**Themen: Asymptotische Laufzeit von Algorithmen**
**Experimentelle Analyse von Algorithmen**

## Aufgabe 1 ( Asymptotische Laufzeit )

Erklären Sie, was man unter der asymptotischen Laufzeit eines Algorithmus versteht.

- Was bedeutet es, wenn man von der *unteren Schranke* und der *oberen Schranke* einer asymptotischen Laufzeit spricht?
- Wie werden asymptotische Laufzeiten von Algorithmen formal beschrieben?

## Aufgabe 2

Logarithmische Funktionen werden in O-Notation als O(log n) beschrieben. Warum kann die Basis des Logarithmus weggelassen werden?

$$log_a b = \frac{log_b n}{log_b a}$$

## Aufgabe 3 ( Laufzeitkurven )

Algorithmische Komplexitätsklassen werden durch Laufzeitfunktionen beschrieben. Zeichnen Sie in ein Koordinatensystem die Laufzeitkurven für folgende Komplexitätsklassen von Algorithmen und beschriften Sie jede Kurven mit der entsprechenden mathematischen Funktion. Geben Sie für jede Komplexitätsklasse – soweit Ihnen bekannt – einen Algorithmus an, für den diese asymptotische Laufzeit zutrifft.

1. konstante Laufzeit
2. logarithmische Laufzeit
3. lineare Laufzeit
4. „n log n“-Laufzeit
5. polynomiale Laufzeit, z.B. $n^2$ und $n^3$
6. exponentielle Laufzeit
7. hyper-exponentielle Laufzeit

**Tipp**: Eine gute Vorbereitung für das Praktikum ist es, wenn Sie das Koordinatensystem und die Laufzeitkurven in Java programmieren und mit Hilfe der Bibliothek *Ptplot5.9* grafisch ausgeben.

## Aufgabe 4 ( Experimentelle Analyse mit dem Power-Test )

**a)** Angenommen, Sie haben bei einer Softwarefirma verschiedene Algorithmen als Java-Klassen gekauft. Mit den Java-Klassen und der Dokumentation erhalten Sie folgende asymptotische Laufzeitinformationen:

Algorithmus 1: $O(n)$
Algorithmus 2: $O(100 * n^2)$
Algorithmus 3: $O(300 * n^{3,5})$
Algorithmus 4: $O(10\ n^4 + n^3 + 100\ n^2)$
Algorithmus 5: $O(5^{2n})$

Sie wollen überprüfen, ob die Angaben der Softwarefirma über die Laufzeiten korrekt sind. Welche der fünf Algorithmen können Sie durch Anwendung des Power-Tests überprüfen und welche nicht?

**b)** Erklären Sie die Vorgehensweise beim Power-Test.

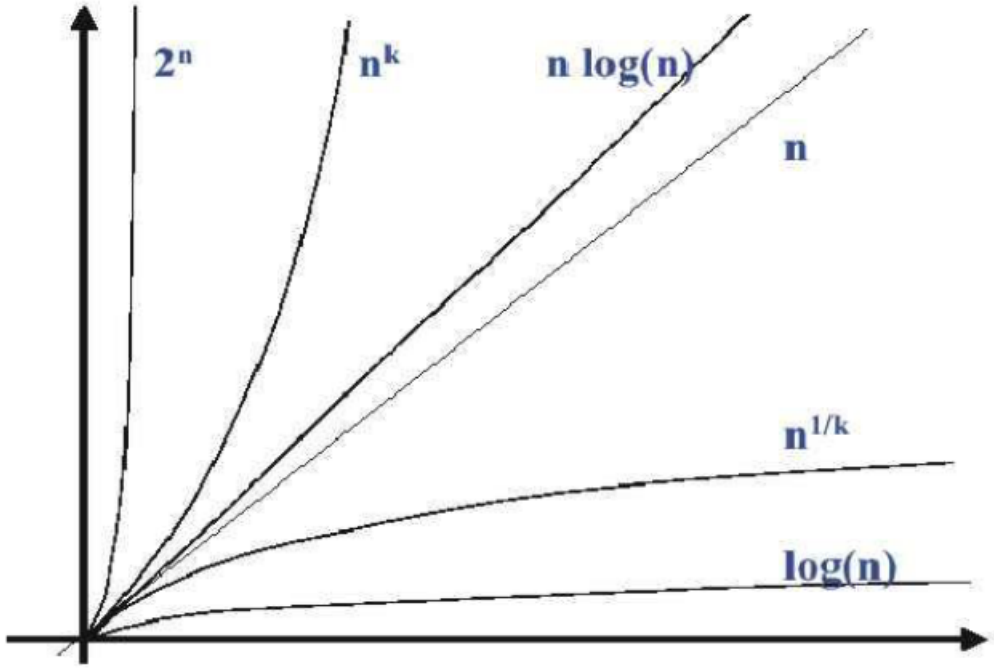
Warum sind eine Regressionsanalyse und die Angabe des Korrelationskoeffizienten r beim Power-Test erforderlich?

## Aufgabe 5 ( Rechnen mit der O-Notation )

Zeigen Sie: aus $f(n) \in O(s(n))$ u. $g(n) = O(r(n))$ **folgt nicht** $f(n) - g(n) \in O(s(n) - r(n))$

Zeigen Sie: aus $f(n) \in O(s(n))$ u. $g(n) = O(r(n))$ **folgt nicht** $f(n) / g(n) \in O(s(n) / r(n))$

## Lösung Aufgabe 3




1. **konstante Laufzeit**

   Viele Anweisungen werden in der Mehrzahl der Programme einmal oder höchstens einige Male ausgeführt. Wenn alle Anweisungen des Programms diese Eigenschaft haben ist die Laufzeit konstant. Dies sollte bei der Entwicklung von Algorithmen angestrebt werden.

