

# Datenstrukturen & Algorithmen

Peppo Brambilla  
Universität Bern  
Frühling 2018

# Übersicht

## Mathematische Hilfsmittel zur Komplexitätsanalyse

- Wachstum von Funktionen
- Rekursionsgleichungen

# Komplexitätsanalyse

- Wie aufwändig ist es, mit einem Computer ein gewisses Problem zu lösen?
- Brauchen einfaches mathematisches Modell, um Aufwand zu beschreiben
  - Zeitkomplexität (Laufzeit)
  - Platzkomplexität (Speicherbedarf)
- Beobachtung
  - Für die meisten Problem wird Aufwand hauptsächlich durch Grösse der Eingabe bestimmt
  - Spezifische Eigenschaften eines Computers spielen geringere Rolle, vor allem ab gewisser Grösse der Eingabe
- Konsequenz
  - Mathematisches Modell soll nur Größenordnung des Aufwands beschreiben
  - Soll Computereigenschaften ignorieren

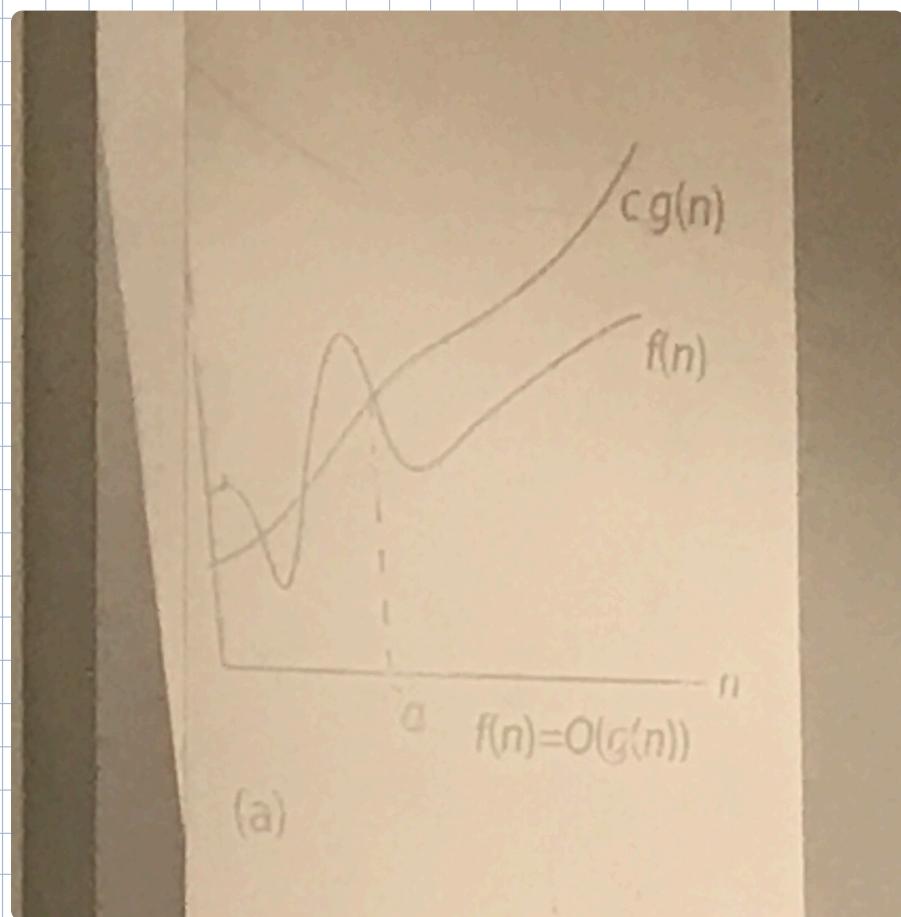
# Asymptotische Notation

- Abschätzen des Wachstumsverhaltens von Funktionen  $f(n)$  für grosse  $n$ 
  - Unsere Anwendung: Laufzeit  $T(n)$  für grosse  $n$
- Präzisiere Aussagen wie
  - Gegeben Funktionen  $f(n) = \frac{1+n^2}{n}$ ,  $g(n) = n$
  - „Für grosse  $n$  sind  $f(n)$  und  $g(n)$  ungefähr gleich gross“
- Landau Notation, „gross-O Notation“

# O-Notation (obere Schranke)

Wir sagen  $f(n) = O(g(n))$  wenn es Konstanten  $c > 0, n_0 > 0$  gibt, so dass  $0 \leq f(n) \leq c g(n)$  für alle  $n \geq n_0$

