

Datenstrukturen, Algorithmen und Programmierung 2

Professor Dr. Petra Mutzel

Lehrstuhl für Algorithm Engineering, LS11

2. VO

6. April 2006

Motivation

„Warum soll ich in DAP2 gehen?“

MERKE: DAP2 IST WICHTIG!!!

„Ich kann doch schon programmieren.“

ABER NICHT IMMER EFFIZIENT!

Überblick

Asymptotische Schranken

Asymptotische Schranken

Asymptotische Schranken

Asymptotische Schranken

WICHTIG!!!

Laufzeitverhalten eines Algorithmus

wichtig für:

- die Vorhersage
 - Wieviel Zeit, wenn sich die Eingabegröße verdoppelt?
- den Vergleich von Algorithmen
 - Welcher ist besser?
- die Analyse von Algorithmen
 - Wie gut ist der Algorithmus?

Algorithmen-Vergleich

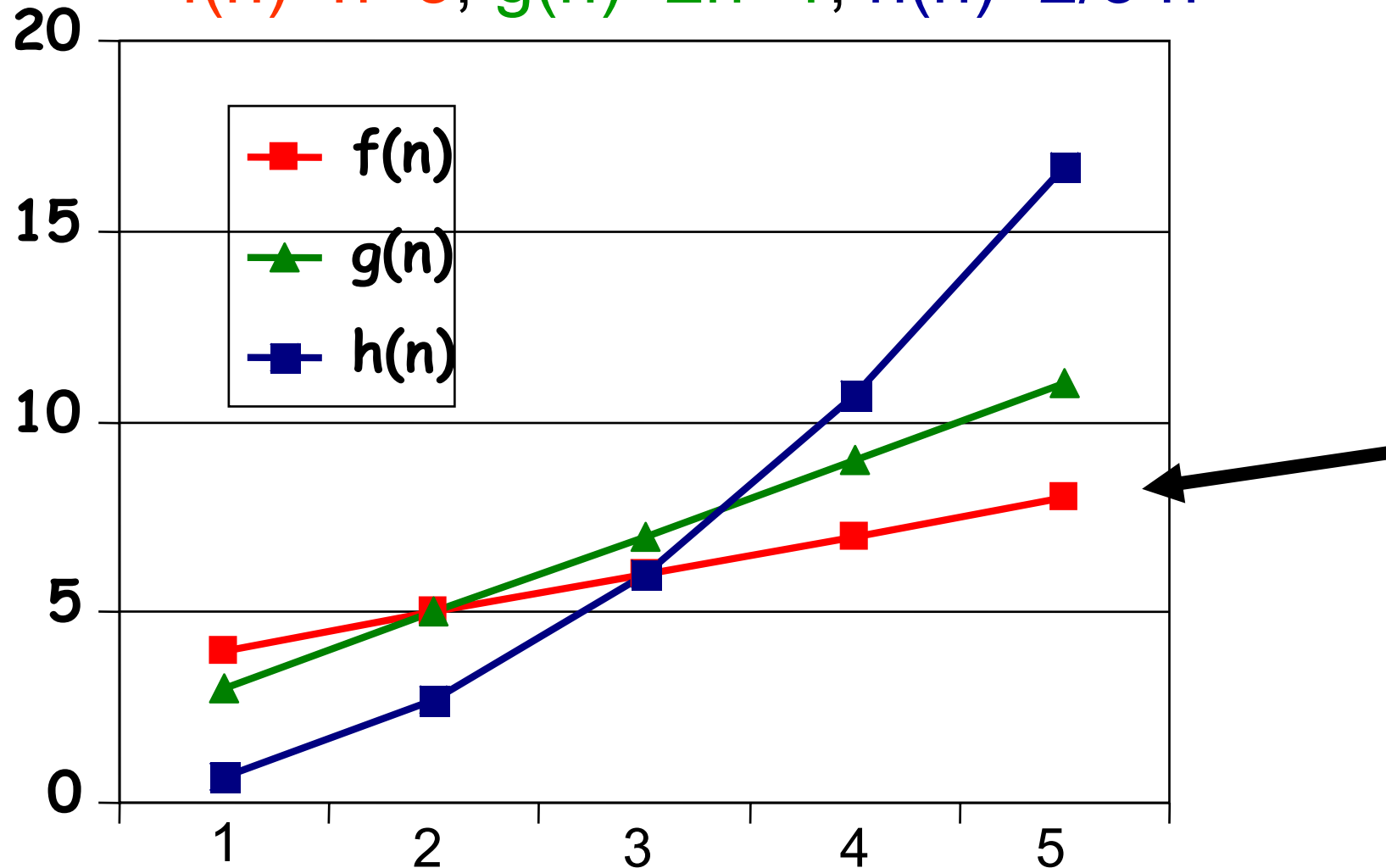
- Fridolin: $f(n)=n+3$
- Gwendoline: $g(n)=2n+1$
- Harlekin: $h(n)=\frac{2}{3} n^2$

