Datenstrukturen (DS)

Sommersemester 2015

Dipl-Inf. Bert Besser
Prof. Dr. Georg Schnitger
Hannes Seiwert, M.Sc.

GOETHE UNIVERSITÄT
FRANKFURT AM MAIN

Institut für Informatik
AG Theoretische Informatik

Übung 1

Ausgabe: 21.04.2015
Abgabe: 05.05.2015

- Bitte schreiben Sie Ihren Namen, Matrikelnummer, den Namen Ihres Tutors sowie Ihre Gruppennummer **gut lesbar** auf Ihre Abgaben.
- Bitte **tackern** Sie Ihre abgegebenen Blätter zusammen.
- Alle Antworten sind zu begründen, außer der Aufgabentext erlaubt eine Begründung entfallen zu lassen.
- Bitte beachten Sie: Bonuspunkte werden nur dann auf die Klausur angerechnet, wenn Sie mindestens eine Lösung zu einer Aufgabe in Ihrem Tutorium präsentieren.

Viel Spaß!

Aufgabe 1.1. *Eine Wachstumshierarchie* (6)

(Vorlesung am 21. April - Kapitel 3.1, 3.2.)

Geben Sie die folgenden Funktionen exakt oder in Θ-Notation an.

a) Wenn jede von n Personen die Hände aller anderen Personen genau einmal schüttelt, wie oft werden dann Hände geschüttelt?
b) Wie viele Runden finden in einem K.o.-Turnier mit n Teams statt (vergleiche Präsenzblatt)?
c) Wie viele verschiedene Ranglisten von n Teams gibt es, wenn keine zwei Teams denselben Rang haben dürfen?
d) Wir falten ein rechteckiges Stück Papier n mal. Jedes Mal falten wir an der Mittellinie der jeweils längeren Seite so, dass sich wiederum ein Rechteck ergibt. Wie viele Lagen Papier sind entstanden?
e) Wie viele Additionen werden bei der Multiplikation zweier $n \times n$-Matrizen ausgeführt, wenn wir die Schulmethode anwenden?

Ordnen Sie die Funktionen aus den vorangegangenen Aufgabenteilen aufsteigend nach ihrem asymptotischen Wachstum. Eine Begründung ist nicht notwendig.

Bitte wenden!

Aufgabe 1.2. *Pseudocode und Asymptotik* (6+3+3)

(Für Teil a)v. und Teil b) Vorlesung am 28.04 - Kapitel 3.4, sonst Vorlesung am 21. April - Kapitel 3.1, 3.2., Teil c) ist sofort bearbeitbar.)

a) Bestimmen Sie, wie oft die folgenden (Pseudo-)Codefragmente das Wort „Hallo“ drucken. Eine Bestimmung der Anzahl in Θ-Notation genügt. Geben Sie für Aufgabenteil v. auch die Rekursionsgleichung für die Anzahl $T(n)$ der gedruckten „Hallos“ beim Aufruf von function(n) an.

i. für $i = 1 \dots n$
    für $j = i + 1 \dots n$
        print “Hallo”

ii. s=1
    solange $s < n$
        $s = 2s$
        print “Hallo”

iii. $l = 0, r = n$
    solange $l < r$
        setze $l = \lceil (r + l)/2 \rceil$
        print “Hallo”

iv. $t = 1$
    solange $t < n$
        $t = 2t$
        rufe den Algorithmus aus ii. auf

v. def function(n):
        wenn $n > 1$:
            print “Hallo”
            function($\lfloor n/2 \rfloor$)
    function(n) // Programm starten

vi. $i = 0$, $k = 0$
    solange $i < n$:
        $k = k + 1$
        $i = i + k$
        print “Hallo”

b) Euklids Algorithmus zur Bestimmung des größten gemeinsamen Teilers zweier natürlicher Zahlen $0 \leq a \leq b$ ist:

ggT(a, b): Wenn $a = 0$, dann gebe b aus, sonst gebe ggT$(b \mod a, a)$ aus.

Bestimmen Sie in $\mathcal{O}$-Notation wie viele Aufrufe der Funktion ggT es in Abhängigkeit von n=a+b gibt.