2. **logarithmische Laufzeit**

   Ist die Laufzeit logarithmisch, so wird das Programm mit wachsendem n langsamer. Dies ist so bei Programmen die umfangreiche Probleme lösen. Sie wandeln diese in ein kleineres um, wobei der Umfang auf einen gewissen konstanten Anteil verringert wird.

3. **lineare Laufzeit**

   Ist die Laufzeit eines Programms linear, entfällt im Allgemeinen ein kleiner Anteil der Verarbeitung auf jedes Element der Eingabedaten. Wäre n zum Beispiel eine Million, so ist die Laufzeit ebenfalls eine Million. Immer wenn n sich verdoppelt, gilt die Verdopplung auch für die Laufzeit. Dies ist die optimale Situation für Algorithmen die n Eingabedaten verarbeiten müssen. (n Ausgabedaten erzeugen müssen.)

4. **„n log n“-Laufzeit**

   Diese Laufzeit kommt bei Algorithmen vor, die ein Problem lösen, indem sie es in kleinere Teilprobleme zerlegen. Diese Teilprobleme werden unabhängig voneinander gelöst, dann

werden die Lösungen kombiniert. Wenn n eine Million wäre, beträgt n log n ungefähr sechs Millionen.

5. **polynomiale Laufzeit, z.B. $n^2$ und $n^3$**

   **$n^2$**
   Wenn ein Algorithmus eine quadratische Laufzeit hat, lässt er sich nur für relativ kleine Probleme anwenden. Diese Algorithmen verarbeiten alle paarweise Kombinationen von Datenelementen(möglicherweise in einer doppelt verschachtelten Schleife). Wenn n eintausend ist, beträgt die Laufzeit eine Million. Würde sich n verdoppeln, vervierfacht sich die Laufzeit.

   **$n^3$**
   Ebenso wie für $n^2$, gilt für $n^3$, dass ein Tripel von Datenelementen verarbeitet wird. Wenn n einhundert beträgt, so ist die Laufzeit eine Million. Würde sich n verdoppeln, erhöht sich die Laufzeit auf das achtfache.

6. **exponentielle Laufzeit, z.B. $2^n$**

   Nur wenige Algorithmen mit exponentieller Laufzeit, geeignet. Sie treten jedoch auf natürliche Weise Problemen auf. Wenn n zwanzig beträgt, ist die Laufzeit Verdopplung von n, wird die Laufzeit quadriert.

7. **Hyper-exponentielle Laufzeit**

   Ackermann-Funktion

## Lösung Aufgabe 4 ( Experimentelle Analyse mit dem Power-Test )

**a)** Angenommen, Sie haben bei einer Softwarefirma verschiedene Algorithmen als Java-Klassen gekauft. Mit den Java-Klassen und der Dokumentation erhalten Sie folgende asymptotische Laufzeitinformationen:

Algorithmus 1: $O(n)$
Algorithmus 2: $O(100 * n^2)$
Algorithmus 3: $O(300 * n^{3,5})$
Algorithmus 4: $O(10\, n^4 + n^3 + 100\, n^2)$
Algorithmus 5: $O(5^{2n})$

Sie wollen überprüfen, ob die Angaben der Softwarefirma über die Laufzeiten korrekt sind. Welche der fünf Algorithmen können Sie durch Anwendung des Power-Tests überprüfen und welche nicht?

**Lösung.** Das Ziel des Logarithmierens beim Powertest ist es, aus einem Polynom größer 2. Grades eine lineare Funktion zu machen. **Algorithmus 1** hat schon lineare Laufzeit, hier kann aus das Logarithmieren verzichtet werden.
**Algorithmus 5** hat exponentielle Laufzeit. Das Logarithmieren würde zu einem Polynom größer gleich 2. Grades führen. Damit kann die lineare Regression nicht angewendet werden. Der Algorithmus kann daher nicht mit dem Powertest analysiert werden.
**Alle anderen Algorithmen** können mit dem Powertest experimentell untersucht werden.

**b)** Erklären Sie die Vorgehensweise beim Power-Test.

Warum sind eine Regressionsanalyse und die Angabe des Korrelationskoeffizienten r beim Power-Test erforderlich?

## Lösung Aufgabe 5 ( Rechnen mit der O-Notation )

Zeigen Sie: aus $f(n) \in O(s(n))$ u. $g(n) = O(r(n))$ **folgt nicht** $f(n) - g(n) \in O(s(n) - r(n))$

$f(n) = 4n^3 + 7n^2 + 5 \quad \in O(n^3+n)$

$g(n) = 4n^2 + n \quad \in O(n^3)$

**$f(n) - g(n)$** $= 4n^3 + 3n^2 - n + 5 \quad \in O(n^3)$

$s(n) = O(n^3+n) \quad r(n) = O(n^3)$ **$s(n) - r(n) = n$**

Aber $f(n) - g(n) = 4n^3 + 3n^2 - n + 5 \notin O(s(n) - r(n)) = O(n)$

Zeigen Sie: aus $f(n) \in O(s(n))$ u. $g(n) = O(r(n))$ **folgt nicht** $f(n) / g(n) \in O(s(n) / r(n))$