Übungen zur Vorlesung Datenstrukturen und Algorithmen

H1 (10 Punkte)

Seien $f, g\colon \mathbf{N} \to \mathbf{R}$ Funktionen. Rekapitulieren Sie, was die Aussagen $f = O(g)$, $f = \Omega(g)$ und $f = \Theta(g)$ formal und intuitiv bedeuten.

Beweisen oder widerlegen Sie die folgenden Aussagen, indem Sie *direkt* die Definition der O-Notation verwenden:

a) $5n^5 + 23n^2 + 1000 = O(n^5)$

b) $1000^{n!} = O\bigl(n^{n^n}\bigr)$

c) $1.01^n = O(n^{100})$

Lösungsvorschlag:

Die Aussage $f = O(g)$, genauer $f \in O(g)$, bedeutet formal, daß es eine positive reelle Konstante c und eine natürliche Zahl n_0 gibt, so daß für alle natürlichen Zahlen $n > n_0$ die Funktionswerte $f(n)$ durch $c \cdot g(n)$ nach oben beschränkt sind.

Es gilt $f = \Omega(g)$ genau dann, wenn $g = O(f)$.

Die Aussage $f = \Theta(g)$ gilt genau dann, wenn gleichzeitig $f = O(g)$ und $f = \Omega(g)$ gelten.

Wir kommen zur intuitiven Bedeutung. Eine Funktion f ist in $O(g)$, wenn sie asymptotisch nicht schneller wächst als g. Sie liegt in $\Omega(g)$, wenn sie asymptotisch nicht langsamer wächst als g. Wenn beides gleichzeitig erfüllt ist, wachsen die Funktionen f und g asymptotisch gleich schnell und wir sagen $f = \Theta(g)$ und $g = \Theta(f)$.

Auf zu den Aufgaben!

a) Diese Aussage ist wahr. Nach Definition ist

$$O(n^5) = \bigl\{\, g\colon \mathbf{N} \to \mathbf{R} \bigm| \exists c \in \mathbf{R}^+\, \exists N \in \mathbf{N}\, \forall n \geq N\colon |g(n)| \leq c \cdot n^5 \,\bigr\}.$$

Offenbar gilt zum Beispiel $5n^5 + 23n^2 + 1000 \leq 6n^5$ ab einem gewissen Wert N. Zum Spaß bestimmen wir das kleinste geeignete N, indem wir ausrechnen, ab wann $n^5 \geq 23n^2 + 1000$ gilt. Es ergibt sich $N = 5$.

b) Auch diese Aussage ist wahr. Nach Definition gilt

$$O(n^{n^n}) = \bigl\{\, g\colon \mathbf{N} \to \mathbf{R} \bigm| \exists c \in \mathbf{R}^+\, \exists N \in \mathbf{N}\, \forall n \geq N\colon |g(n)| \leq c \cdot n^{n^n} \,\bigr\}.$$

Natürlich ist $n! \leq n^n$. Weiterhin gilt für $n \geq 1000$ und $i \geq 1$ auch $1000^i \leq n^i$. Also ist für alle $n \geq 1000$ auch

$$1000^{n!} \leq n^{n^n}.$$

c) Diese Aussage ist falsch, was sich bequem mit der alternativen Definition zeigen läßt. Genauer gesagt wollen wir beweisen, daß

$$\limsup_{n\to\infty} \frac{1.01^n}{n^{100}} = \limsup_{n\to\infty} \frac{e^{\ln(1.01)n}}{n^{100}}$$

nicht existiert. Sowohl die obere als auch die untere Funktion gehen gegen unendlich und sind unendlich oft ableitbar. Eine hundertfache Anwendung von L'Hospital ergibt

$$\lim_{n\to\infty} \frac{e^{\ln(1.01)n}}{n^{100}} = \lim_{n\to\infty} \frac{\ln(1.01)^{100} \cdot e^{\ln(1.01)n}}{100!} = \infty.$$

H2 (10 Punkte)

Ein nicht sehr effizienter Algorithmus, um ein Array **int** [] a zu sortieren, sieht so aus:

```
void sort(int[] a, Random generator) {
  boolean sorted;
  do {
    for(int i=a.length-1; i>0; i--) {
      int j=generator.nextInt(i+1);
      int temp=a[i]; a[i]=a[j]; a[j]=temp;
    }
    sorted=true;
    for(int i=0; i<a.length-1; i++)
      if(a[i]>a[i+1]) sorted=false;
  } while(!sorted);
}
```

Das Array wird immer wieder zufällig permutiert, bis es sortiert ist. Analysieren Sie die durchschnittliche Laufzeit dieses Programms. Gehen Sie dabei davon aus, daß das Array die Größe n hat und n verschiedene Zahlen enthält. (nextInt(i+1) liefert eine zufällige ganze Zahl aus $\{0, 1, \ldots, i\}$.)

