Prof. Dr. Stefan Göller
Dr. Da-Jung Cho
Daniel Kernberger

Einführung in die Informatik

WS 2019/2020

Übungsblatt 5
21.11.2019 - 28.11.2019

Abgabe: Bis zum 28.11. 18:00 Uhr über moodle. Reichen Sie pro Aufgabe, die Sie bearbeitet haben, genau eine Textdatei mit dem Namen `aufgabe_i.py`, (wobei `i` die Aufgabennummer ist) ein, welche die Lösung ihrer Gruppe enthält. Beachten Sie den speziellen Dateinamen, der in Aufgabe 2 gefordert ist. Verwenden Sie ggf. Vorlagen zu diesen Dateien, die wir Ihnen in Moodle zur Verfügung stellen.

Aufgabe 1 (Umtauschen von Ziffern) (20 Punkte):

Im folgenden sei ein *Zahlenstring* eine Zeichenkette, die nur aus Ziffern (0 bis 9) besteht. Schreiben Sie eine Funktion `maximiere`, die zwei Parameter, nämliche einen Zahlenstring `x` und einen Integer `k` entgegennimmt. Ihre Funktion `maximiere` soll rekursiv den größte Zahlenstring berechnen, der aus `x` entsteht, wenn maximal `k` Mal Zeichen vertauscht werden dürfen. Beispielsweise soll `maximiere(x,k)` auf Eingabe `x="223953"` und `k=2` die Zeichenkette `"953223"` zurückgegeben werden, indem zuerst Position 0 (dort steht eine `"2"`) mit Position 4 vertauscht werden (dort steht eine `"9"`) und dann Position 1 (dort steht eine `"2"`) und Position 5 vertauscht werden (dort steht eine `"5"`). Kopieren Sie folgenden beiden Funktionen in Ihr Programm, studieren Sie ihre Funktionsweise und verwenden Sie sie als Hilfsfunktionen.

```
def vertausche(s,i,j):
    return s[0:i]+s[j]+s[i+1:j]+s[i]+s[j+1:len(s)]

def sortiere(s):
    return "".join((sorted(s)[::-1]))
```

Aufgabe 2 (Fibonacci-Zahlen und der euklidische Algorithmus) (15 Punkte):

Geben Sie Ihre Lösung zu dieser Aufgabe in einer Datei namens `aufgabe_2.txt` ab. Der euklidische Algorithmus (Programm `Euklid.py` aus der Vorlesung) auf Eingabe $a = 13$ und $b = 8$ stellt fest, dass $13 \not\equiv 8 \bmod 3$ gilt, und liefert das Ergebnis des rekursiven Aufrufs $\texttt{Euklid}(8, 13 \bmod 8) = \texttt{Euklid}(8, 5)$ zurück (ein erster rekursiver Aufruf geschieht). Beim Aufruf von $\texttt{Euklid}(8, 5)$ wird ebenfalls festgestellt, dass $8 \not\equiv 5 \bmod 5$ gilt, ein rekursiver Aufruf geschieht $\texttt{Euklid}(5, 8 \bmod 5) = \texttt{Euklid}(5, 3)$. Beim Aufruf von $\texttt{Euklid}(5, 3)$ wiederum wird ebenfalls festgestellt, dass $5 \not\equiv 0 \bmod 3$, ein rekursiver Aufruf $\texttt{Euklid}(3, 5 \bmod 3) = \texttt{Euklid}(3, 2)$ geschieht. Beim Aufruf von $\texttt{Euklid}(3, 2)$ wiederum wird ebenfalls festgestellt, dass $3 \not\equiv 0 \bmod 2$, ein rekursiver Aufruf $\texttt{Euklid}(2, 3 \bmod 2) = \texttt{Euklid}(2, 1)$ geschieht. Beim Aufruf von $\texttt{Euklid}(2, 1)$ geschieht kein weiterer rekursiver Aufruf und es wird $2 \bmod 11 = 1$ zurückgegeben. Die *Rekursionstiefe* des euklidische Algorithmus auf Eingabe $a = 13$ und $b = 8$ beträgt demnach 4: $\texttt{Euklid}(13, 8)$ ruft $\texttt{Euklid}(8, 5)$ auf, was wiederum $\texttt{Euklid}(5, 3)$ aufruft, was wiederum $\texttt{Euklid}(3, 2)$ aufruft, was wiederum $\texttt{Euklid}(2, 1)$ aufruft, bei dem kein rekursiver Aufruf geschieht.

