Technische Universität München
Institut für Informatik I-16
Dr. Tobias Lasser

WiSe 2012/13
10. Dezember 2012

Lösungsvorschläge zur Musterklausur zu Algorithmen und Datenstrukturen

Aufgabe 1 Rechnen mit Landau-Symbolen (4 Punkte)

Untersuchen Sie, ob folgende Behauptung wahr ist und begründen Sie Ihre Aussage.

$$4n^2 + 3n - 5 = O(n^3)$$

Wir setzen in die Ungleichung aus der Definition von $O(n^3)$ ein:

$$4n^2 + 3n - 5 \leq c \cdot n^3$$

Anschließend teilen wir durch $n^3 > 0$ (für $n > 0$):

$$\frac{4}{n} + \frac{3}{n^2} - \frac{5}{n^3} \leq c$$

Die Folgen $1/n^i$ konvergieren für alle $n > n_0 = 1$ gegen 0. Wählen wir zusätzlich c hinreichend groß, etwa $c = 3$, so gilt die Ungleichung, und die Behauptung ist gezeigt. □
Bewertung: *1 Punkt für Ungleichung von $O(n^3)$, 1 Punkt für Umformen der Ungleichung, 1 Punkt für eine korrekte Konstante, 1 Punkt für korrekte Schlussfolgerung.*

Aufgabe 2 Boolesche Logik (4 Punkte)

Zeigen Sie, dass der logische Ausdruck $a \Rightarrow b$ identisch ist mit $\neg a \vee b$.

Der Beweis kann etwa über eine Wahrheitstabelle erledigt werden:

a	b	$\neg a$	$\neg a \vee b$	$a \Rightarrow b$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	0	1	1

Die den beiden Seiten der Gleichung entsprechenden Spalten sind gleich, damit ist die Behauptung gezeigt. □
Bewertung: *1 Punkt für Aufstellung der Wahrheitstabelle mit a, b, 1 Punkt für Spalte $\neg a$, 1 Punkt für Spalte $\neg a \vee b$, 1 Punkt für Spalte $a \Rightarrow b$.*

Aufgabe 3 Zahldarstellung (4 Punkte)

Konvertieren Sie die Zahl 7_{10} in eine 4-bit Binärzahl. Berechnen Sie dann das 2-Komplement dieser 4-bit Binärzahl. Als welche zwei Dezimalzahlen lässt sich das entstandene Bitmuster interpretieren?

Konversion in Binärzahl:

$$7_{10} = 4_{10} + 2_{10} + 1_{10} = 0111_2$$

Zweierkomplement ist bitweise Inversion und Addition von 1*:*

$$\overline{0111_2} + 1_2 = 1000_2 + 1_2 = 1001_2$$

Das Muster 1001_2 *kann interpretiert werden als*

- *unsigned-Typ mit Wert* $8_{10} + 1_{10} = 9_{10}$
- *signed-Typ mit Wert* -7_{10}

Bewertung: *1 Punkt für korrekte 4-bit Binärzahl, 1 Punkt für korrektes 2-Komplement, jeweils 1 Punkt für korrekte Interpretation des Bitmusters.*

Aufgabe 4 Sequentielle Liste (1 Punkt)

Gegeben sei ein Feld A implementiert als sequentielle Liste (Array). Was ist die Komplexität der Einfüge-Operation `insert`?

Man muss Elemente hinter der Einfügeposition nach hinten verschieben, diese Operation hat die Komplexität $O(n)$*.*
Bewertung: *1 Punkt für korrekte Komplexität.*

Aufgabe 5 Stack (5 Punkte)

Beschreiben Sie kurz eine Methode, um den abstrakten Datentyp Stack mittels elementarer Datenstrukturen zu implementieren. Erläutern Sie dann ausführlich die Implementation der Operation `push`. Was ist die Komplexität dieser Operation?