- Welcher ist der effizienteste?

n	1	2	3	4	5
f(n)	4	5	6	7	8
g(n)	3	5	7	9	11
h(n)	0,7	2,7	6	10,7	16,7

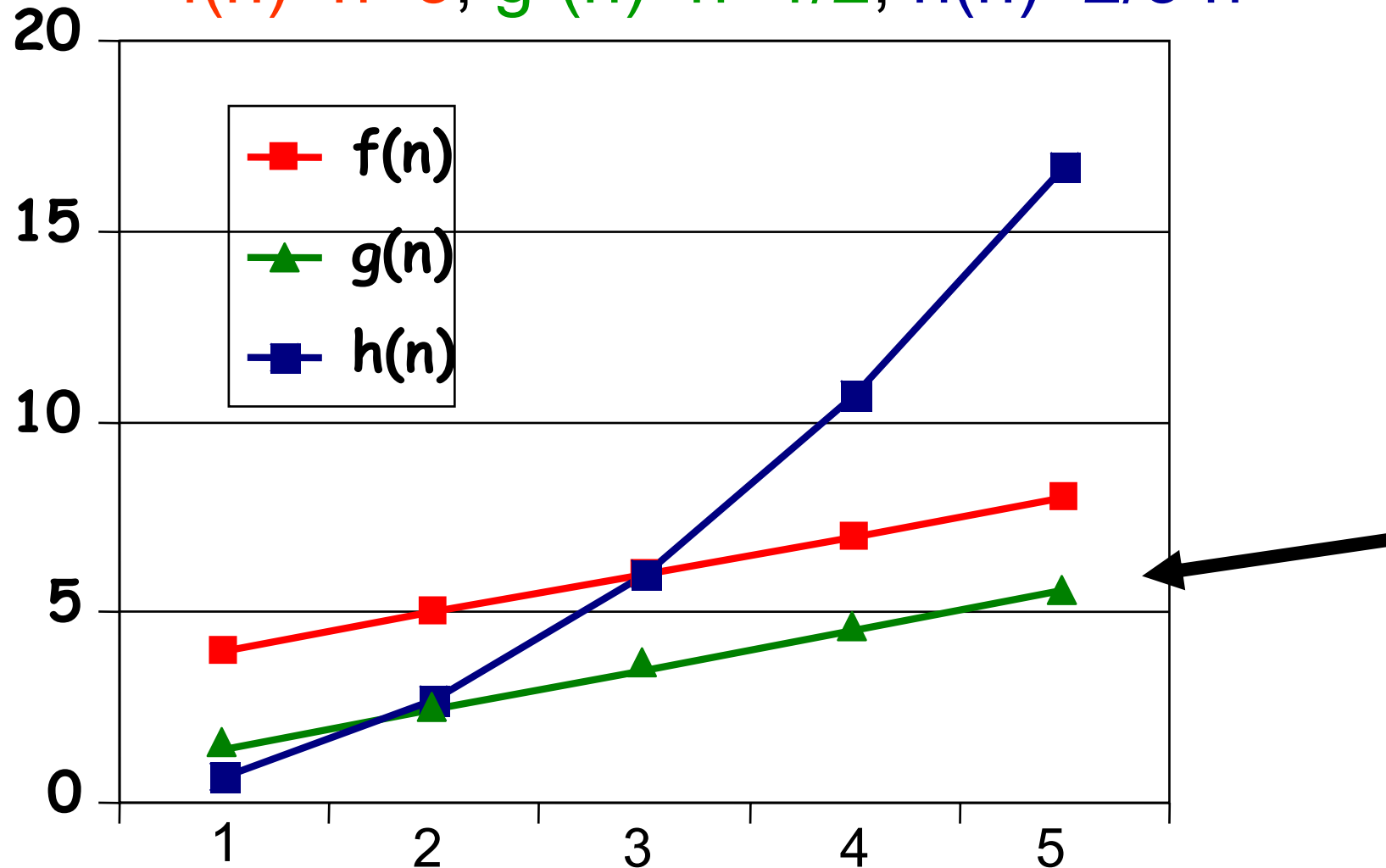
Algorithmen-Vergleich

$$f(n)=n+3, \quad g(n)=2n+1, \quad h(n)=\frac{2}{3} n^2$$



G kauft sich einen neuen Rechner:

$$f(n)=n+3, \quad g'(n)=n+1/2, \quad h(n)=2/3 n^2$$



Laufzeitvergleich

- Effizienz eines Algorithmus darf nicht vom Rechner abhängen!

