Fachbereich Mathematik & Informatik
Freie Universität Berlin
Prof. Dr. Ralf Kornhuber, Tobias Kies

8. Übung zur Vorlesung

Computerorientierte Mathematik I

WiSe 2017

http://numerik.mi.fu-berlin.de/wiki/WS_2017/CoMaI.php

Abgabe: Donnerstag, 11. Januar 2018, 14:00 Uhr

1. Aufgabe (4 Bonus TP)
Sei $\mathrm{ggT}(a, b)$ der größte gemeinsame Teiler zweier positiver natürlicher Zahlen a und b. Beweisen Sie

$$\mathrm{ggT}(a, b) = \mathrm{ggT}(b, a \bmod b).$$

2. Aufgabe (4 TP)
Im Folgenden betrachten wird die $\mathcal{O}$ sowie die o-Notation stets für $x \to \infty$. Seien $\alpha, \beta \in \mathbb{R}$ und $g_1, g_2 \colon \mathbb{R} \to \mathbb{R}$ mit $g_1(x) \neq 0 \neq g_2(x)$ für alle $x \in \mathbb{R}$. Beweisen oder widerlegen Sie:

a) $x^\alpha \in \mathcal{O}(x^\beta) \iff \alpha \leq \beta$

b) $x^\alpha \in o(x^\beta) \iff \alpha < \beta$

c) $f_1 \in \mathcal{O}(g_1(x)) \wedge f_2 \in \mathcal{O}(g_2(x)) \iff f_1 + f_2 \in \mathcal{O}(|g_1(x)| + |g_2(x)|)$

d) $f_1 \in \mathcal{O}(g_1(x)) \wedge f_2 \in \mathcal{O}(g_2(x)) \implies f_1 \cdot f_2 \in \mathcal{O}(g_1(x) \cdot g_2(x))$

3. Aufgabe (4 TP)
Für $x \in \mathbb{R}$ und $n \in \mathbb{N}$ betrachten wir den folgenden Algorithmus, der hier in Form eines Pseudocodes dargestellt ist:

```
y = 1
for k = 1,...,n
  y = y + x^k/(k!)
end
```

Hierbei werden x^k und $k!$ in jedem Schleifendurchlauf berechnet durch

$$x^k = x \cdot x \cdots x, \qquad k! = k \cdot (k-1) \cdot \cdots \cdot 2 \cdot 1.$$

a) Beweisen Sie, dass der Aufwand dieses Algorithmus in $\mathcal{O}(n^2)$ für $n \to \infty$ ist. Mit Aufwand ist hier die Anzahl der nötigen arithmetischen Grundrechenoperationen $+, -, \cdot, /$ gemeint.

b) Geben Sie einen alternativen Algorithmus an, der denselben Wert y berechnet, dessen Aufwand aber in $\mathcal{O}(n)$ für $n \to \infty$ liegt. Beweisen Sie dies.

4. Aufgabe (8 PP + 4 Bonus PP)
Wir wollen die Laufzeiten verschiedener Algorithmen zur Bestimmung von $\mathrm{ggT}(a, b)$ zweier positiver natürlicher Zahlen a, b testen. Gehen Sie dafür wie folgt vor:

a) Implementieren Sie den TumbGGT-Algorithmus aus der Vorlesung als eine Funktion `[c,k] = ggT_tumb(a,b)`. Hierbei soll der Rückgabewert `c` der größte gemeinsame gemeinsame Teiler der Eingabevariablen `a` und `b` sein. In `k` soll die Anzahl der im Algorithmus ausgeführten Divisionen zurückgegeben werden. Eine Anwendung von `mod` ist als einfache Division zu werten.

b) Implementieren Sie analog den Algorithmus TumbGGT++ als Funktion mit der Signatur `[c,k] = ggT_tumbpp(a,b)`.

c) Implementieren Sie analog den euklidischen Algorithmus als Funktion mit der Signatur `[c,k] = ggT_euclid(a,b)`.

d) Schreiben Sie eine Skriptdatei `run.m` mit dem folgenden Ablauf:
   - Sei $n = 1000$. Es werden zwei Vektoren $a, b \in \mathbb{R}^n$ mit gleichverteilten Zufallszahlen a_i, $b_i \in \{100, \ldots, 1000\}$ erstellt. Nutzen Sie hierfür die Funktion `randi`.
   - Für jeden der obigen drei ggT-Algorithmen wird ein Vektor $k \in \mathbb{R}^n$ erstellt, sodass k_i die Anzahl der nötigen Divsionen ist, wenn man den jeweiligen Algorithmus für die Eingabedaten a_i und b_i anwendet.

     Im Anschluss an die Berechnung von k wird ein Histogramm für die Häufigkeit der verschiedenen k_i geplottet, in dessen Titel der Name des Algorithmus sowie die Werte von `kmin` $= \min_{i=1,\ldots,n} k_i$ und `kmax` $= \max_{i=1,\ldots,n} k_i$ genannt werden.

   Speichern Sie die Plots jeweils in den Dateien `hist_tumb.png`, `hist_tumbpp.png` und `hist_euclid.png` ab.

e) Geben Sie für jedes der drei Verfahren theoretische untere und obere Schranken für die Werte von $\min_{i=1,\ldots,n} k_i$ und $\max_{i=1,\ldots,n} k_i$ an. Beschreiben Sie außerdem Ihre Beobachtungen aus den erzeugten Daten und interpretieren Sie die Histogramme hinblicklich der theoretischen Schranken. Schreiben Sie Ihre Antwort in die Datei `beobachtungen.txt`.

Hinweis:

- Sie dürfen annehmen, dass für die Eingabdaten stets $a \geq b$ erfüllt ist. Allerdings müssen Sie dann in `run.m` selbst dafür sorgen, dass die Eingabedaten diese Voraussetzung erfüllen. Alternativ können Sie Ihre Algorithmen auch so implementieren, dass diese die Werte von a und b vertauschen, falls $b > a$ gilt.
- Für das Plotten könnten die Funktionen `hist`, `histogram` oder `bar` hilfreich sein. Die Titelzeile können Sie beispielsweise mit `title` und `sprintf` erstellen.

Zur Abgabe der Programme: Packen Sie die Dateien `ggT_tumb.m`, `ggT_tumbpp.m`, `ggT_euclid.m`, `hist_tumb.png`, `hist_tumbpp.png`, `hist_euclid.png` und `beobachtungen.txt` in ein ZIP-Archiv. Benennen Sie das ZIP-Archiv mit Ihrem ZEDAT-Accountnamen und schicken Sie dieses per Email an Ihren Tutor. Achten Sie auch auf Groß- und Kleinschreibung bei den Dateinamen.