TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Fachbereich Informatik
AG Algorithmen und Komplexität

Ausgabedatum: 06.11.2014
Version: 2014-11-05 17:21

2. Übungsblatt für alle Tracks zur Vorlesung Entwurf und Analyse von Algorithmen, WS 14/15

Abgabe: *Bis Freitag, 14.11.2014, 12:00 Uhr, Kasten im Treppenhaus 48-6.*

Basisaufgaben

B2.1: Ein paar Asymptotiken

3 Punkte

Welche der folgenden Aussagen sind wahr?

1) $n^\alpha \in \Theta(n^{\alpha+\varepsilon})$ für alle festen $\alpha, \varepsilon \in \mathbb{R}^+$.
2) $\log(n^\alpha) \in \Theta(\log n)$ für alle festen $\alpha \in \mathbb{R}^+$.
3) $n^n \sim n(n+1)^{n-1}$.

Zeigen bzw. widerlegen Sie sie jeweils!

B2.2: Asymptotiken und *Lemma Auspacken*

3 Punkte

a) Zeigen Sie mit *Lemma Auspacken*[1], dass $e^{\sqrt{n+1}} \in o(2^n)$.

b) Welche der folgenden Aussagen sind wahr?

1) $c^{n+1} \in \Theta(c^n)$ für jedes (feste) $c \in \mathbb{R}^+$.
2) $(n+1)! \in \Theta(n!)$.
3) $(n+1)^k \in \Theta(n^k)$ für jedes (feste) $k \in \mathbb{N}$.

Zeigen bzw. widerlegen Sie sie jeweils!

[1] Sie finden das Lemma im Zusatzdokument „Rechenregeln für Grenzwerte von Quotienten“ auf der Übungswebsite bzw. unter: `http://wwwagak.cs.uni-kl.de/Veroffentlichungen/EAA/EAA-WS-13/14/Rechenregeln-fur-Grenzwerte-von-Quotienten.html`

B2.3: Algorithmenanalyse

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.

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.

B2.4: Dynamische Arrays

3 Punkte

Betrachten Sie eine Listenimplementierung, die ein Array für die Speicherung der Elemente nutzt. Dabei wird die Größe des Arrays immer dann verdoppelt, wenn das Array voll ist und ein weiteres Element eingefügt werden soll. Weil in typischen Speicherverwaltungen ein Array nicht mehr vergrößert werden kann, müssen wir ein neues Array doppelter Länge anlegen und die vorhandenen Elemente umkopieren. Zu Beginn – also im Falle der leeren Liste – starten wir mit einem (leeren) Array der Länge 1.

Berechnen Sie die amortisierten Kosten dafür, ein Element ans Ende der Liste anzuhängen, wenn n Anhängeoperationen (direkt) nacheinander ausgeführt werden. Dabei

betrachten wir als Kosten die Schreibzugriffe, die beim erstmaligen Speichern eines Elements sowie bei Umkopieren anfallen, d. h. das Anhängen eines Elements in ein ausreichend großes Array verursacht Kosten von 1 und das Verlängern des Arrays von Länge i auf $i+j$ kostet i Schreibzugriffe. (Wir gehen davon aus, dass das Array nicht initialisiert wird; dann verbleibt als Aufwand das Kopieren der alten Elemente.)

Hinweis: Die hier analysierte Datenstruktur ist Basis für viele Implementierungen in populären Bibliotheken, zum Beispiel `java.util.ArrayList` in Java Runtime Library oder `std::vector` in der C++ STL.

Aufbauaufgaben

A2.1: Lemmata über Asymptotiken

3 + 1 + 1 + 3 + 3 Punkte

Beweisen Sie die folgenden nützlichen Lemmata:

a) Sei $P(n) = a_0 + a_1 n + \cdots + a_k n^k$ ein beliebiges Polynom mit $a_k > 0$. Dann gilt

$$|P(n)| \sim a_k n^k \ . \qquad \text{(Lemma A: Polynome)}$$

b) Für alle $\alpha, \varepsilon \in \mathbb{R}^+$ gilt

$$n^\alpha \in o\big(n^{\alpha+\varepsilon}\big) \ . \qquad \text{(Lemma A: Poly-Hierarchie)}$$

c) Für alle $c, \varepsilon \in \mathbb{R}^+$ gilt

$$c^n \in o\big((c+\varepsilon)^n\big) \ . \qquad \text{(Lemma A: Exp-Hierarchie)}$$

d) Für alle $\alpha \in \mathbb{R}$ und $c \in (1, \infty)$ gilt

$$\omega(c^{-n}) \ni n^\alpha \in o\left(c^n\right) \ . \qquad \text{(Lemma A: Poly vs. Exp)}$$

e) Für alle $\alpha, \beta \in \mathbb{R}^+$ gilt

$$\log^\alpha n \in o(n^\beta) \ . \qquad \text{(Lemma A: Log vs. Poly)}$$

Hinweis: Man beachte, dass direkt folgt: Für alle $\alpha \in \mathbb{R}_0^+$ und $\beta, \varepsilon \in \mathbb{R}^+$ gilt

$$n^{\alpha-\varepsilon} \prec \frac{n^\alpha}{\log^\beta n} \prec n^\alpha \prec n^\alpha \log^\beta n \prec n^{\alpha+\varepsilon} \ . \qquad \text{(Korollar A: Polylog vs. Poly)}$$

A2.2: 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!

A2.3: Amortisierte Analyse

4 Punkte

Berechnen Sie mit Hilfe von amortisierten Kosten eine möglichst scharfe obere Schranke für die Gesamtkosten (gemessen in der Anzahl der modifizierten Ziffern) des Zählens von 0 nach n bei einer Zahlendarstellung zur Basis b.