Algorithmik SS 2003				Prof. Dr. Heiner Klocke
Praktikum 1:	Asymptotische und experimentelle Analyse von Algorithmen			
Späteste Abnahme:	A2, B2: 29.04.	C2: 30.04.	A1, B1: 06.05.	C1: 07.05.
Name:		Matr-Nr:		
Datum: …………	Unterschrift des Dozenten (wenn bestanden): ………..………………			

Die folgenden zwei Aufgaben beschäftigen sich mit dem für die Algorithmik grundlegenden Thema der asymptotischen und experimentellen Laufzeitanalyse. Das Ziel ist es, das Laufzeitverhalten von Algorithmen unabhängig von der Hard- und Softwareumgebung zu untersuchen. Die Beurteilung der Qualität eines Algorithmus muss unabhängig von der Performance des Rechners, des Betriebssystems und der gewählten Programmierumgebung erfolgen. Alle diese Parameter sind technischer Art und werden sich mit der Zeit verändern. Um zum Beispiel die Qualität zweier Algorithmen miteinander vergleichen zu können, dürfen Parameter der Laufzeitumgebung nicht betrachtet werden. Die *asymptotische* **und** *experimentelle* Analyse eines Algorithmus untersucht sein Laufzeitverhalten in Abhängigkeit von stetig wachsenden Eingabegrößen der Daten. Es geht hier also um die Frage: Wie verhält sich die Laufzeit des Algorithmus, wenn man die Datenmenge, auf denen der Algorithmus operiert, immer weiter vergrößert?

Aufgabe 1 (Präfixdurchschnitt einer Zahlenfolge, asymptotische Analyse)

Gegeben sei ein Array X, welches n Zahlen speichert. Sie sollen ein Array A berechnen, 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 (engl. smoothing) von Parametern, die sich schnell verändern (→ Bild 1).

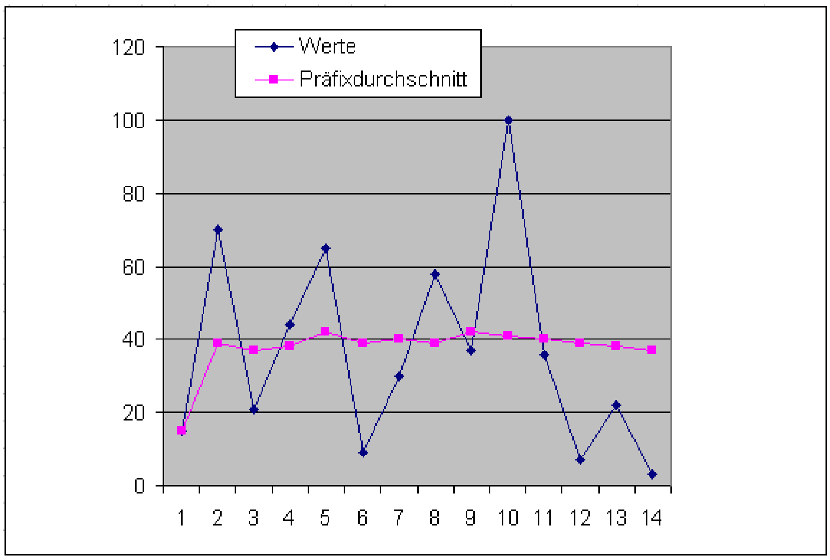

Bild 1: Beispiel einer Präfixdurchschnittfunktion, die sehr unterschiedliche Datenwerte glättet.

Die folgenden Algorithmen prefixAverages1 und prefixAverages2 lösen beide das Präfixdurchschnitt-Problem, jedoch mit unterschiedlicher Performance.

Programmierung. Implementieren Sie die beiden Algorithmen prefixAverages1 und prefixAverages2. Erzeugen Sie jeweils eine Textdatei, in der Sie die Daten X[i] und A[i], i=0, …, n-1, sowie die Beschriftung der x- und y-Achse speichern.

Darstellung. Importieren Sie die Textdatei in eine Excel-Tabelle, und lassen Sie sich die Daten als Liniendiagramm (Scatterplot), ähnlich wie in Bild 1, anzeigen. Eleganter ist es natürlich, eine Java-Klassenbibliothek zur Darstellung der Diagramme zu verwenden wie z. B. "*Ptplot 5.1p2 - Java Plotter"*. Ein Link befindet sich auf der Webseite des Praktikums.