- „Ab einer gewissen Grösse von  $n$  ist  $f$  sicher kleiner oder gleich  $g$ “
- „ $g$  wächst schneller oder gleich wie  $f$ “
- Beispiel:  $2n^2 = O(n^3)$
- Überprüfe mit  $c = 1, n_0 = 2$
- Beachten: Ausdrücke mit  $n$  sind Funktion von  $n$ , nicht Werte



# Mengenschreibweise

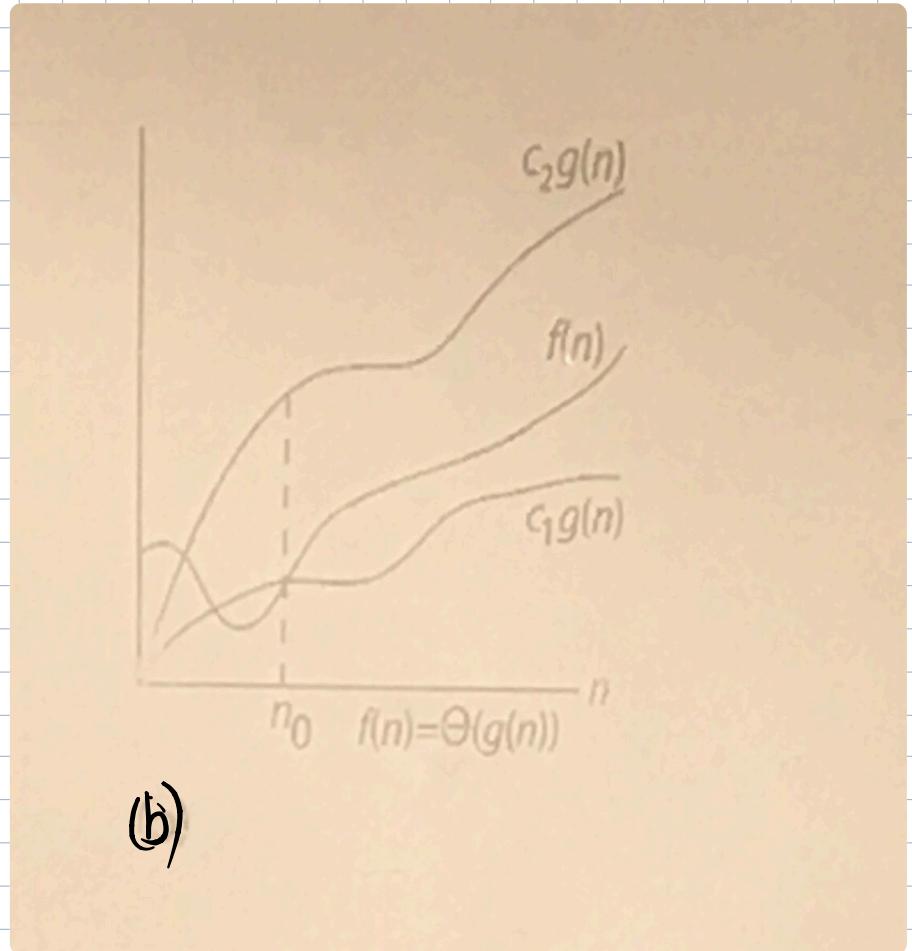
$O(g(n)) = \{f(n) : \text{es existieren positive Konstanten } c \text{ und } n_0, \text{ so dass } 0 \leq f(n) \leq c g(n) \text{ für alle } n \geq n_0\}$

- Ähnlich wie  $\Theta$  in der letzten Vorlesung
- Gleichheitszeichen in  $f(n) = O(g(n))$  ist als “ $f(n)$  ist Teil der Menge  $O(g(n))$ “ zu verstehen
- $O$ -Notation ist obere Schranke  
 $f(n) = O(g(n))$  bedeutet  $g$  ist obere Schranke für  $f$

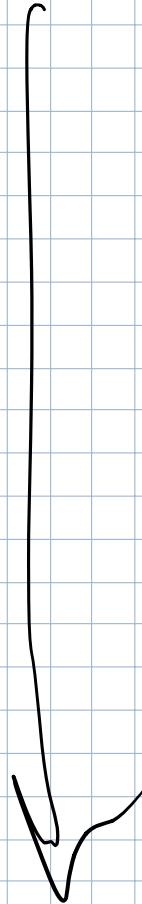
# $\Omega$ -Notation (untere Schranke)

$\Omega(g(n)) = \{f(n): \text{es existieren positive Konstanten } c \text{ und } n_0, \text{ so dass } 0 \leq c g(n) \leq f(n) \text{ für alle } n \geq n_0\}$