- $f(n)$ und $g(n)$ sind ungefähr gleich gut bzgl. Effizienz

- $h(n)$ ist zwar für kleine n besser als $f(n)$, aber uns interessieren „große“ n

- „ $f(n)$ wächst ungefähr gleich wie $g(n)$ “
„ $f(n)$ wächst langsamer als $h(n)$ “

Asymptotische Laufzeit: O-Notation

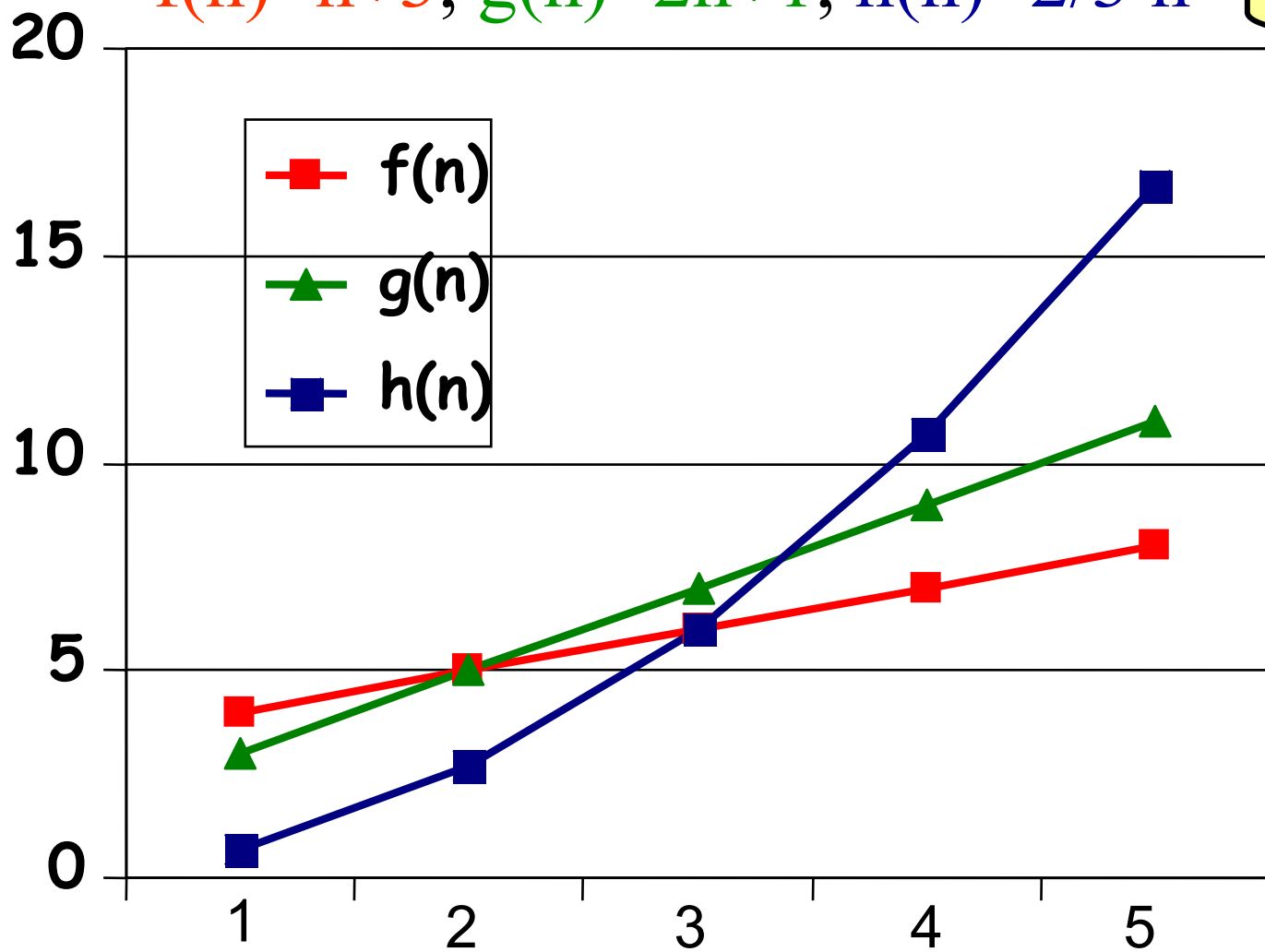
- $O(g(n))$ ist die Menge aller positiven Funktionen $f(n)$, für die ein c und ein n_0 (beide positiv und unabhängig von n) existieren, so dass für alle $n \geq n_0$ gilt:
 $f(n) \leq cg(n)$.

