Algorithmen und Programmierung 3, WS 2003/2004 — 1. Übungsblatt

Abgabe bis Donnerstag, 30. Oktober 2003

1. (i) (4 Punkte) Sortieren Sie die folgenden Laufzeiten aufsteigend. (Wenn $f(n) = O(g(n))$ ist, aber nicht $g(n) = O(f(n))$, dann soll $f(n)$ vor $g(n)$ kommen.)

   (a) n^3 (b) $\log_2 n$ (c) $1{,}8^n$ (d) n (e) 3^n (f) $\sqrt{n}$ (g) $n(\log_2 n)^2$ (h) n^2

   (ii) (4 Punkte) Wenn ein Programm die Laufzeit $f(n)$ aus Aufgabe (i) hat und die Computer in zehn Jahren um den Faktor 1000 schneller werden, dann kann man Probleme *welcher Größe* in derselben Zeit lösen, in der man heute Probleme der Größe (1) $n = 20$, (2) $n = 1000$ lösen kann?

2. (2 Punkte) Die Computer werden immer schneller, die Speicherelemente immer billiger. Wird der Entwurf von effizienten Algorithmen angesichts dieser Entwicklung in Zukunft an Bedeutung verlieren? Bei welchen Computeranwendungen werden Effizienzfragen eine größere/kleinere Rolle spielen?

   Diskutieren Sie diese Fragen. (mindestens 5–10 Zeilen)

3. (0 Punkte) Das folgende Programmstück ist ein Versuch, Sortieren durch Einfügen zu implementieren.

```
    void insertionSort(int[] a)
    { int x;
      for (int i = 0; i<a.length; i++)
      { † x = a[i];
        // Sortiere x zwischen a[0..i-1] ein
        // Bestimme zunächst die richtige Position j:
(*)     int j=i-1; † while (j>=0 && a[j-1]>x) j-- † ;
        // Nun verschiebe einen Teil des Feldes und füge x an dieser Stelle ein:
      † for (int k = i-1; k>=j; k--) a[k+1]=a[k];
        a[j]=x; †
      }
    }
```

   Stellen Sie dieses Programm richtig und fügen Sie an den durch † bezeichneten Stellen in Ihrem Programm Schleifeninvarianten in der Form vom Zusicherungen (assertions) ein, aus denen die Korrektheit des Programmes hervorgeht. (Ein formaler Beweis ist nicht erforderlich, aber die Zusicherungen müssen aussagekräftig sein und vor allem zutreffen.)

4. (0 Punkte) Kann man Sortieren durch Einfügen schneller machen, indem man die Einfügestelle `j` schneller findet als oben in Zeile $(*)$, zum Beispiel durch binäres Suchen? Wie ist es im besten und im schlechtesten Fall?

5. (0 Punkte) Zeigen Sie:

   (a) Wenn $f(n) = a_d n^d + a_{d-1} n^{d-1} + \cdots + a_1 n + a_0$ ein Polynom vom Grad d mit positivem Leitkoeffizienten ($a_d > 0$) ist, dann ist

   $$f(n) = \Theta(n^d).$$

   (b) $\log_a n = \Theta(\log_b n)$ für $a, b > 1$.

   (c) Unter welchen Bedingungen folgt $g(n) = \Omega(f(n))$ aus $f(n) = O(g(n))$?

6. (0 Punkte) Welche der beiden folgenden Laufzeiten $f(n)$ und $g(n)$ ist für große Werte von n schneller, und welche für kleine n? Bei welchem Wert von n ändert sich die Antwort?

   (a) $f(n) = 10n(\log_2 n)^2$, $g(n) = 2n^{3/2}$

   (b) $f(n) = 5 \cdot 2^n$, $g(n) = 100n^2 \log_2 n$

7. (0 Punkte)

   (a) Beweisen Sie: Wenn $f(n) = O(g(n))$ ist, dann ist $f(n) + g(n) = O(g(n))$.

   (b) Beweisen Sie: $\max\{f(n), g(n)\} = \Theta(f(n) + g(n))$ für $f(n), g(n) > 0$.

   (c) Welche der folgenden Aussagen sind richtig (Begründen Sie Ihre Antworten):
   $n\sqrt{n} = O(n(\log n)^2)$; $n(\log n)^2 = O(n\sqrt{n})$; $n \log n \cdot (\log \log n) = O(n^2)$;
   $n^2 = O(n \log n \cdot (\log \log n))$; $n^2 \cdot 2^n = O(2^{n+2})$; $n^2 \cdot 2^n = O(3^n)$; $3^n = O(n^2 \cdot 2^n)$;

8. (4 Punkte) Finden Sie möglichst einfache Ausdrücke der Form $\Theta(\cdot)$ für folgende Funktionen:
   (a) $3n^2 - 4n + 32 + 27\,n \cdot \lceil \log_2 n \rceil / 2$; (b) $\max\{n\lceil \log_2 n \rceil, (\lceil \log_2 n \rceil)^4\}$; (c) $2^{2n+\lceil \log_2 n \rceil}$

9. (6 Punkte) Formulieren Sie *Sortieren durch Auswahl* in Java (als Methode wie in Aufgabe 3) oder in Haskell (als lauffähige Funktion).