Allgemeiner bezeiche $\mathrm{RT}(a, b)$ die *Rekursionstiefe* des euklidischen Algorithmus auf Eingaben $a > b > 0$, also $\mathrm{RT}(a, b) = 0$ falls $b|a$.

Wir wollen in dieser Aufgabe die Rekursionstiefe des euklidischen Algorithmus auf zwei aufeinanderfolgenden Fibonacci-Zahlen untersuchen.

Zur Erinnerung, die n-te Fibonacci-Zahl $\mathsf{fib}(n)$ ist induktiv wie folgt definiert: $\mathsf{fib}(0) = \mathsf{fib}(1) = 1$ und $\mathsf{fib}(n) = \mathsf{fib}(n-1) + \mathsf{fib}(n-2)$ für alle $n > 1$. Zeigen.

Beweisen Sie durch vollständige Induktion dass für alle $n \geq 1$ gilt,

$$\mathrm{RT}(\mathsf{fib}(n+1), \mathsf{fib}(n)) = n - 1.$$

Wichtige Hinweise: Begründen Sie im Induktionsschritt jeden einzelnen Schritt und geben Sie explizit an wann Sie die Induktionsvoraussetzung verwenden.

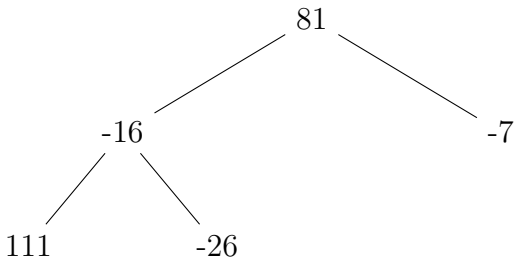
Aufgabe 3 (Baumtraversierung) (25 Punkte):

Ein Baum ist ein in der Informatik häufig genutztes Konzept um strukturierte Daten zu repräsentieren. In dieser Aufgabe behandeln wir sogenannte Integer-Bäume, die wie folgt induktiv mit Hilfe von Tupeln definiert sind:

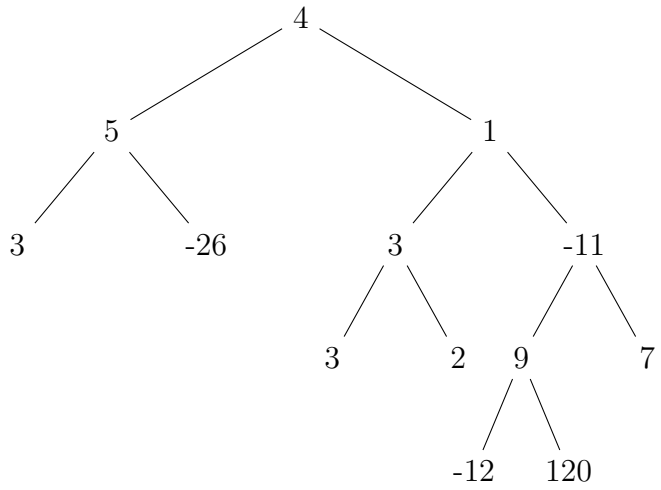
1. Jeder Integer ist ein Integer-Baum.
2. Wenn b_1 und b_2 Integer-Bäume sind und `n` ein Integer, dann ist auch das Tupel `(`b_1`,n,`b_2`)` ein Integer-Baum.