$$O(g(n)) = \{f(n) : N \rightarrow R^+ \mid (\exists c, n_0 > 0), (\forall n \geq n_0) : f(n) \leq cg(n)\}$$

- „ $cg(n)$ ist obere Schranke für $f(n)$ für große n “

$$f(n)=n+3, g(n)=2n+1, h(n)=\frac{2}{3} n^2$$

„=“ statt „∈“



~~f(n)=O(n)
g(n)=O(n)
h(n)=O(n)~~

Beweistechnik für O-Notation

Beispiel: Wir zeigen: $f(n)=n+3=O(n) \Rightarrow g(n)=n$

- Wir müssen also ein c und ein n_0 (beide positiv und unabhängig von n) finden, so dass für alle $n \geq n_0$ gilt: $f(n) \leq cn$

- Es muß gelten: $n+3 \leq cn$.
- Die gilt genau dann, wenn $1+3/n \leq c$.
- Dies ist z.B. erfüllt für $n \geq 3$ und $c=2$. Oder auch $n \geq 100$ und $c=1,5$.

Tipp: Es gibt beliebig viele Möglichkeiten. Wir müssen für den Beweis eine davon angeben.

Beweistechnik für $O\Omega\Theta$ -Notation

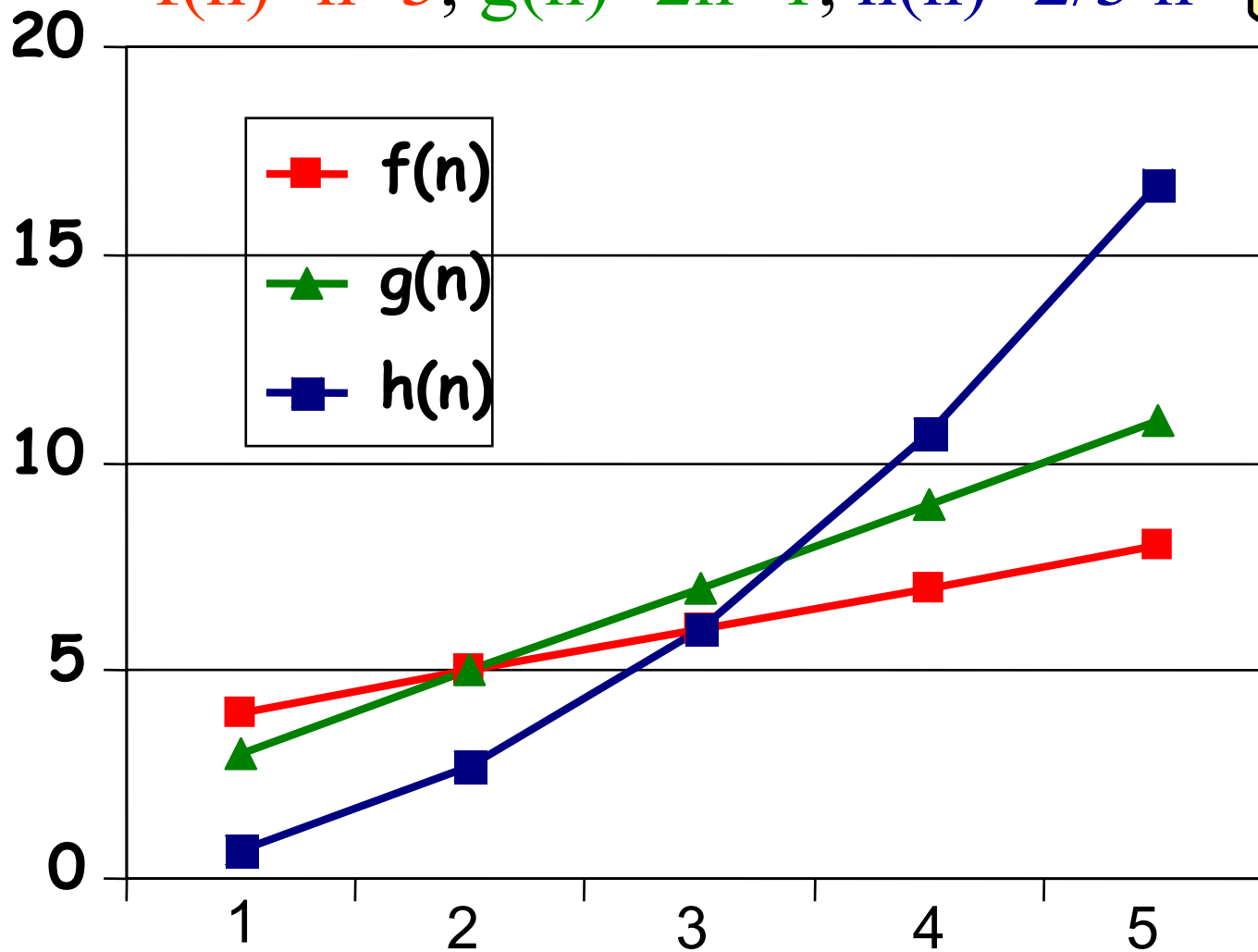
Beispiel: Wir zeigen: $h(n)=2/3n^2 \neq O(n) \Rightarrow g(n)=n$

