Technische Universität
Kaiserslautern

Fachbereich Informatik
AG Algorithmen und Komplexität

Ausgabedatum: 24.10.2013
Version: 2013-11-05 19:26

2. Übungsblatt zur Vorlesung Entwurf und Analyse von Algorithmen, WS 13/14

Abgabe: *Bis* ***Donnerstag****, 31.10.2013, 16:00 Uhr, Kasten im Treppenhaus 48-6.*

Basisaufgaben

B1: Fehler finden I

3 Punkte

Benennen Sie alle Fehler in den folgenden Herleitungen; hierbei ist nachzuweisen, dass die Fehler *tatsächlich* solche sind. Leiten Sie außerdem – sofern möglich – korrekte $\mathcal{O}$- bzw. Θ-Schranken her und schreiben Sie Ihre Herleitungen *ohne Notationsmissbrauch*[1] auf.

a)
$$\begin{aligned}\sum_{i=1}^{n}(2i-1)\ln(n) &= \ln(n)\left(\sum_{i=1}^{n}2i-\sum_{i=1}^{n}1\right)\\ &= 2\ln(n)\sum_{i=1}^{n}i - n\ln(n)\\ &= 2\ln(n)\frac{1}{2}n(n+1) - n\ln(n)\\ &= n^2\ln(n)\\ &\in \Theta(n^2\ln(n))\end{aligned}$$

b)
$$\sum_{k=1}^{\infty}\frac{1}{k}\cdot n = n\sum_{k=1}^{\infty}\frac{1}{k} = n\cdot c \in \mathcal{O}(n)$$

Hinweis: Wir verstehen hier $\mathcal{O}$ immer für $n \to \infty$.

[1] Also ohne "=" von und Arithmetik mit Landautermen; "$7n+3 = \mathcal{O}(n)+\mathcal{O}(1) = \mathcal{O}(n)$" wäre demnach nicht zulässig.

B2: Fehler finden II

3 Punkte

Benennen Sie alle Fehler in den folgenden Herleitungen; hierbei ist nachzuweisen, dass die Fehler *tatsächlich* solche sind. Leiten Sie außerdem – sofern möglich – korrekte $\mathcal{O}$- bzw. Θ-Schranken her und schreiben Sie Ihre Herleitungen *ohne Notationsmissbrauch*[2] auf.

a)
$$\begin{aligned}\sum_{k=1}^{n} k\mathcal{O}(n) &= \sum_{k=1}^{n} \mathcal{O}(kn)\\ &= \sum_{k=1}^{n} \mathcal{O}(n)\\ &= n\mathcal{O}(n)\\ &= \mathcal{O}(n^2)\end{aligned}$$

b) $2^{3n} = 2^{\mathcal{O}(n)} = \mathcal{O}(2^n)$

Hinweis: Wir verstehen hier $\mathcal{O}$ immer für $n \to \infty$.

B3: Algorithmenanalyse I

3 Punkte

Schätzen Sie die Worst-Case Laufzeiten (in $\mathcal{O}$-Notation) folgender Prozeduren in Abhängigkeit von `n` möglichst genau ab.

a) Ein (vermeintlicher) Sortieralgorithmus:

```
1  PROCEDURE sort(VAR A:ARRAY OF CARDINAL);
2    VAR i,j,k,n:CARDINAL;
3  BEGIN
4    n := HIGH(A);
5    FOR i := 0 TO n-1 DO
6      FOR j := n TO i+1 BY -1 DO
7        IF A[j-1] > A[j] THEN
8          k := A[j-1];
9          A[j-1] := A[j];
10         A[j] := k;
11       END;
12     END;
13   END;
14 END sort;
```

Sie dürfen `m>1` hier als konstant annehmen.

[2] Also ohne "=" von und Arithmetik mit Landautermen; "$7n+3 = \mathcal{O}(n)+\mathcal{O}(1) = \mathcal{O}(n)$" wäre demnach nicht zulässig.

b) Eine mysteriöse Prozedur:

```
1  PROCEDURE proc(VAR i:CARDINAL; n,m:CARDINAL);
2    VAR j:CARDINAL;
3  BEGIN
4    i := 0;
5    j := m;
6    WHILE ( j <= n ) DO
7      j := m*j;
8      i := i+1;
9    END;
10 END proc;
```

Sie dürfen `m>1` hier als konstant annehmen.

B4: Algorithmenanalyse II

3 Punkte

Schätzen Sie die Worst-Case Laufzeiten (in $\mathcal{O}$-Notation) folgender Prozeduren in Abhängigkeit von `n` möglichst genau ab.

a) Nehmen Sie an, dass `A` ein Array der Größe `n` ist, und dass für alle Elemente $\texttt{A}[i] \in [1..\texttt{m}]$, $1 \leq i \leq \texttt{n}$, gilt. Außerdem sei `B` ein Array der Länge `m`, dessen Einträge mit 0 initialisiert sind.

```
1  PROCEDURE distsort(VAR A: ARRAY OF CARDINAL);
2    VAR i,j,k : CARDINAL;
3  BEGIN
4    FOR i:=1 TO n DO
5      B[A[i]] := B[A[i]] + 1
6    END
7    k:=1
8    FOR j:=1 TO m DO
9      FOR i:=1 TO B[j] DO
10       A[k] := j
11       k := k + 1
12     END
13   END
14 END distsort;
```

Beachten Sie, dass die Laufzeit hier in Abhängigkeit von `n` *und* `m` anzugeben ist!

Aufbauaufgaben

4. Aufgabe

5 Punkte

Zeigen Sie, dass

$$f(n) := \sum_{i=3}^{\frac{n}{2}} \frac{1}{i} \left(\frac{i}{n}\right)^i \in \mathcal{O}(n^{-3}) \,.$$

Hinweis: $f(n) \leq c \cdot g(n) \iff \frac{f(n)}{g(n)} \leq c$, wenn stets $g(n) > 0$.

5. Aufgabe

5 Punkte

Beweisen Sie:

$$n^2\left(a^{\frac{1}{n}} - a^{\frac{1}{n-1}}\right) \sim -\ln a$$

für alle $a > 0$.