**Praktikum Algorithmik SS 2008** Prof. Dr. Heiner Klocke

| | | |
|---|---|---|
| **Aufgabe 1:** | O-Notation | 14.04.2008 |
| **Aufgabe 2:** | Asymptotische Analysen | |
| **Aufgabe 3:** | Experimentelle Analysen | |
| **Aufgabe 4:** | Entscheidbarkeit von Antiviren-Software | |

Name: ..... Matr-Nr: .....

Datum: ……Unterschrift des Dozenten (wenn bestanden): ……

In den folgenden beiden Aufgaben geht es um die *asymptotische* und *experimentelle* Analyse der Laufzeit von Algorithmen. Untersucht werden soll die Laufzeit von Algorithmen in Abhängigkeit von der Größe der Eingabedaten, d.h. wie verhält sich die Laufzeit, wenn die Datenmenge, auf denen der Algorithmus operiert, vergrößert wird?

**Aufgabe 1** (O-Notation)

Zeigen Sie,

- $7n - 2$ liegt in $O(n)$
- $3n^3 + 20n^2 + 5$ liegt in $O(n^3)$
- $3 \log n + \log \log n$ liegt in $O(\log n)$

**Aufgabe 2** (Präfixdurchschnitt einer Zahlenfolge, BubbleSort)

Gegeben sei ein Array X, welches n Zahlen speichert. Bestimmen Sie ein Array A, so dass A[i] der Durchschnitt der Elemente X[0], …, X[i] für i=0, …, n-1 ist. Man nennt A den **Präfixdurchschnitt** (engl. prefix average) der Zahlenfolge X.

$$A[i] = \frac{\sum_{j=0}^{i} x[j]}{i+1}$$

Der Präfixdurchschnitt hat viele Anwendungen in der Ökonomie (z.B. Fondwertentwicklung über die letzten Jahre) und Statistik. Eine weitere wichtige Bedeutung liegt in der Glättung von Parametern, die sich schnell verändern (→ Bild 1).

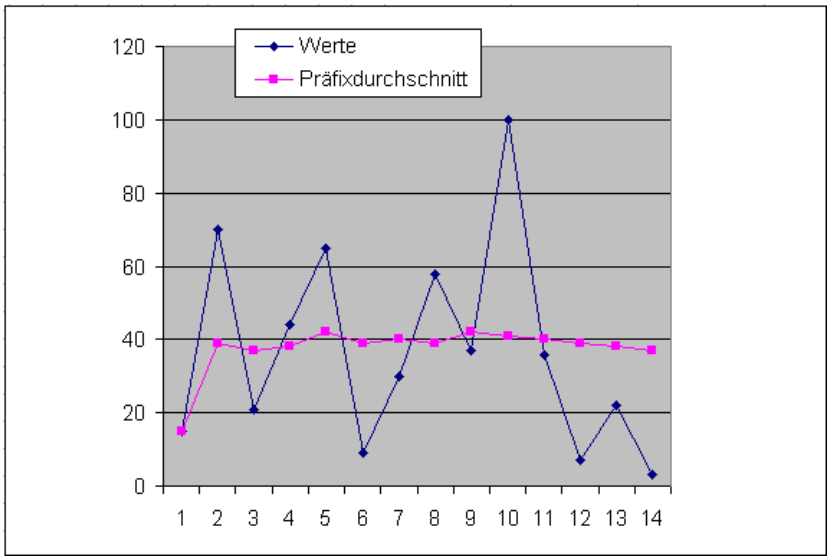


**Bild 1**: Beispiel einer Präfixdurchschnitt-funktion, die sehr unterschiedliche Datenwerte glättet.

Die Algorithmen **prefixAverages1** und **prefixAverages2** lösen beide das Präfixdurchschnitt-Problem mit unterschiedlicher Performance.

**a)** Implementieren Sie die beiden Algorithmen **prefixAverages1** und **prefixAverages2**. Erzeugen Sie die Daten X[i] und A[i], i=0, …, n-1 mit einem Javaprogramm und speichern sie diese in Java-Objekten.

**b)** Stellen Sie die Daten als Liniendiagramm (Scatterplot), ähnlich wie in Bild 1, dar. Verwenden Sie zur Darstellung der Diagramme eine Java-Klassenbibliothek wie z. B. *Ptplot 5.7* (Der Link http://ptolemy.berkeley.edu/java/ptplot/ befindet sich auch auf meiner Homepage)

**c)** Führen Sie für die beiden Algorithmen anhand des Pseudocodes eine asymptotische Laufzeitanalyse für die obere Laufzeitschranke durch.