- Wir zeigen, dass kein c und n_0 (positiv und unabhängig von n) existiert, so dass für alle $n \geq n_0$ gilt: $h(n) \leq cn$

- Sonst würde gelten: $2/3n^2 \leq cn$ für alle $n \geq n_0$.
- Die gilt g.d.w. $2/3n \leq c$ für alle $n \geq n_0$.
- Für „ n gegen unendlich“ wird es nie eine konstante Zahl c (unabhängig von n) geben, die dies erfüllt. Dies ist der Widerspruch.

$$f(n)=n+3, g(n)=2n+1, h(n)=\frac{2}{3} n^2$$

„=“ statt „∈“



$$f(n)=O(g(n))$$

$$g(n)=O(f(n))$$

$$f(n)=O(h(n))$$

~~$$h(n)=O(f(n))$$~~

$$g(n)=O(h(n))$$

~~$$h(n)=O(g(n))$$~~

$$g(n)=O(f(n)) \text{ und } f(n)=O(h(n)) \Rightarrow g(n)=O(h(n))$$

Asymptotische Laufzeit: Ω -Notation

- $\Omega(g(n))$ ist die Menge aller positiven Funktionen $f(n)$, für die ein c und ein n_0 (beide positiv und unabhängig von n) existieren, so dass für alle $n \geq n_0$ gilt: $f(n) \geq cg(n)$.

$$\Omega(g(n)) = \{f(n) : N \rightarrow R^+ \mid (\exists c, n_0 > 0), (\forall n \geq n_0) : f(n) \geq cg(n)\}$$

- „ $cg(n)$ ist untere Schranke für $f(n)$ für große n “

Asymptotische Laufzeit: Θ -Notation

- $\Theta(g(n))$ ist die Menge aller positiven Funktionen $f(n)$, für die ein c_1 und c_2 und ein n_0 (alle positiv und unabhängig von n) existieren, so dass für alle $n \geq n_0$ gilt: $c_1 g(n) \leq f(n) \leq c_2 g(n)$.

$$\Theta(g(n)) = \{f(n) : N \rightarrow R^+ \mid (\exists c_1, c_2, n_0 > 0), (\forall n \geq n_0) : c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

- „ $\Theta(g(n))$ ist die Menge aller Funktionen, die ungefähr gleich stark wachsen wie $g(n)$ “

Beweistechnik für $O\Omega\Theta$ -Notation

Beispiel: Wir zeigen: $f(n)=n+3=\Theta(n) \Rightarrow g(n)=n$

- Wir müssen also je ein c_1, c_2 und ein n_0 (positiv und unabhängig von n) finden, so dass für alle $n \geq n_0$ gilt: $c_1 n \leq f(n) \leq c_2 n$
- Es muß gelten: $c_1 n \leq n+3 \leq c_2 n$.
- Die gilt genau dann, wenn $c_1 \leq 1+3/n \leq c_2$.
- Dies ist z.B. erfüllt für $n \geq 3$, $c_1=1$, $c_2=2$. Oder auch $n \geq 100$ und $c_1=0,5$ und $c_2=1,2$.

