RWTH Aachen **SS 2016**
Lehrgebiet Theoretische Informatik Lösungsvorschlag 1
Rossmanith–Hensel–Kuinke–Sánchez 18. April 2016

Übung zur Vorlesung Datenstrukturen und Algorithmen

Aufgabe T1

Wir verwenden die Notation $f(n) \preceq g(n)$ für $f(n) = O(g(n))$:

$$\log(n^3)/\log\log(n) \preceq \log(n) \preceq n\log(n) \preceq n^e \preceq \log(n)^{\log(n)} = n^{\log\log(n)} \preceq n^{\sqrt{n}} \preceq 2^n$$

Aufgabe T2

$$\frac{1}{n+1} - \frac{1}{n} = \frac{n-(n+1)}{n(n+1)} = -\frac{1}{n^2+n}$$

Es ist dann klar, daß $\frac{1}{n^2+n} = O\Big(\frac{1}{n^2}\Big)$, da $\frac{1}{n^2+n} \leq \frac{1}{n^2}$ für $n \geq 1$.

Aufgabe T3

Wir schreiben die Summe zunächst als Integral, was noch nicht viel hilft, schätzen das letztere dann aber mit glatten Funktionen von unten und oben ab.

$$\sum_{k=1}^{n} \frac{1}{k} = \int_1^{n+1} \frac{1}{\lfloor x \rfloor}\, dx$$

Für eine untere Schranke können wir so rechnen:

$$\int_1^{n+1} \frac{1}{\lfloor x \rfloor}\, dx \geq \int_1^{n+1} \frac{1}{x}\, dx = \ln(n+1) \geq \ln(n)$$

Auf ähnliche Weise erhalten wir eine obere Schranke, die nur ein bißchen größer als die untere ist:

$$\int_1^{n+1} \frac{1}{\lfloor x \rfloor}\, dx = 1 + \int_2^{n+1} \frac{1}{\lfloor x \rfloor}\, dx \leq 1 + \int_2^{n+1} \frac{1}{x-1}\, dx = 1 + \int_1^{n} \frac{1}{x}\, dx = 1 + \ln(n)$$

Auf diese Weise wissen wir, daß die Summe zwischen $\ln(n)$ und $\ln(n)+1$ liegt und können $f(n) = \ln(n)$ wählen.
Noch genauer: Für $n \geq 3$ gilt $\ln(n) + 1 \leq 2\ln(n)$ und daher liegt die Summe zwischen $\ln(n)$ und $2\ln(n)$ ab einem gewissen n. Das ist genau, was die Definition von Θ fordert mit $c_1 = 1$, $c_2 = 2$ und $N = 3$.

Aufgabe T4

So könnte man eine entsprechende Methode in Java implementieren:

```
void append(Listnode⟨K, D⟩ head1, Listnode⟨K, D⟩ head2) {
    head1.pred.succ = head2.succ;
    head2.succ.pred = head1.pred;
    head1.pred = head2.pred;
    head2.pred.succ = head1;
}
```

Man beachte, daß dies auch funktioniert, wenn eine (oder gar beide) der Listen leer sind. In einer Sprache ohne *garbage collection* müsste man den zweiten Listenkopf explizit löschen.

Aufgabe H1

a) $n^2(1 + 1/n) = n^2 + n = n^2 + O(n)$

b) $\sum_{k=1}^{n} k = \frac{n(n+1)}{2} = \frac{n^2}{2} + \frac{n}{2} = \frac{n^2}{2} + O(n)$

c) $n!/n^n = 0 + O(n)$ (es gilt sogar $n!/n^n = O(1)$, da $n^n > n!$)

d) $\log(\cos(en)n + n^5) \leq \log(n + n^5) \leq 6\log(n) = 0 + O(n)$

e) $(n+1)^{n+1}/n^n = (n+1)\left(\frac{n+1}{n}\right)^n = (n+1)\left(1 + \frac{1}{n}\right)^n \leq (n+1)e = 0 + O(n)$

f) $\binom{n}{3} = \frac{n(n-1)(n-2)}{6} = \frac{n^3 - 3n^2 + 2n}{6} = \frac{n^3}{6} - \frac{n^2}{2} + O(n)$

Aufgabe H2

Wir schätzen die gegebene Funktion zuerst durch eine obere Schranke ab. Für $n \geq 2$ gilt

$$\sqrt{2n(n+2)(n-1)} \leq \sqrt{2n \cdot 2n \cdot n} = \sqrt{4n^3} = 2n^{3/2}.$$

Für $n \geq 2$ finden wir außerdem folgende untere Schranke.

$$\sqrt{2n(n+2)(n-1)} \geq \sqrt{2n \cdot n \cdot n/2} = \sqrt{n^3} = n^{3/2}.$$

Damit existieren Konstanten $c_1 = 1$, $c_2 = 2$ und $N = 2$, sodaß

$$c_1 \cdot n^{3/2} \leq \sqrt{2n(n+2)(n-1)} \leq c_2 \cdot n^{3/2}$$

für alle $n \geq N$.

Aufgabe H3

```
void vertauscheMitDemNachfolgerRichtig(Listnode⟨K, D⟩ x) {
  Listnode⟨K, D⟩ y = x.succ, a = x.pred, b = y.succ;
  a.succ = y; y.succ = x; x.succ = b;
  b.pred = x; x.pred = y; y.pred = a;
}
```

Eigentlich ist die Lösung nicht so schwierig, wenn man neue Variablen einführt. Man sollte sich aber überzeugen, daß die Lösung auch in Randfällen korrekt ist, zum Beispiel wenn die ganze List nur aus zwei Knoten besteht (was hier der Fall ist). Wir können davon ausgehen, daß es mindestens zwei Knoten gibt und insbesondere, daß der Knoten, den wir erhalten einen Nachfolger besitzt. Das folgt aus der Formulierung „Als Eingabe erhalten Sie den ersten der beiden Listenknoten“.

Eine Quelle vieler Fehler bei solchen Programmen liegt im Versäumnis, über Randfälle nachzudenken.