Beispiel:

- `((111,-16,-26),81,-7)` ist ein Integer-Baum. Grafisch kann man sich diesen wie folgt vorstellen:



- `((3,5,-26),4,((3,3,2),1,((-12,9,120),-11,7)))` ist ein Integer-Baum. Grafisch kann man sich diesen wie folgt vorstellen:



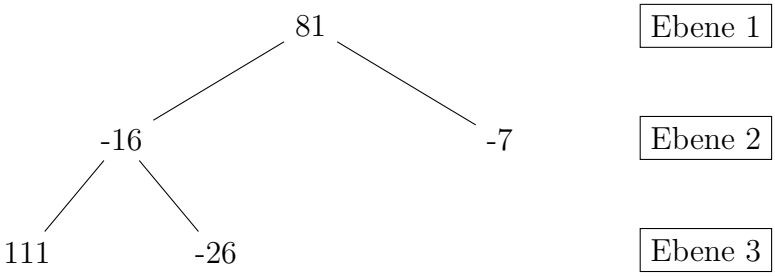
a) Schreiben Sie eine Funktion `summiere`, die die Zahlen in einem gegebenen Integerbaum, multipliziert mit der jeweiligen Ebenennummer, aufsummiert. Sie sollen dieses Problem mittels Rekursion lösen.

Beispiel:

- Für den Integer-Baum

  `((111,-16,-26),81,-7))`

  bildlich



  soll die Funktion also die Summe

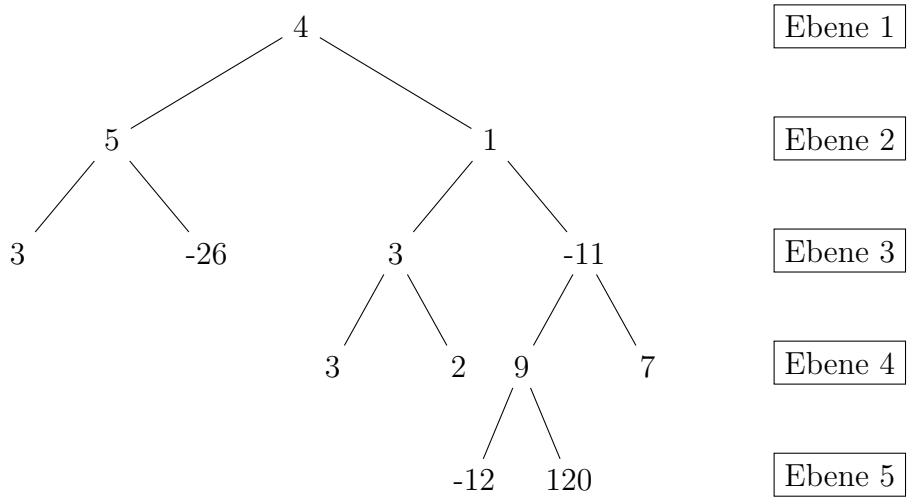
$$1 \cdot 81 + 2 \cdot (-16 - 7) + 3 \cdot (111 - 26) = 290$$

  berechnen.

- Für den Integer-Baum

  `((3,5,-26),4,((3,3,2),1,((-12,9,120),-11,7)))`

  bildlich



  soll die Funktion also die Summe

$$1 \cdot 4 + 2 \cdot (5 + 1) + 3 \cdot (3 - 26 + 3 - 11) + 4 \cdot (3 + 2 + 9 + 7) + 5 \cdot (-12 + 120) = 547$$

  berechnen.

Hinweis: Es ist sinnvoll, eine weitere Hilfsfunktion zu schreiben, der sowohl der Baum als auch die Ebene übergeben werden. Diese Funktion soll rekursiv vorgehen. Um Ihre Funktion zu testen, legen Sie sich selber in einer Variablen Integer-Bäume, wie oben definiert, an und übergeben Sie diese Ihrer Funktion.

b) **(Sternchenaufgabe)** Schreiben Sie eine Funktion welche eine Zeichenkette bestehend aus `"("` , `")"` , `","` , `"-"` und Zahlen einliest. Falls es sich bei der eingelesenen Zeichenkette um die Kodierung eines Integer-Baums handelt, soll der entsprechende Integer-Baum zurückgegeben werden und falls nicht, soll `None` zurückgegeben werden.