**Algorithmus** prefixAverages1(X)
*Input:* ein n-elem. Zahlenarray X
*Output:* ein n-elem. Zahlenarray A, so dass A[i] der ∅ der Elemente X[0], … X[n] ist

```
Sei A ein Array mit n Zahlen.
for i ← 0 to n-1 do
    a ← 0
    for j ← 0 to i do
        a ← a + X[j]
    A[i] ←a/(i+1)
return array A
```

**Algorithmus** prefixAverages2(X)
*Input:* ein n-elem. Array X von Zahlen
*Output:* ein n-elem. Zahlenarray A, so dass A[i] der ∅ der Elemente X[0], … X[n] ist

```
Sei A ein Array mit n Zahlen.
s ← 0
for i ← 0 to n-1 do
    s ← s + X[i]
    A[i] ←s/(i+1)
return array A
```

**d)** Schreiben Sie den Sortieralgorithmus ***BubbleSort*** als Pseudocode auf. Implementieren Sie ***BubbleSort*** und stellen Sie das Laufzeitverhalten grafisch, z.B. mit Hilfe der *Ptplot 5.7* – Bibliothek auf dem Bildschirm dar. Zählen Sie bei der Ausführung von *BubbleSort* die Elementaroperationen und geben Sie die Daten sowohl in einer 2-spaltigen Tabelle als auch grafisch auf dem Bildschirm aus. Tragen Sie auf der X-Achse die Größe der Eingabedaten und auf der Y-Achse die

Anzahl der Elementaroperationen ein. Analysieren Sie *BubbleSort* anhand des Pseudocodes und vergleichen Sie das Ergebnis mit dem grafischen Ergebnis.

Verwenden Sie für das Experiment verschiedene Eingabedaten und wiederholen Sie die Test für jede Eingabegröße mind. 10 mal. Verwenden Sie

1. Gut gemischte, zufällig erzeugte Daten
2. absteigend vorsortierte Daten
3. aufsteigend vorsortierte Daten

**Aufgabe 3** (zwei experimentelle Analysen)

Führen Sie für die drei Algorithmen aus Aufgabe 2 experimentelle Analysen als Blackboxtest durch. Dazu müssen Sie die Laufzeit für jede Eingabegröße in Millisekunden messen. Damit das möglich ist, müssen genügend große Datenmengen verwendet werden. Außerdem muss für jede Eingabegröße die Messung mehrfach durchgeführt und dann der Mittelwert gebildet werden, damit keine Artefakte die Messung verfälschen.

**Experimentelle Analyse 1.** Führen Sie für die beiden Algorithmen **prefixAverages1** und **prefixAverages2** eine sorgfältige experimentelle Analyse durch, um Aussagen über die Laufzeit zu gewinnen. Verwenden Sie den Ratio Test ($\rightarrow$ Bild 2) und den Power Test ($\rightarrow$ Bild 3). Geben Sie die Laufzeiten als Funktion der Eingabegrößen für jeden Algorithmus in Form eines Liniendiagramms aus. Verwenden Sie für die grafische Darstellung zwei verschiedene Skalen: eine linear-linear Skala und eine log-log Skala. Verwenden Sie repräsentative Werte für n und führen Sie **mindestens 10 Tests** für jede Größe von n durch.

**Experimentelle Analyse 2.** Analysieren Sie nach dem gleichen Schema den BubbleSort-Algorithmus.

**Hinweis:** Verwenden Sie für das Zeichnen der Diagramme die Javabibliothek „Ptplot“, http://ptolemy.berkeley.edu/java/ptplot/

