Datenstrukturen und Algorithmen, SS 2019
Prof. Dr. P. Rossmanith
S. Dollase, J. Dreier, H. Lotze

Datum: 11.04.2019

Übungsblatt mit Lösungen 01

Aufgabe T1

Ordnen Sie die folgenden Funktionen in der Reihenfolge ihres asymptotischen Wachstums (ohne lange Nachzudenken). Mit $\log n$ bezeichnen wir den Logarithmus zur Basis 2.

- $\log(n)$
- $\log(n^5)/\log\log(n)$
- $n^{\log\log n}$
- $\log(n)^{\log n}$
- n^2
- $n\log n$
- 2^n
- $\sqrt{n}$

Lösungsvorschlag

Mit $f(n) \prec g(n)$ drücken wir aus, dass $f(n) = O(g(n))$ gilt.

$$\log(n^5)/\log\log(n) \prec \log(n) \prec \sqrt{n} \prec n\log n \prec n^2 \prec n^{\log\log n} = \log(n)^{\log n} \prec 2^n$$

Aufgabe T2

Sei $x \in \mathbf{R}$ und $\lfloor x \rfloor = \max\{\, k \in \mathbf{Z} \mid k \leq x \,\}$.
Zeigen Sie:

$$\lfloor x \rfloor \leq x < \lfloor x \rfloor + 1$$

Lösungsvorschlag

Konstruktiver Beweis:
Ohne Beschränkung der Allgemeinheit nehmen wir an, dass $x = z + a$ mit $z \in \mathbf{Z}$ und $0 \leq a < 1$.
Dann gilt:

$$\begin{array}{rcll} \lfloor x \rfloor & = & \max\{k \in \mathbf{Z} \mid k \leq x\} & \\ & = & \max\{k \in \mathbf{Z} \mid k \leq z + a\} & \text{mit } x = z + a \\ & = & z & \text{mit } 0 \leq a < 1 \end{array}$$

Es folgt:

$$\begin{array}{rcll} & & 0 \leq a < 1 & \\ \Longrightarrow & & z \leq z + a < z + 1 & \mid +z \\ \Longrightarrow & & \lfloor x \rfloor \leq z + a < \lfloor x \rfloor + 1 & \text{mit } \lfloor x \rfloor = z \\ \Longrightarrow & & \lfloor x \rfloor \leq x < \lfloor x \rfloor + 1 & \text{mit } x = z + a \end{array}$$

Widerspruchsbeweis:
Wir beweisen beide Teile seperat.
$\lfloor x \rfloor \leq x$: Angenommen, die Aussage gelte nicht. Dann existiert ein $c \in \mathbf{R}$, sodass $\lfloor c \rfloor > c$ gilt. Wir ersetzen die Abrundungsklammern durch ihre Definition: $c < \max\{k \in \mathbf{Z} \mid k \leq c\}$. Sei k_1

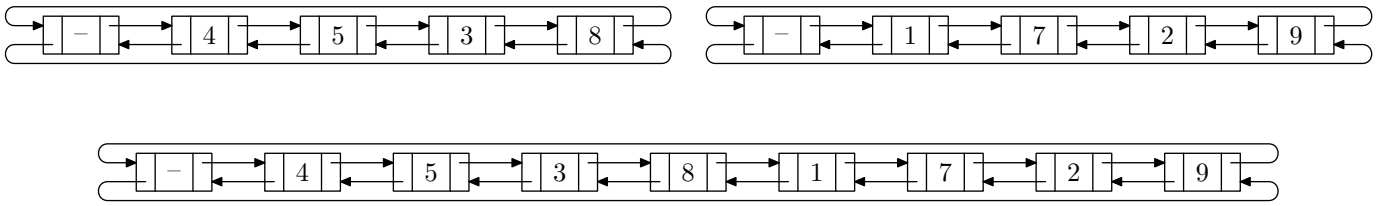
nun dieses größte $k \in \mathbf{Z}$. Dann erhalten wir die Ungleichung $c < k_1 \leq c$, was einen Widerspruch darstellt. Damit exisiert keine reelle Zahl, für die die Aussage nicht gilt.

$x < \lfloor x \rfloor + 1$: Angenommen, die Aussage gelte nicht. Dann existiert ein $c \in \mathbf{R}$, sodass $c \geq \lfloor c \rfloor + 1$ gilt. Wir ersetzen die Abrundungsklammern durch ihre Definition: $c \geq \max\{k \in \mathbf{Z} \mid k \leq c\} + 1$. Sei k_1 nun dieses größte $k \in \mathbf{Z}$. Dann erhalten wir die Ungleichungen $c \geq k_1 + 1$ und $c \geq k_1$, was einen Widerspruch zur Maximilität von k_1 darstellt. Damit exisiert keine reelle Zahl, für die die Aussage nicht gilt.