- „Ab einer gewissen Grösse von  $n$  ist  $f$  sicher grösser oder gleich  $g$ “
- „ $f$  wächst schneller oder gleich wie  $g$ “
- $\Omega$  Notation ist untere Schranke  
 $f(n) = \Omega(g(n))$  bedeutet  $g$  ist untere Schranke für  $f$
- Beispiel:  $\sqrt{n} = \Omega(\lg n)$ , ( $c = 1, n_0 = 16$ )



(b)



# $\Theta$ -Notation (scharfe Schranke)

$$\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$$

- $f(n) = \Theta(g(n))$ :  $f$  wächst gleich schnell wie  $g$
- Beispiel  $\frac{1}{2}n^2 - 2n = \Theta(n^2)$

$$c_1 n^2 \leq \frac{1}{2} n^2 - 2n \leq c_2 n^2 \quad | : n^2$$

$$c_1 \leq \frac{1}{2} - \frac{2}{n} \leq c_2$$

- $c_1 = \frac{1}{4}$ : gültig für  $n_0 = 8$   $\xrightarrow{\text{größeres } n_0 \text{ überwiegt}}$
- $c_2 = \frac{1}{2}$ : gültig für  $n_0 = 1 \leq n$

$$f(n) = \frac{1}{2}n^2 - 2n$$

$$g(n) = n^2$$

$$c_1 g(n) \leq f(n) \leq c_2 \cdot g(n)$$

$$n \geq n_0$$

$$c_1 n^2 \leq \frac{1}{2}n^2 - 2n \leq c_2 n^2 \quad | : n^2$$

$$c_1 \leq \frac{1}{2} - \frac{2}{n} \leq c_2 \quad c_1 = \frac{1}{4} \Rightarrow n_0 = 8 \text{ Bed. erfüllt}$$

$$\frac{1}{4}$$

$$\frac{1}{2}$$

$$c_2 = \frac{1}{2} \Rightarrow n_0 = 1 \text{ Bed. erfüllt}$$

# $\Theta$ - und $\omega$ -Notation

- $\Theta$ - und  $\Omega$ -Notation entsprechen  $\leq$  und  $\geq$
- $\Theta$ - und  $\omega$ -Notation entsprechen  $<$  und  $>$

$\omega(g(n)) = \{f(n) : \text{für jede positive Konstante } c > 0 \text{ existiert eine Konstante } n_0 > 0, \text{ so dass } 0 \leq c g(n) < f(n) \text{ für alle } n \geq n_0\}$

- Beispiel:

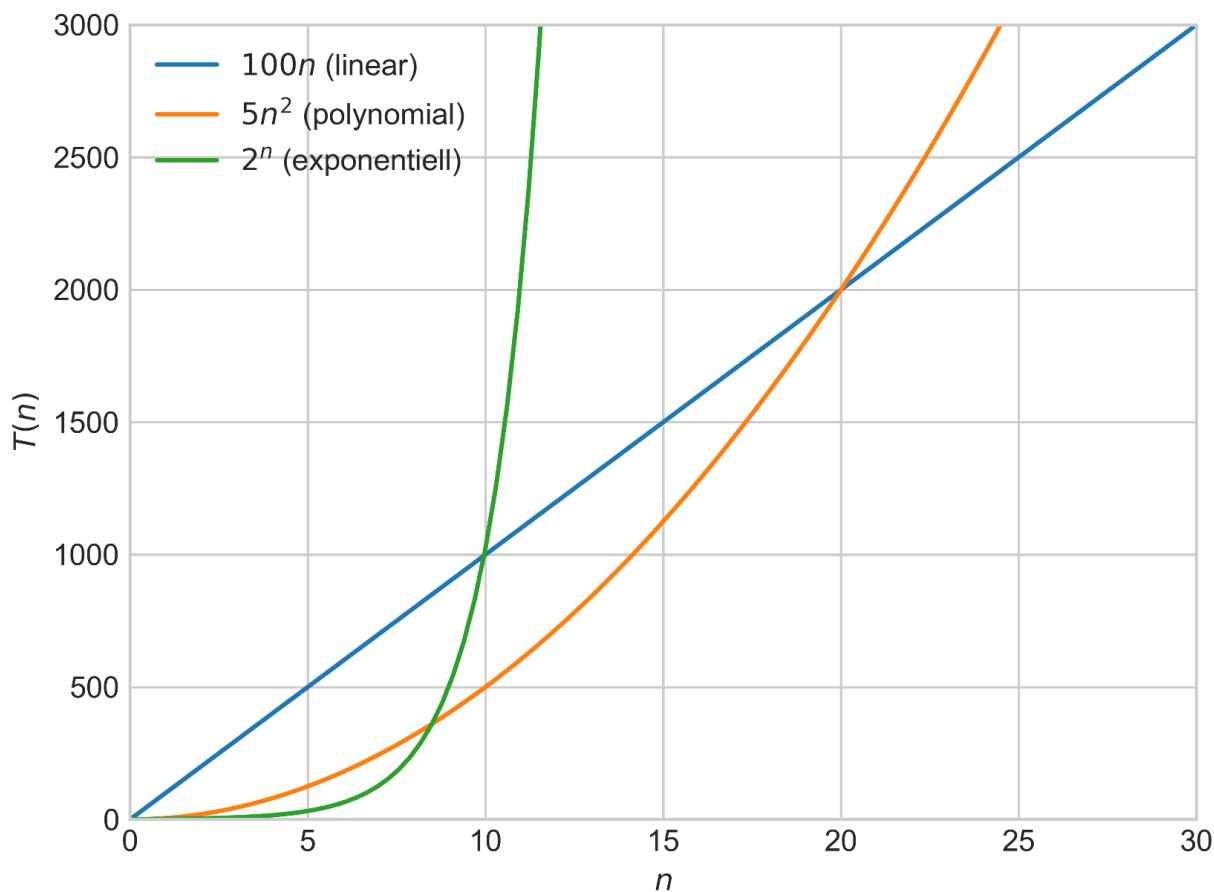
$$\sqrt{n} = \omega(\lg n), (n_0 = 1 + 1/c)$$