Hinweis: Sie dürfen annehmen, dass $a + (b \mod a) \leq \frac{2}{3} \cdot (a + b)$ gilt. $b \mod a$ ist der Rest nach Division von b durch a.

c) Gegeben sei das Array $A[0] \dots A[n-1]$, in dem jede Zelle eine Zahl aus der Menge $M = \{0, 1, \dots, k\}$ enthält. Wir sortieren A mit folgendem Algorithmus, der zählt wie oft jede Zahl aus M in A vorkommt und anschließend das Array A neu beschreibt:

Erstelle das Array $B[0], \dots, B[k]$
für $i = 0 \dots k$:
    setze $B[i] = 0$ // initialisiere den i-ten Zähler
für $i = 0 \dots n - 1$:
    inkrementiere den Zähler $B[A[i]]$
$i = 0$
für $j = 0 \dots k$: // überschreibe A
    für $z = 1, \dots, B[j]$:
        setze $A[i] = j$
        $i = i + 1$

Wie oft wird ein Zähler im Array B mit Null initialisiert? Wie oft wird ein Zähler im Array B inkrementiert? Wie oft wird eine Zelle von A neu beschrieben? Sie brauchen Ihre Antworten nicht begründen.

Aufgabe 1.3. *Mischen und Sortieren* (4+5)

(Für Teil (a) Vorlesung am 21.04 - Kapitel 3.1, 3.2, für Teil (b) Vorlesung am 28.04 - Kapitel 3.4.)

a) Gegeben seien zwei aufsteigend sortierte Arrays A_l und A_r von ganzen Zahlen. Wir wollen A_l und A_r `mischen`, also alle Zahlen aus A_l und A_r in aufsteigender Reihenfolge in einem neuen Array aufführen. Beschreiben Sie eine Funktion

$$\texttt{mische}(A_l; A_r)$$

in Pseudocode, und analysieren Sie ihre Anzahl benötigter Vergleiche (zwischen je zwei Zahlen) in Θ-Notation. Ein Beispiel:

$$A_l = [1,3,4,6] \text{ und } A_r = [2,4,5,8,9] : \quad \texttt{mische}(A_l; A_r) = [1,2,3,4,4,5,6,8,9]\,.$$

Hinweis: Eine lineare Anzahl $\Theta(|A_l| + |A_r|)$ von Vergleichen ist möglich. Dabei steht $|A_l|$ bzw. $|A_r|$ für die jeweilige Array-Länge, d.h. die Anzahl der jeweils gespeicherten Zahlen.

b) Gegeben ist ein unsortiertes Array $C[0], \ldots, C[n-1]$ von ganzen Zahlen. Wir wollen C in aufsteigender Reihenfolge sortieren. Dazu verwenden wir den folgenden rekursiven Algorithmus mit dem Aufruf Algo(0, $n-1$):

```
function Algo(links, rechts) {      // das Teilarray C[links], ..., C[rechts] wird sortiert
  falls rechts − links == 0: return      // ein Teilarray der Länge eins ist schon sortiert
  mitte = ⌊(rechts + links)/2⌋                                          // die „Mitte“
  Algo(links, mitte)                                  // sortiere die linke Hälfte rekursiv
  Algo(mitte + 1, rechts)                             // sortiere die rechte Hälfte rekursiv
  B = mische(C[links], ..., C[mitte]; C[mitte + 1], ..., C[rechts])   // Hälften mischen
  überschreibe C[links], ..., C[rechts] mit dem Inhalt von B
}
```

i. Geben Sie den Rekursionsbaum für den Aufruf Algo(0, 7) an.

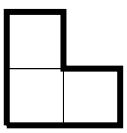
ii. Sei $T(n)$ die Anzahl der im Aufruf Algo(0, $n-1$) benötigten Vergleiche (zwischen je zwei Zahlen). Geben Sie eine Rekursionsgleichung für $T(n)$ an.

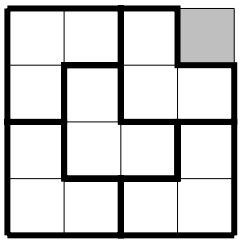
iii. Lösen Sie Ihre Rekursionsgleichung für $T(n)$ z.B. mit Hilfe des Master-Theorems.

Aufgabe 1.4. *Induktion* (5)

(Sofort bearbeitbar: Vorlesung am 14. April - Kapitel 2.3.4.)

Gegeben sei ein quadratisches Schachbrett von $2^n \times 2^n$ Feldern für $n \in \mathbb{N}_{\geq 1}$. Wir entfernen ein beliebiges der vier Eckfelder. Wir wollen alle verbleibenden Felder mit Kacheln auslegen. Dabei überdeckt jede Kachel genau drei Felder und hat die Form wie links dargestellt. In einer legalen Kachelung dürfen sich keine Kacheln überlappen. Hier ist ein Beispiel:





Zeigen Sie durch vollständige Induktion über n, dass stets eine legale Kachelung existiert.