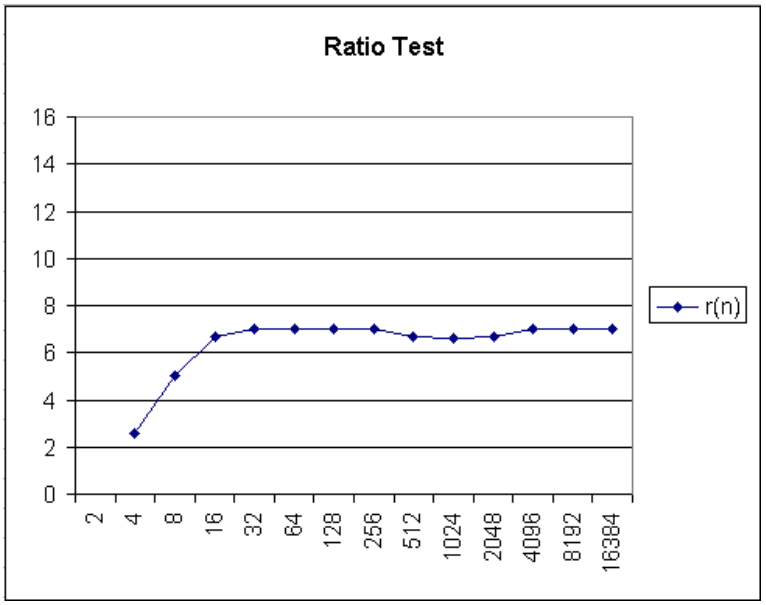


**Bild 2:** Vermutete Laufzeit des Algorithmus $f(n) = n^c$. Laufzeit für eine Problemgröße n: t(n). Prüfe, ob die mittlere Laufzeit des Algorithmus gleich $\Theta(n^c)$ ist.

**Ratio Test**: Zeichne für verschiedenen Werte t(n) das Verhältnis: r(n) = t(n) / f(n). Das Beispielbild links zeigt, einen Plot mit r(n) = 7.

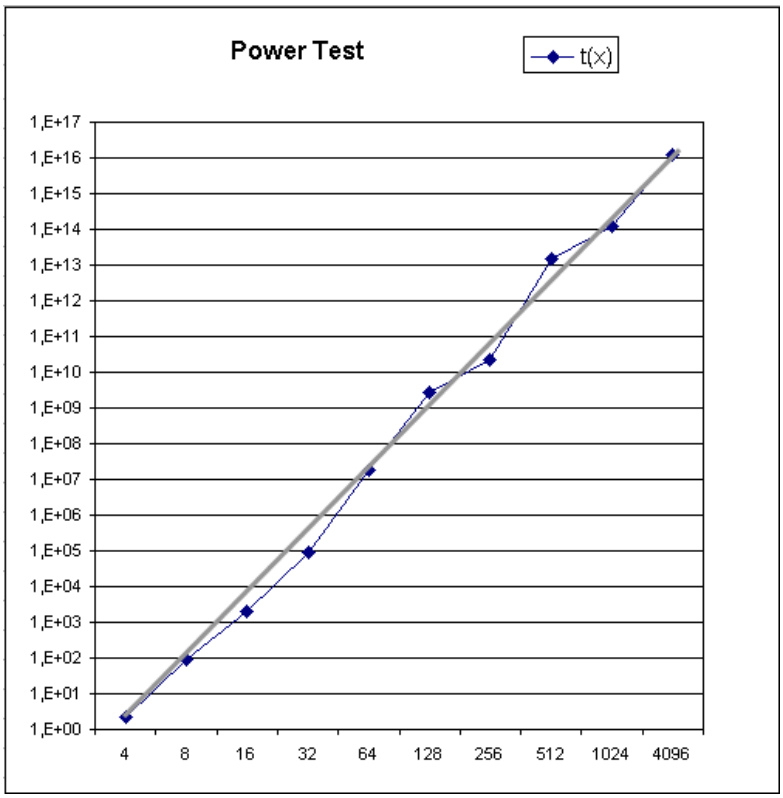


**Bild 3:** Beim **Power Test** werden Paare (x, y) erzeugt mit y = t(x), wobei t(x) die ermittelte Laufzeit für die Größe x einer Beispieleingabe ist.

Transformiere (x, y) $\rightarrow$ (x', y'), wobei x' = log x und y' = log y. Zeichne alle Paare (x', y') und prüfe die Ergebnisse.

Aus dem Plot des Power Tests links schätzen wir, dass gilt: $y' = (4/3)x' + 2$; daraus schätzen wir $t(n) = 2n^{4/3}$

**Anmerkung**: Es handelt sich hier nur um ein grafisches Muster.

**Aufgabe 4** ( Entscheidbarkeit von Antiviren-Software )

Wir definieren das Computerviren-Problem wie folgt:

> Ist es möglich, ein universelles Programm zu schreiben, das für jedes Programm p als Eingabe entscheidet, ob p den Computer infiziert?

Zeige, dass das Computerviren-Problem nicht entscheidbar ist.

Hinweis: Betrachte das Halteproblem.