Algorithmen und Programmieren III, WS 1999/2000 — 5. Übungsblatt

Abgabe bis Dienstag, 7. Dezember 1999

24. (3 Punkte) Schreiben Sie Funktionen vom Typ `Int->T` in Haskell, die für gegebenes n (a) einen möglichst ausgeglichenen Baum, (b) einen nach links entarteten Baum, (c) einen nach rechts entarteten Baum, beziehungsweise (d) einen entarteten Zickzackbaum mit n Knoten erzeugen. Bei (a) sollen die beiden Unterbäume jedes Knotens möglichst gleich viele Knoten enthalten. Bei (b) hat kein Knoten ein rechtes Kind; der Baum besteht aus einem einzigen Pfad, der immer nach links führt, und analog für (c). Bei (d) gibt es ebenfalls auf jeder Ebene nur einen Knoten, aber der Weg von der Wurzel zum einzigen Blatt führt abwechselnd nach links und nach rechts. (Die Werte in den Knoten können beliebig sein.)

25. (11 Punkte) Laufzeitverhalten von Funktionen in Haskell. Betrachten Sie folgende 2 Funktionen `sR1` und `sR2`[1] zum Durchlaufen eines Baumes in symmetrischer Reihenfolge und zur Ausgabe aller Knoten in einer Liste:

```
sR1:: T -> [Int]
sR1 Nil = []
sR1 (Baum l w r) =
    (sR1 l) ++ [w] ++ (sR1 r)

len:: T -> Int
len Nil = 0
len (Baum l _ r) = 1+len l+len r
```

```
data T = Nil | Baum{l::T, wert::Int, r::T}
    deriving (Show,Eq)

sR2:: T -> [Int]
sR2 b = sR' b [] where
  sR':: T -> [Int] -> [Int]
  sR' Nil x = x
  sR' (Baum l w r) x = sR' l (w:(sR' r x))
```

(a) (0 Punkte) Zeigen Sie, dass `(sR' b x) ++ y = sR' b (x++y)` ist.

(b) (11 Punkte) Vergleichen Sie die Laufzeit von `sR1` und `sR2` für die Bäume (a)–(c) aus der vorigen Aufgabe, für jeweils $n = 2^i$ Knoten, $i = 3, \ldots, 11$. Wenn man im HUGS-System `:set +s` eingibt, wird nach jeder Eingabe die Anzahl der *Reduktionen* (und auch die Anzahl der verwendeten Speicherzellen) ausgegeben, die man als Maß für die Laufzeit nehmen kann. Um von der Erzeugung des Baumes und von der Zeit für die Ausgabe der Liste abzusehen, vergleichen Sie für jeden Baum `b` jeweils folgende Eingaben:
(1) `len b`, als Maß für das einfachste Durchlaufen des Baumes,
(2) `length (sR1 b)`, sowie
(3) `length (sR2 b)`.
Bilden sie jeweils die Differenz der Reduktionenzahl zwischen (2) und (1), und zwischen (3) und (2). Schätzen Sie sodann experimentell einen Wert für die Konstante c unter der Annahme, dass sich die gemessenen Laufzeitdifferenzen nach der Formel $c \cdot n$, $c \cdot n \log_2 n$, beziehungsweise $c \cdot n^2$ verhalten, sodass die Formel möglichst gut mit Ihren Daten übereinstimmt. Welche Formel halten Sie jeweils für die passende?

(c) (Zusatzfrage, 3 Punkte) Erklären Sie die Ergebnisse, die Sie erhalten haben.

26. (7 Punkte) Nehmen wir an, dass jeder Knoten eines Baumes auch einen Zeiger zur Mutter hat:

```
class Baum2 extends Baum
{ Baum2 m;
  ...
```

Der Mutterzeiger der Wurzel ist `null`. Schreiben Sie ein nichtrekursives Programm, das die Knoten des Baumes in symmetrischer Reihenfolge ausgibt.