Asymptotische Analyse. Führen Sie für die beiden Algorithmen prefixAverages1 und prefixAverages2 jeweils eine asymptotische Analyse 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[j] ←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
```

Aufgabe 2 (zwei experimentelle Analysen)

Bei der asymptotischen Analyse wird der Pseudocode eines Algorithmus untersucht. Mit mathematischen Werkzeugen wie Amortisation, Summation, Differenzgleichungen, usw. versucht man, eine obere Schranke für die Laufzeit des Algorithmus zum Beispiel bei einer Worst-Case-Wahl von Daten zu beschreiben. Die asymptotische Analyse gibt jedoch keine Auskunft über konstante Faktoren, die sich in der O-Notation verbergen. Ebenfalls erhält man keine Auskunft darüber, ob und wann es sinnvoll ist, einen langsamen Algorithmus mit kleinen konstanten Faktoren zu verwenden oder einen schnellen Algorithmus mit großen konstanten Faktoren. Außerdem ist bei komplizierten Algorithmen die asymptotische Analyse meist schwierig. In solchen Fällen kann die experimentelle Analyse helfen, Aussage über die Laufzeiteigenschaften eines Algorithmus zu gewinnen. – Ihre Aufgabe: Führen Sie eine der beiden folgenden experimentellen Analysen durch.

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 (→Bild 2) und den Power Test (→ 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 Wert für n und führen Sie mindestens fünf Tests für jede Größe von n durch.

Experimentelle Analyse 2. Vergleichen Sie in einer sorgfältigen experimentellen Analyse die Laufzeiten der folgenden Methoden. Benutzen Sie die in der Vorlesung besprochenen Tests *Ratio Test* (→ Bild 2) und *Power Test* (→ Bild 3), um die Laufzeiten der verschiedenen Methoden abzuschätzen. Stellen Sie die Ergebnisse mit Hilfe einer Excel-Tabelle und entsprechender Diagramme grafisch dar.

Algorithmus `Loop1`(n):
  s ← 0
  for i ← 1 **to** n **do**
    s ← s+i

Algorithmus `Loop2`(n):
  p ← 1
  for i ← 1 **to** 2n **do**
    p ← p·i

Algorithmus `Loop3`(n):
  p ← 1
  for i ← 1 **to** n^2 **do**
    p ← p·i

Algorithmus `Loop4`(n):
  s ← 0
  for i ← 1 **to** 2n **do**
    for j ← 1 **to** i **do**
      s ← s+i

Algorithmus `Loop5`(n):
  s ← 0
  for i ← 1 **to** n^2 **do**
    for j ← 1 **to** i **do**
      s ← s+i

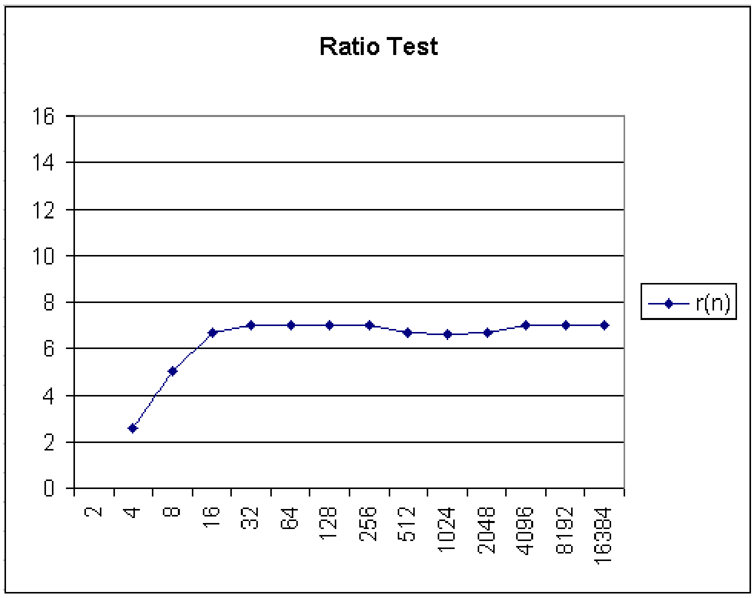

Bild 2: Vermutete Laufzeit des Algorithmus f(n) = n^c. Laufzeit für eine Problemgröße n: t(n). Prüfe, ob die mittlere Laufzeit des Algorithmus gleich $\Theta(n^c)$ ist.

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.

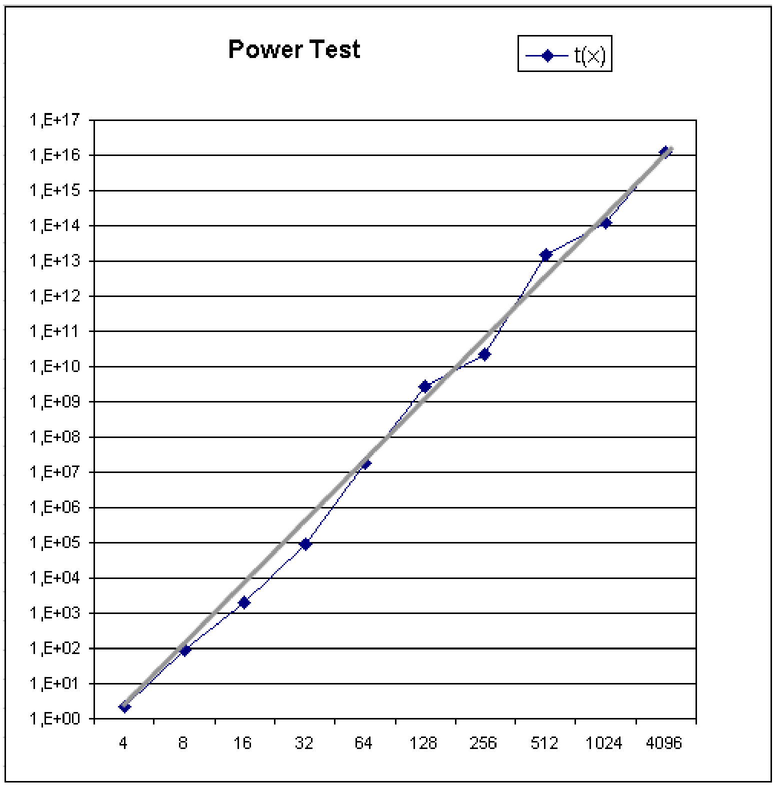

Bild 3: Beim Power Test werden Paare (x, y) erzeugt mit y = t(x), wobei t(x) die ermittelte Laufzeit für die Größe x einer Beispieleingabe ist.

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

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

Anmerkung: Es handelt sich hier nur um ein grafisches Muster. Die Konstanten sind in dem Diagramm wegen der ungünstigen Excelformatierung nicht genau abzulesen.