Aufgabe T3

Entwerfen Sie einen einfachen Algorithmus, der zwei doppelt verkettete Listen aneinanderhängt und auf diese Weise eine neue doppelt verkettete Liste erzeugt. Können Sie dabei auf bedingte Verzweigungen verzichten?

Schreiben Sie den Algorithmus in Pseudocode oder in einer Programmiersprache nieder. Die folgende Grafik zeigt zwei Listen vor und nach der Verkettung.



Lösungsvorschlag

Es seien *head1* und *head2* die Listenköpfe der beiden Listen.

Wir müssen den Nachfolger von *head2* an das Ende der ersten Liste hängen und dann die zyklischen Verweise reparieren. Ein Sonderfall, den wir leider beachten müssen, besteht aus einer leeren zweiten Liste. In diesem Fall müssen wir natürlich gar nichts machen.

```
if(head2.succ ≠ head2) {
    head1.pred.succ = head2.succ;
    head2.succ.pred = head1.pred;
    head2.pred.succ = head1;
    head1.pred = head2.pred;
}
```

Jetzt ist *head1* der Listenkopf der neuen Liste und *head2* wird nicht mehr benötigt.

Florian Tümmers und andere Studierende hatten eine Idee, wie man die Fallunterscheidung vermeiden kann: Man hängt tatsächlich *head2* an das Ende der ersten Liste, so dass jetzt ein zweiter Kopf in der Liste ist, welcher einfach am Ende gelöscht wird:

```
Searchtreenode⟨K, D⟩ last = head2.pred;
head1.pred.succ = head2;
head2.pred.succ = head1;
head2.pred = head1.pred;
head1.pred = last;
head2.delete();
```

Dies sind zwar deutlich mehr Anweisungen, aber wir brauchen tatsächlich keine einzige Fallunterscheidung.

Aufgabe T4

Beweisen oder widerlegen sie, dass $\frac{1}{2}n^2 + 7n + 14 = O(n^2)$ gilt.

Lösungsvorschlag

Wir müssen zeigen, dass es ein $c \in \mathbf{R}^+$ und ein $N \in \mathbf{N}$ gibt, sodass für alle $n \geq N$ gilt, dass $|\frac{1}{2}n^2 + 7n + 14| \leq c|n^2|$. Dazu wählen wir $c = \frac{1}{20}$ und $n = 100$. Wir zeigen, dass die Differenz der Funktionen monoton steigend ist, was mit Standardmethoden der Analysis leicht möglich ist. Außerdem ist die Differenz für die gewählten Werte bereits positiv, wie man leicht nachrechnen kann. Damit haben wir gezeigt, dass die Aussage gilt.

Aufgabe H1 (10 Punkte)
Sei $x \in \mathbf{R}$ und $\lceil x \rceil = \min\{k \in \mathbf{Z} \mid k \geq x\}$.
Zeigen Sie:

$$\lceil x \rceil - 1 < x \leq \lceil x \rceil$$

Lösungsvorschlag
Konstruktiver Beweis:
Ohne Beschränkung der Allgemeinheit nehmen wir an, dass $x = z+a$ mit $z \in \mathbf{Z}$ und $-1 < a \leq 0$.
Dann gilt:

$$\begin{array}{rcll} \lceil x \rceil & = & \min\{k \in \mathbf{Z} \mid k \geq x\} & \\ & = & \min\{k \in \mathbf{Z} \mid k \geq z + a\} & \text{mit } x = z + a \\ & = & z & \text{mit } -1 < a \leq 0 \end{array}$$

Es folgt:

$$\begin{array}{rcll} & & -1 < a \leq 0 & \\ \Longrightarrow & & z - 1 < z + a \leq z & \mid +z \\ \Longrightarrow & & \lceil x \rceil - 1 < z + a \leq \lceil x \rceil & \text{mit } \lceil x \rceil = z \\ \Longrightarrow & & \lceil x \rceil - 1 < x \leq \lceil x \rceil & \text{mit } x = z + a \end{array}$$

Widerspruchsbeweis:
Wir beweisen beide Teile seperat.
$\lceil x \rceil - 1 < x$: Angenommen, die Aussage gelte nicht. Dann existiert ein $c \in \mathbf{R}$ sodass $\lceil c \rceil - 1 \geq c$ gilt. Wir ersetzen die Aufrundungsklammern durch ihre Definition: $c \leq \min\{k \in \mathbf{Z} \mid k \geq c\} - 1$. Sei k_1 nun dieses kleinste $k \in \mathbf{Z}$. Dann erhalten wir die Ungleichungen $c \leq k_1 - 1$ und $c \leq k_1$, was einen Widerspruch zur Minimalität von k_1 darstellt. Damit exisiert keine reelle Zahl, für die die Aussage nicht gilt.
$x \leq \lceil x \rceil$: Angenommen, die Aussage gelte nicht. Dann existiert ein $c \in \mathbf{R}$, sodass $c > \lceil c \rceil$ gilt. Wir ersetzen die Aufrundungsklammern durch ihre Definition: $c > \min\{k \in \mathbf{Z} \mid k \geq c\}$. Sei k_1 nun dieses kleinste $k \in \mathbf{Z}$. Dann erhalten wir die Ungleichung $c > k_1 \geq c$, was einen Widerspruch darstellt. Damit exisiert keine reelle Zahl, für die die Aussage nicht gilt.

