Informatik II (D-ITET)
Frühjahrssemester 2015
Prof. Friedemann Mattern
Christian Beckel (beckel@inf.ethz.ch)

ETH Zürich
Institut für Pervasive Computing
Gruppe für Verteilte Systeme
http://vs.inf.ethz.ch

Übungsserie Nr. 2

Ausgabe: 4. März 2015
Abgabe: 11. März 2015

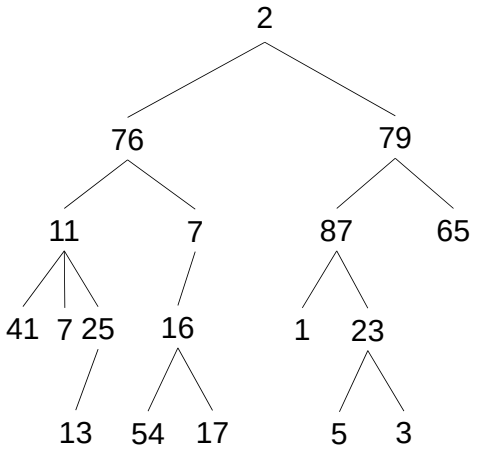
Hinweise

Für diese Serie benötigen Sie das Archiv
`http://vs.inf.ethz.ch/edu/FS2015/I2/downloads/u2.zip`.
Zur Erreichung der vollen Punktzahl bei den Programmieraufgaben ist es notwendig, dass alle mitgelieferten Tests unverändert bestanden werden. Ausserdem muss die Implementierung der jeweiligen Aufgabenstellung entsprechen. Orientieren Sie sich bei Detailfragen an der vorhandenen Dokumentation.

1. Aufgabe: (8 Punkte) Wurzelbäume

In der Vorlesung haben Sie die Klammerdarstellung, sowie die Darstellung in eingerückter Form und den Wurzelbaum als speziellen gerichteten Graphen kennengelernt.

(1a) *(2 Punkte)* Geben Sie zu dem folgenden Baum die zugehörige Klammerdarstellung und Darstellung in eingerückter Form an:



(1b) *(2 Punkte)* Geben Sie zur folgenden Klammerdarstellung den zugehörigen Baum als gerichteten Graphen und in eingerückter Form an:

```
S(R(H(K)),P(A(N,O),Q,T),V(J,F(G)))
```

(1c) *(1 Punkt)* Können Sie aus den obigen Klammerdarstellungen den dazugehörigen Baum eindeutig rekonstruieren? Begründen Sie Ihre Antwort.

(1d) *(3 Punkte)* Beantworten Sie für die Bäume aus Teilaufgaben *a)* und *b)* folgende Fragen:

i) Was ist die Höhe des Baumes? (Ein Baum mit nur *einem* Knoten habe die Höhe 1.)

ii) Welches sind die längsten Pfade in diesem Baum? (*Hinweis*: Bei Wurzelbäumen handelt es sich um gerichtete Graphen.)

iii) Welche Knoten bilden die Blätter dieses Baumes?

2. Aufgabe: (7 Punkte) Sortieren

Vervollständigen Sie die Implementierung der Klasse `u2a2.RandomArray`.

(2a) *(1 Punkt)* Implementieren Sie zuerst den Konstruktor, der ein Array der angegebenen Grösse erstellt und mit Zufallszahlen füllt. Hinweis: Für die Erzeugung von Zufallszahlen kann die Klasse `java.util.Random` verwendet werden.

(2b) *(2 Punkte)* Implementieren Sie die Funktion `toString`, die eine Stringrepräsentation des Arrays erzeugt.

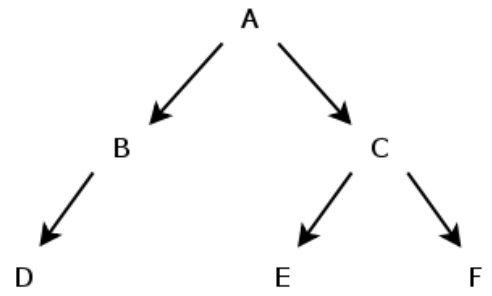
(2c) *(4 Punkte)* Implementieren Sie die Funktion `recursiveSort(int until)`, die das Array rekursiv anhand des folgenden Algorithmus absteigend sortiert:

1. **Rekursionsannahme**: `recursiveSort(until-1)` sortiert die `until-1` grössten Zahlen des Arrays absteigend in die ersten `until-1` Positionen des Arrays ein.
2. **Rekursionsschritt**: Nach der Sortierung der ersten `until-1` Positionen wird die grösste Zahl aus dem Rest des Arrays mit der Zahl an der Stelle `until-1` vertauscht. Dann sind die `until` grössten Zahlen in den ersten `until` Positionen absteigend einsortiert.
3. **Rekursionsabbruch**: Das leere Array ist absteigend sortiert.
4. **Rekursionsanfang**: `recursiveSort(length)` sortiert das gesamte Array absteigend.

3. Aufgabe: (5 Punkte) Binärbäume als Arrays

Binärbäume können in Arrays gespeichert werden, indem man die Werte der Knoten niveauweise von links nach rechts abspeichert. Ein nicht vorhandener Knoten wird dabei durch einen speziellen Wert repräsentiert.

Die Klasse `u2a3.BinaryTree` arbeitet mit Binärbäumen, die als `char`-Array implementiert sind. Die Repräsentation eines nicht vorhandenen Knotens ist dabei das Leerzeichen.




```
char[] tree = {
    'A',
    'B',
    'C',
    'D',
    ' ',
    'E',
    'F'
};
```

Vervollständigen Sie die Implementierung der Klasse `u2a3.BinaryTree`.

(3a) *(1 Punkt)* Implementieren Sie die Funktionen `leftChild`, `rightChild` und `father`, die zu einem Index eines Knotens den Index des linken Kindes, des rechten Kindes bzw. des Vaters berechnen.

(3b) *(2 Punkte)* Implementieren Sie die Funktion `toString`, die den Binärbaum in eingerückter Form zurückgibt. *Hinweis*: Die rekursive Implementierung ist einfacher und eleganter als die iterative Implementierung.

(3c) *(2 Punkte)* Implementieren Sie die Funktion `checkTree`, die das übergebene Array daraufhin überprüft, ob es eine gültige Repräsentation eines Binärbaumes ist. Nicht vorhandene Blätter am Ende des Arrays sollen dabei keine Rolle spielen, das Array

```
['A','B',' ']
```

ist also ebenso ein korrekter Binärbaum wie das Array

```
['A','B'].
```