Lösungsvorschlag:

Wenn n die Größe des Arrays ist, dann benötigen wir für das Permutieren und Testen lineare Zeit, wenn wir annehmen, daß der Zufallsgenerator uns in konstanter Zeit die nächste Zufallszahl liefert.

Nach jeder Permutation ist die Wahrscheinlichkeit $1/n!$, daß das Array sortiert ist. Im Erwartungswert werden also $n!$ Permutation bis zum Erfolg probiert (geometrische Verteilung!).

Wer die geometrische Verteilung nicht kennt, kann den Erwartungswert auch zu Fuß ausrechnen:

$$\sum_{k=1}^{\infty} k\frac{1}{n!}\Big(1 - \frac{1}{n!}\Big)^{k-1} = n!$$

Die Laufzeit ist daher $n! \cdot \Theta(n) = \Theta(n! \cdot n)$.

H3 (10 Punkte)

Geben Sie möglichst einfache Formeln an, die mit den folgenden Ausdrücken bis auf einen additiven Term $O(1/n)$ übereinstimmen. In der Vorlesung kam $\sqrt{n+1} = \sqrt{n} + O(1/\sqrt{n})$

vor, was noch *nicht* genau genug ist. Genau genug wäre $\sqrt{n}+O(1/n)$, was aber natürlich falsch ist.

a) $\sqrt{n+1}$

b) $\log(2^n + n^7)$

Lösungsvorschlag:

a) $\sqrt{n+1} = \sqrt{n} \cdot \sqrt{1 + \frac{1}{n}} = \sqrt{n}\left(1 + \frac{1}{2n} + O(n^{-2})\right) = \sqrt{n} + \frac{1}{2\sqrt{n}} + O(1/n)$

b) $\log(2^n + n^7) = \log(2^n(1+n^7 2^{-n}) = n + \log(1+n^7 2^{-n}) = n + O(n^7 2^{-n}) = n + O(1/n)$, denn $n^7 2^{-n} = O(1/n)$.

* * * Viel Erfolg! * * *

Hinweise

Diese Woche fanden wegen des Feiertags am Montag noch keine Tutorübungen statt. Daher wird dieses Übungsblatt während der Vorlesung verteilt. Ab nächster Woche erhalten Sie die Übungsblätter in Ihrer Tutorübung.

Hausaufgaben sind mit **H** gekennzeichnet. Sie können die Hausaufgaben in Ihrer Tutorübung abgeben und erhalten Sie korrigiert zurück. Es empfiehlt sich in kleinen Gruppen, zum Beispiel zu zweit, zu arbeiten, da man seine Ideen dann dem Gegenüber erklären muß. Wenn Sie in Gruppen arbeiten, müssen aber dennoch alle Aufgaben von allen Beteiligten bearbeitet werden. Die Aufgaben „aufzuteilen" ist nicht erlaubt. Ebenso muß jeder die Ergebnisse individuell aufschreiben und abgeben.

Für den Studiengang Informatik gilt: Für die Teilnahme an der Klausur sind 50% der Hausaufgabenpunkte zu erbringen. Für andere Studiengänge können andere Regelungen gelten, die individuell erfragt werden können. Das Bearbeiten der Hausaufgaben wird dennoch für alle dringend empfohlen.

Ab nächster Woche werden während der Tutorübung gemeinsam weitere Aufgaben bearbeitet, die auf den kommenden Blättern mit **T** bezeichnet werden. Zusätzlich wird zur Vorbereitung auf die Klausur und zur realistischen Selbsteinschätzung jede Woche eine kurze Aufgabe unter simulierten Klausurbedingungen gerechnet. Diese Aufgaben werden in Zukunft mit **K** gekennzeichnet. Auf diese Weise erhalten Sie ein wöchentliches Feedback und eine frühzeitige Warnung, falls Sie den Stoff nicht erschöpfend verstehen. Dieses Zusatzangebot ist eine Qualitätssicherungsmaßnahme zu Ihrem Besten und keineswegs als Zusatzbelastung zu verstehen. Für den Studiengang Informatik gilt widerum, daß die Durchschnittsnote für diese Aufgaben die Note 4,0 nicht unterschreiten darf.

Aufgrund der getrennten Tutor- und Hausaufgaben gibt es in dieser Vorlesung gegenüber anderen Vorlesungen relativ wenig Hausaufgaben. Auf Wunsch können wir weitere Aufgaben stellen oder vorstellen. Das Schicksal der Zentralübung ist aber leider noch offen. Aufgrund der von der Fachgruppe Informatik nicht zu vertretenden Verschiebung der Vorlesung Stochastik sind wir noch auf der Suche nach einem Ausweichtermin.