Algorithmen und Programmierung 2, SS 2016 — Probeklausur

Abgabe bis Mittwoch, 8. Juni 2016, 13:45 Uhr

---

42. Rekursion, 10 Punkte

Die folgende Funktion berechnet die Anzahl der Binärziffern (ohne führende Nullen) einer natürlichen Zahl n. (Beispiel: $n = 23 = (10111)_2 \mapsto 5$)

```
def binaryDigits(n):
    if n==0:
        return 0
    else:
        return 1 + binaryDigits(n//2)
```

(a) Schreiben Sie für diese Aufgabe eine Funktion ohne Rekursion.

(b) Schreiben Sie für diese Aufgabe eine Funktion ohne Division (und ohne irgendwelche Bibliotheksfunktionen).
(Sie können Teil (a) und (b) auch gemeinsam durch *eine* Funktion lösen.)

(c) Geben Sie die Laufzeit und den Speicherbedarf für die gegebene Funktion und für Ihre Lösungen an. (Nehmen Sie dabei an, dass jede der vier Grundrechnungsarten ($+, -, *$ und //) mit ganzen Zahlen in $O(1)$ Zeit durchgeführt werden kann.)

43. O-Notation, 10 Punkte

(a) Gegeben sind zwei positive Funktionen $S, T\colon \mathbb{N} \to \mathbb{R}_{>0}$. Beweisen Sie: Aus $S(n) = \Omega(f(n))$ und $T(n) = \Omega(g(n))$ folgt, dass $S(n) \cdot T(n) = \Omega(f(n)g(n))$ ist.

(b) Geben Sie einen möglichst einfachen Ausdruck der Form $\Theta(f(n))$ für folgenden Ausdruck an:

$$25n^2 - 100\lfloor n/2 \rfloor + 365$$

(c) Geben Sie einen möglichst einfachen Ausdruck der Form $\Theta(f(n))$ für folgenden Ausdruck an:

$$23n + 12n \log_2 n + 1666$$

3. siehe Rückseite

44. Quicksort, 10 Punkte

Das folgende PYTHON-Programm ordnet einen Abschnitt $a[l : r]$ einer Liste a um und zerlegt ihn in drei Teile $a[l : i-1]$, $a[i]$, und $a[i+1 : r]$, sodass alle Elemente im ersten Teil $\leq a[i]$ sind und alle Elemente im dritten Teil $\geq a[i]$ sind.

```
def zerlege(a,l,r):
    """Zerlege a[l:r]=a[l],a[l+1],...,a[r-1] für Quicksort

    Voraussetzung 1: l<r
    Voraussetzung 2: a[r] existiert, und a[r] >= a[i] für l<=i<r.
    Rückgabewert = i = Position des Pivotelements
    """
    pivot = a[l] # a[l] wird als Pivotelement gewählt.
    k, g = l+1, r-1
    while True:

        ------------------------------------------------------- (1.)

        ------------------------------------------------------------
        while a[k]<pivot:
            k = k+1
        while a[g]>pivot:
            g = g-1

        ------------------------------------------------------- (2.)

        ------------------------------------------------------------
        if g<=k:
            a[l],a[g]=a[g],pivot
            return g
        a[k],a[g] = a[g],a[k] # vertausche

        ------------------------------------------------------- (3.)

        ------------------------------------------------------------
        k = k+1
        g = g-1
```

(a) Fügen Sie an den nummerierten Stellen aussagekräftige Invarianten ein, aus denen die Korrektheit des Programmes hervorgeht. Sie können dabei mathematische Notation verwenden und brauchen keine PYTHON-Syntax zu beachten. Zum Beispiel könnte die erste Invariante ungefähr so aussehen:

$$(1.) \quad \mathit{pivot} = a[l] \ \wedge \ k > l \ \wedge \ \forall i\colon (\dots) \Rightarrow a[i] \leq \mathit{pivot} \ \wedge \ \dots$$

(b) Beweisen Sie, dass das Programm auf keine Listenelemente außer auf

$$a[l], a[l+1], \dots, a[r]$$

zugreift. Sie können dabei auf Ihre Invarianten von Teil (a) zurückgreifen. (Die Gültigkeit dieser Invarianten brauchen Sie nicht zu beweisen.)

(c) (3 Zusatzpunkte) Beweisen Sie, dass das Programm terminiert.