Tipp: Es gibt beliebig viele Möglichkeiten. Wir müssen für den Beweis eine davon angeben.

Tipp: Üben Sie mehr davon! Klausurverdächtig!!!

Beispiel: Wir zeigen: WorstCase Laufzeit von InsertionSort ist in $\Theta(n^2)$.

- Es muß gelten: $c_1 n^2 \leq an^2 + bn + c \leq c_2 n^2$.
- Die gilt g.d.w. $c_1 \leq a + b/n + c/n^2 \leq c_2$ für große n .
- Wir wissen, dass $a > 0$.
- Sei $T = b/n + c/n^2$. Wir können n_0 so wählen, dass $a > 2|T|$ gilt für alle $n \geq n_0$.
- Dann ist z.B. erfüllt für $c_1 = a/2$ und $c_2 = 2a$.
- Denn: $a/2 > |T| \Rightarrow a + T \geq a/2$ (für $T < 0$ und $T \geq 0$)
und $a + T \leq 2a$. Die c_i sind positive Konstanten.

Eigenschaften der $O\Omega\Theta$ -Notation

- Aus der Worst-Case Laufzeit $\Theta(n^2)$ folgt eine Laufzeit von $O(n^2)$ für beliebige Eingaben.

- Aus der Worst-Case Laufzeit $\Theta(n^2)$ folgt NICHT eine Laufzeit von $\Theta(n^2)$ für beliebige Eingaben.

- Aus der Best-Case Laufzeit $\Theta(n)$ folgt eine Laufzeit von $\Omega(n)$ für beliebige Eingaben.

$O\Omega\Theta$ -Notation in Gleichungen

- Manchmal sieht man auch die O -Notation als Teilterme in Gleichungen, wie z.B. $2n^2+3n+1=2n^2+\Theta(n)$.

- Dies bedeutet: Es gibt eine Funktion $f(n)\in\Theta(n)$ mit $2n^2+3n+1=2n^2+f(n)$.
- Man schreibt dies, um zu verdeutlichen, dass die Funktion im wesentlichen mit $2n^2$ wächst, zu dem „nur“ ein linear wachsender Anteil hinzukommt.

Weitere asymptotische Schranken

- $o(g(n))$ ist die Menge aller positiven Funktionen $f(n)$, für die der Limes von $f(n)/g(n)$ für „ n gegen unendlich“ gegen 0 geht.

$$o(g(n)) = \{f(n) : N \rightarrow R^+ \mid \lim_{n \rightarrow \infty} f(n)/g(n) = 0\}$$

Äquivalent hierzu ist:

$$o(g(n)) = \{f(n) : N \rightarrow R^+ \mid (\forall c > 0), (\exists n_0 > 0), (\forall n \geq n_0) : f(n) < cg(n)\}$$

- „ $o(g(n))$ umfaßt alle Funktionen, die echt schwächer wachsen als $g(n)$ selbst“.

Weitere asymptotische Schranken

$\omega(g(n))$ ist die Menge aller positiven Funktionen $f(n)$, für die der Limes von $f(n)/g(n)$ für „ n gegen unendlich“ gegen ∞ geht.

$$\omega(g(n)) = \{f(n) : N \rightarrow R^+ \mid \lim_{n \rightarrow \infty} f(n)/g(n) = \infty\}$$

Äquivalent hierzu ist:

$$\omega(g(n)) = \{f(n) : N \rightarrow R^+ \mid (\forall c > 0), (\exists n_0 > 0), (\forall n \geq n_0) : f(n) > cg(n)\}$$

„ $\omega(g(n))$ umfaßt alle Funktionen, die echt stärker wachsen als $g(n)$ selbst.“

Eigenschaften der $O\Omega\Theta$ -Notation

- $O(g(n)) = \Theta(g(n)) \cup o(g(n))$
- $\Theta(g(n)) \cap o(g(n)) = \emptyset$
- $\Omega(g(n)) = \Theta(g(n)) \cup \omega(g(n))$
- $\Theta(g(n)) \cap \omega(g(n)) = \emptyset$
- $f(n)=O(g(n)) \Leftrightarrow g(n)=\Omega(f(n))$
- $f(n)=o(g(n)) \Leftrightarrow g(n)=\omega(f(n))$

- **Tipp: Beweisen Sie diese Aussagen!
Übungstest- und Klausurverdächtig!!!**