim Kontrollieren Ihrer Implementierung diese Flags 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 Ist Ihre Lösung korrekt gibt das Framework `OK` zurück. Andernfalls bekommen Sie eine Fehlermeldung mit einem Hinweis zur Ursache.

Testdaten

In TUWEL werden 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.

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

Abgabe

Die Abgabe Ihrer Implementierung der Klasse `Tree` 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 (1 Punkt)

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

- Was versteht man unter einem binären Suchbaum?
- Kann der implementierte binäre Suchbaum zu einer linearen Liste degenerieren?
- Wie würden Sie die Funktionalität der Methode `int position(int val)` implementieren, wenn die Werte aus der Eingabedatei nacheinander in einem Array gespeichert werden?
- Welche asymptotische Laufzeit besitzt die Methode `position` im Best- und im Worst-Case in Abhängigkeit der Knotenanzahl?
- Welche asymptotische Laufzeit besitzt die Methode `simpleBalance` im Best- und im Worst-Case in Abhängigkeit der Knotenanzahl?
- Welche asymptotische Laufzeit besitzt die Methode `values` in Θ-Notation in Abhängigkeit der Knotenanzahl?

Abgabesystem

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

```
java ads1.ss15.pa.Tester 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.

Bitte beachten Sie, dass unser Abgabesystem **kein** Debugging-Tool ist!

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:

- Alle Instanzen korrekt gelöst (mit einer korrekten allgemeinen Lösung)
- 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 die AlgoDat1-Hotline unter `algodat1-ss15@ac.tuwien.ac.at` oder posten Sie Ihre Frage im TUWEL-Diskussionsforum.