Aufgabe H2 (10 Punkte)
Doppelt verkettete Listen wurden in der Vorlesung so implementiert:

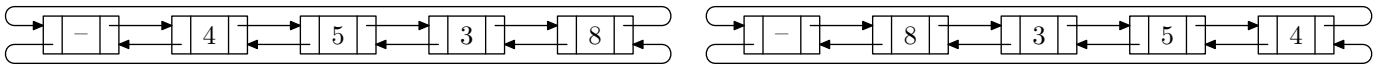
```
public class ADList⟨K, D⟩ extends AbstractMap⟨K, D⟩ {
  private Listnode⟨K, D⟩ head;
  public ADList() {
    head = new Listnode⟨K, D⟩(null, null);
    head.pred = head;
    head.succ = head;
  }
  ...
}

public class Listnode⟨K, D⟩ {
    K key;
    D data;
    Listnode⟨K, D⟩ pred, succ;
    Listnode(K k, D d) { key = k; data = d; pred = null; succ = null; }
    ...
}
```

Implementieren Sie eine neue Methode *reverse*() in der Klasse *ADList*, welche die Reihenfolge in der Liste umkehrt. Dabei sollten Sie so vorgehen, dass kein zusätzlicher Speicher verwendet wird (abgesehen von konstant vielen Variablen), die Implementierung möglichst effizient ist und keine Rekursion verwendet wird.

Beachten Sie, dass Ihre Methode immer funktionieren muss, zum Beispiel auch mit einer leeren Liste.

Hier ist ein Beispiel, wie die Liste vor und nach dem Aufruf von *reverse*() aussehen wird:



Lösungsvorschlag

Eine Möglichkeit besteht darin, durch Zeigerverbiegen die Reihenfolge umzukehren. Im wesentlichen müssen hierfür die Vorgänger und Nachfolger jedes Knotens vertauscht werden. Folgendes Programm erledigt die Aufgabe auf diese Weise:

```
public class ADList⟨K, D⟩ extends AbstractMap⟨K, D⟩ {
  private Listnode⟨K, D⟩ head;
  public ADList() {
    head = new Listnode⟨K, D⟩(null, null);
    head.pred = head;
    head.succ = head;
  }
  public void reverse() {
    Listnode⟨K, D⟩ node = head, tmp;
    do {
      tmp = node.succ;
      node.succ = node.pred;
      node.pred = tmp;
      node = tmp;
    } while(node ≠ head);
  }
  ...
}
```

Eine alternative Methode besteht darin, die *Inhalte* der Knotenpaare $(1, n), (2, n-1), (3, n-2), \ldots, (\lfloor n/2 \rfloor, \lceil n/2 \rceil)$ zu vertauschen:

```
  public void reverse2() {
    Listnode⟨K, D⟩ left = head.succ;
    Listnode⟨K, D⟩ right = head.pred;
    Listnode⟨K, D⟩ tmp = new Listnode⟨K, D⟩(null, null);
    while(right ≠ left && right.succ ≠ left) {
      tmp.copy(left);
      left.copy(right);
      right.copy(tmp);
      left = left.succ;
      right = right.pred;
    }
  }
```

Aufgabe H3 (10 Punkte)

Gegeben seien die Funktionen $f(n)$ und $g(n)$. Beweisen oder widerlegen Sie:

a) $O(f) \cdot O(g) = O(f \cdot g)$

b) Falls $g = O(f)$ und $h = O(f)$, dann gilt auch $g = O(h)$

Lösungsvorschlag

a)
Die Aussage ist wahr. Sei $f_* = O(f)$ und $g_* = O(g)$, dann existieren $c_1, c_2 \in \mathbf{R}^+$ und $n_1, n_2 \in \mathbf{N}$, sodass für alle $n > n_1$, $|f_*(n)| \leq c_1|f(n)|$ und für alle $n > n_2$, $|g_*(n)| \leq c_2|g(n)|$ gilt. Sei $n_0 = \max(n_1, n_2)$ und $c_0 = c_1 \cdot c_2$. Dann gilt für alle $n > n_0$, $|f_*(n)| \cdot |g_*(n)| \leq c_1|f(n)| \cdot c_2|g(n)| = c_0(|f(n) \cdot g(n)|) = O(f \cdot g)$.

b)
Die Aussage ist falsch. Es gilt z.B. $n^2 = O(n^3)$ und $n = O(n^3)$, aber nicht $n^2 = O(n)$.