Fakultät IV – Elektrotechnik und Informatik
MPGI 1 WS 2013/14
Pepper / Höger / Reicherdt / Zuber / Tutoren

Aufgabenblatt 14

letzte Aktualisierung: 30. 01, 11:02

Ausgabe: 29.01.2014
Abgabe: 06/07.02.2014

Thema: *Aufwand, Sortieralgorithmen*

Die Vorstellung der Projektaufgabe findet in den Tutorien am 10./11.02.2014 statt.

In jeder Gruppe müssen alle Mitglieder ihren Anteil an der Aufgabenlösung vorstellen können.

Achtung: Bei diesem Aufgabenblatt ist nichts zu programmieren, aber trotzdem einiges aufzuschreiben. Gebt eure Lösungen bitte als PDF ab. Dieses PDF ist idealerweise mit LaTeX generiert. [1] Im Notfall akzeptieren wir auch simple Textdateien (`*.txt`) oder Scans von handschriftlichen Lösungen.

1. Aufgabe: Aufwandsabschätzung für rekursive Funktionen

Aussagen wie $A \in \mathcal{O}(1)$ kann man lesen als: Die Funktion A wächst für gegen unendlich strebende Eingabewerte bis auf einen konstanten Faktor maximal so stark wie $\lambda n.1$.
Die folgende Tabelle ermöglicht, anhand der Rekursionsstruktur einer Aufwandsabschätzungsfunktion $A : \mathbb{N} \to \mathbb{R}$ die Klasse $\mathcal{O}(A)$ der höchstens genau so stark wachsenden Aufwandsfunktionen anzugeben.

	Rekurrenzgleichung		$A \in ...$
1.	$A(n) = A(n-1) + bn^k$		$\mathcal{O}(n^{k+1})$
2.	$A(n) = cA(n-1) + bn^k$	mit $c > 1$	$\mathcal{O}(c^n)$
3.	$A(n) = cA(n/d) + bn^k$	mit $c > d^k$	$\mathcal{O}(n^{\log_d c})$
4.	$A(n) = cA(n/d) + bn^k$	mit $c < d^k$	$\mathcal{O}(n^k)$
5.	$A(n) = cA(n/d) + bn^k$	mit $c = d^k$	$\mathcal{O}(n^k \log n)$

1.1. (Tut) Fakultätsfunktion Bestimmt die Klasse der Aufwandsabschätzung $A_{\texttt{fac}}$ zur Funktion `fac` in Abhängigkeit von der Größe der Eingabe n. Gebt den Aufwandsterm, die Herleitung der passenden Rekurrenzgleichung sowie die eingesetzten Koeffizienten an!

```
FUN fac : nat → nat
DEF fac(n) ==
  IF n = 0 THEN 1
  ELSE n * fac(n-1)
  FI
```

[1] Wenn ihr dabei den Komfort eines normalen Dokumentenverarbeitungsprogramms genießen wollt, sei euch **LyX** ans Herz gelegt, `http://www.lyx.org/`.

1.2. (Tut) Fibonacci Wie verhält sich der Aufwand der Funktion `fib`? Geht vor wie in der vorigen Aufgabe!

```
FUN fib : nat → nat
DEF fib(0)        == 0
DEF fib(succ(0)) == 1
DEF fib(n)        == fib(n-1) + fib(n-2)
```

1.3. (Pflicht-Hausaufgabe) Potenzieren Bestimmt eine möglichst kleine Klasse der Aufwandsabschätzung $A_{\texttt{exp}}$ zur Funktion `exp` in Abhängigkeit von der Größe der Eingabe n. (Die erste Eingabe, x, sollte sich nicht auf den Aufwand der Funktion auswirken.)

Entwickelt (1) den Aufwandsterm, leitet (2) die passende Rekurrenzgleichung inklusive eingesetzter Koeffizienten her und benennt (3) die ermittelte Aufwandsklasse!

```
FUN exp : real ** nat → real
DEF exp(x, n) ==
  IF n = 0 THEN 1
  IF n = 1 THEN x
  IF even?(n) THEN
    LET s == exp(x, n/2) IN
    s*s
  IF odd?(n) THEN
    LET s == exp(x, (n-1)/2) IN
    x * (s*s)
  FI
```

Hinweis: $A_{\texttt{even?}}$ und $A_{\texttt{odd?}}$ sind konstant.

2. Aufgabe: Aufwand Selectionsort

Beispielcode für Selectionsort:

```
FUN ssort: seq[nat] → seq[nat]
DEF ssort(<>) == <>
DEF ssort(a::A)  ==
  LET minimum  == reduce(min,a)(a::A)
      (rL, rR) == split(_ |= minimum, a::A)
  IN  minimum :: ssort(rL ++ rt(rR))
```

Hinweis: Die Funktion `min'Nat : nat ** nat -> nat` gibt von zwei Zahlen die kleinere zurück und hat konstanten Aufwand.

2.1. (Tut) Handsimulation Wie wertet der Algorithmus die Funktion `ssort(<7,3,5>)` aus?

2.2. (Tut) Aufwand Bestimmt die Aufwandsklassen für Selectionsort sowohl für den Best- als auch den Worst-Case. Gebt für `ssort` den Aufwandsterm, die Herleitung der Rekurrenzgleichung sowie deren Koeffizienten an.

Unter welchen Annahmen tritt der Best- bzw. der Worst-Case auf?

3. Aufgabe: Aufwand Quicksort

Beispielcode für Quicksort:

```
FUN qsort: seq[nat] → seq[nat]
DEF qsort(<>) == <>
DEF qsort(a::A) ==
  LET (lesser, greaterEq) == partition(_ < a, A)
  IN  qsort(lesser) ++ a :: qsort(greaterEq)
```

3.1. (Tut) Handsimulation Wie wertet der Algorithmus die Funktion `qsort(<7,3,8,5>)` aus?

3.2. (Pflicht-Hausaufgabe) Aufwand Bestimmt die Aufwandsklassen für Quicksort sowohl für den Best- als auch den Worst-Case. Gebt den Aufwandsterm, die Herleitung der Rekurrenzgleichung sowie ihre Koeffizienten an.

Unter welchen Annahmen tritt der Best- bzw. der Worst-Case auf?

3.3. (Freiwillige Hausaufgabe) Besserer Best-Case Betrachtet folgende Quicksort-Implementierung. Sei c eine beliebige feste natürliche Zahl. Welche Aufwandsklasse ergibt sich dann für Eingabe-Listen beliebiger Länge, deren Elemente maximal c unterschiedliche Werte haben?

```
FUN qsort: seq[nat] → seq[nat]
DEF qsort(<>) == <>
DEF qsort(a::A) ==
  LET lesser  == filter(_ < a)(A)
      equal   == a :: filter(_ = a)(A)
      greater == filter(_ > a)(A)
  IN  qsort(lesser) ++ equal ++ qsort(greater)
```