[1] siehe `http://www.inf.fu-berlin.de/~rote/Lere/alp3/symmetrischeReihenfolge.hs`

27. (9 Punkte) Eine *Triangulierung* eines konvexen n-Ecks P_n ist eine Zerlegung in $n-2$ Dreiecke durch $n-3$ nichtkreuzende Diagonalen, die je zwei nichtbenarchbarte Ecken verbinden.

   (a) (0 Punkte) Zeigen Sie, dass es 5 verschiedene Triangulierungen des Fünfecks P_5 gibt. (Die Ecken des P_5 sind nummeriert.) Zeigen Sie, dass es 5 verschiedene binäre Bäume mit 3 Knoten gibt. Zeigen Sie, dass es genauso viele Triangulierungen des P_n wie binäre Bäume mit $n-2$ Knoten gibt.

   (b) (4 Punkte) Eine Möglichkeit, eine Triangulierung des P_n anzugeben, ist die *Gradliste* $(a_1, \ldots, a_n)$, wobei a_i die Anzahl der Diagonalen ist, die in der i-ten Ecke zusammenkommen. So hat zum Beispiel jede Triangulierung des P_5 die Gradliste $(1, 0, 2, 0, 1)$, bis auf zyklische Verschiebungen.

   Zeigen Sie, dass es immer eine Ecke vom Grad $a_i = 0$ geben muss. (Was entspricht einer solchen Ecke geometrisch? Was passiert, wenn man eine solche Ecke entfernt?)

   (c) (0 Punkte) Zeigen Sie, dass eine Triangulierung durch ihre Gradliste eindeutig bestimmt ist.

   (d) (5 Punkte) Schreiben Sie ein Programm, das testet, ob eine gegeben Folge $(a_1, \ldots, a_n)$ die Gradliste einer Triangulierung ist.

28. (4 Punkte) Zufällige Permutationen. Wie kann man eine gegebene Folge $a_1, \ldots, a_n$ von Zahlen in eine zufällige Reihenfolge bringen? Schätzen Sie die Laufzeit und den Speicherplatz Ihres Algorithmus ab. Wie viele Zufallszahlen benötigen Sie? (Für einen Linearzeitalgorithmus gibt es einen Zusatzpunkt.)

29. (-2 Punkte) Ändern Sie die Warteschlangensimulation aus Aufgabe 18(a) wie folgt.

   Bestimmen Sie die durchschnittliche Länge ℓ der Warteschlange im Zeitraum zwischen 12:00 und 13:00 Uhr, sowie den Zeitpunkt t, wann der letzte Kunde fertig ist. Führen Sie 100 unabhängige Simulationsläufe durch, und bestimmen Sie dem maximalen, den minimalen, und den Mittelwert von ℓ und t.

30. (0 Punkte) Verschmelzen. Addition stückweise linearer Funktionen.
   Eine stückweise lineare Funktionen $f\colon [u, v] \to \mathbb{R}$ ist durch eine zusammenhängende Folge von Intervallen mit entsprechenden linearen Stücken $f(x) = ax + b$ gegeben, zum Beispiel:

$$f(x) = \begin{cases} x+1, & \text{für } -1 < x \le 0 \\ -1, & \text{für } 0 < x < 2 \\ 6-2x, & \text{für } 2 \le x \le 4 \end{cases}$$

   (a) Überlegen Sie sich eine Datenstruktur (Haskell-Datentyp/Java-Klasse) zur Darstellung solcher Funktionen. Schreiben Sie eine Funktion/Methode, die den Wert einer solche Funktion $f(x)$ an einer gegebenen Stelle x ausrechnet.

   (b) Schreiben Sie eine Funktion zur Addition zweier stückweise linearer Funktionen. Der Definitionsbereich von $f + g$ soll dabei der Durchschnitt der Definitionsbereiche von f und g sein.

   (c) Schreiben Sie eine Funktion/Methode zur Integration einer stückweise *konstanten* Funktion $f\colon [u, v] \to \mathbb{R}$. Das heißt, es soll die stückweise lineare und stetige Stammfunktion F von f berechnet werden:

$$F\colon [u, v] \to \mathbb{R}, \qquad F(x) = \int_u^x f(x)\, dx$$