Eine Methode nutzt die einfach verkettete Liste mit einem Zusatzzeiger `last` *auf das letzte Element (aber keine Rückwärts-Zeiger im Container, also keine doppelt verkettete Liste).*

Die Push-Operation erzeugt einen neues Element, hängt dieses dann an die Liste bei `last` *an, und setzt schließlich* `last` *auf das neue Element.*

Durch den Zusatzzeiger muss die verkettete Liste nicht durchlaufen werden, daher ist der Aufwand unabhängig von der Stack-Größe, also $O(1)$*.*

Bewertung: *2 Punkte für korrekte Implementierung von Stack mittels elementarer Datenstruktur, 2 Punkte für korrekte Beschreibung der* `push`*-Operation. 1 Punkt für korrekte Komplexität.*

Aufgabe 6 Fibonacci-Folge, Verifikation **(6 Punkte)**

Die Fibonacci-Folge $(f_n)_{n\in\mathbb{N}}$ ist rekursiv definiert als:

$$\begin{aligned} f_1 &= f_2 &&= 1 \\ f_n &= f_{n-1} + f_{n-2} \end{aligned}$$

Nachfolgend ist die aus der Vorlesung bekannte iterative Implementierung nach dem Prinzip der dynamischen Programmierung angegeben als Funktion **fib**(n). Die Verifikation dieser Implementierung ist Gegenstand dieser Aufgabe.

fib(n) :
$fib =$ leeres Feld;
$fib[1] = 1$;
$fib[2] = 1$;
$k = 3$;
$\{C_1 := P \wedge (k = 3)\}$
while $(k \leq n)$ {
$\{C_2 := P \wedge (3 \leq k \leq n)\}$
$fib[k] = fib[k-1] + fib[k-2]$;
$k = k + 1$;
$\{C_3 := P\}$
}
$\{C_4 := P \wedge (k > n) = NACH\}$
return $fib[n]$;

a) Geben Sie jeweils eine geeignete Vor- und Nachbedingung der Funktion **fib**(n) an!

$$\begin{aligned} VOR &= (n > 0) \\ NACH &= (f_n = fib[n]) \end{aligned}$$

Bewertung: *1 Punkt für korrekte Vorbedingung, 1 Punkt für Nachbedingung.*

b) Geben Sie eine geeignete Schleifeninvariante P für die **while** $(k \leq n)$ { ... } Schleife an!

$$P = (f_{k-1} = fib[k-1])$$

Bewertung: *1 Punkt für korrekte Invariante.*

c) Welche drei Bedingungen müssen überprüft werden, damit die Korrektheit der **while** Schleife mittels der Invariante P gezeigt werden kann? Zeigen Sie, dass diese drei Bedingungen in diesem Fall erfüllt sind!

Zu zeigen sind:

- *Schleifeneintritt:* $C_1 \Rightarrow P$
  Trivial, Definition von f_0 *und* f_1.
- *Schleifenkörper:* $\{C_2 := P \wedge (k \leq n)\}...\{P\}$
  Update wie Rekursionsvorschrift, P garantiert richtige Werte im Array.
- *Schleifenende:* $(C_4 := P \wedge \neg(k \leq n)) \Rightarrow NACH$
  P garantiert richtige Werte im Array, $k > n$, *also* $fib[n] = f_n$, *also NACH.*

Bewertung: *1 Punkt jeweils für Bedingung und Anwendung.*

Aufgabe 7 Sortieren **(4 Punkte)**

a) Gesucht ist ein Sortier-Algorithmus, der im Mittel mit Komplexität $O(n \log n)$ sortiert. Welchen Algorithmus schlagen Sie hierfür vor? Nach welchem Algorithmen-Muster wurde dieser Algorithmus entworfen?

Divide and Conquer: Merge oder Quick Sort
Bewertung: *1 Punkt für Merge oder Quick Sort, 1 Punkt für Divide and Conquer.*

b) Gegeben sei eine bereits sortierte Liste von natürlichen Zahlen der Länge n. In diese Liste soll nun ein weiteres Element einsortiert werden. Schlagen Sie einen möglichst effizienten Algorithmus für diese Aufgabe vor und geben Sie dessen Komplexität an.

Insertion Sort ohne die äußere Schleife, damit dann $O(n)$.
Bewertung: *1 Punkt für Insertion Sort, 1 Punkt für Komplexität.*