# Terminologie

- Eingabegrösse  $n$
- Zeitkomplexität  $T(n)$
- Komplexitätsklasse      Beispiel
  - Konstant  $T(n) = \Theta(1)$  (z.B. Addition)
  - Logarithmisch  $T(n) = \Theta(\lg n)$
  - Linear  $T(n) = \Theta(n)$
  - Polynomiel  $T(n) = \Theta(n^m)$ , ( $m \in \mathbb{N}$ , fest)
  - Exponentiell  $T(n) = \Theta(2^n)$
- Weitere siehe

[http://en.wikipedia.org/wiki/Time\\_complexity](http://en.wikipedia.org/wiki/Time_complexity)

# Vergleich



# Vergleich

- Zunahme der Rechenzeit bei Vergrösserung des Problems
- Annahme: Lösung des Problems der Grösse  $n = 1$  dauert bei allen Programmen  $10^{-6}$  s

Zeitkomplexität $T(n)$	$n = 20$	$n = 40$	$n = 60$
$n$	0.00002 s	0.00004 s	0.00006 s
$n \log n$	0.00009 s	0.0002 s	0.0004 s
$n^5$	3.2 s	1.7 m	13 m
$2^n$	1.0 s	12.7 d	366 a

s = Sekunden, m = Minuten, d = Tage, a = Jahre

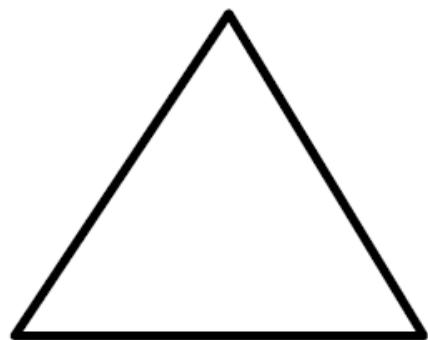
# Vergleich

- Zunahme der lösbaren Problemgrösse bei Verzehnfachung der Rechenzeit
- Annahme: eine Operation kostet 1 s

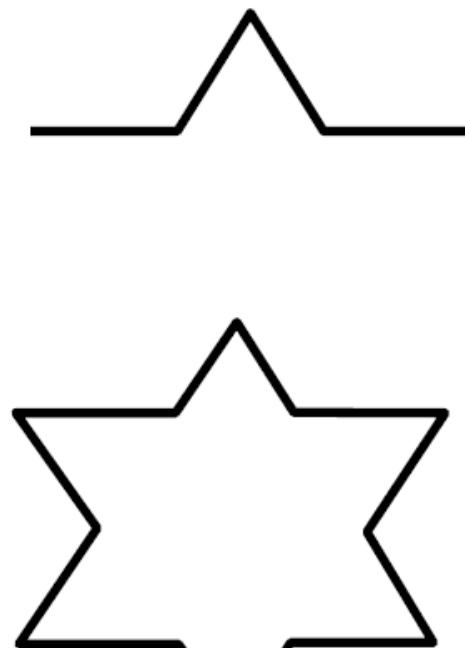
Zeitkomplexität $T(n)$	Problemgrösse lösbar in $10^3$ s	Problemgrösse lösbar in $10^4$ s	Zunahme um Faktor
$100n$	10	100	10.0
$5n^2$	14	45	3.2
$n^3/2$	12	27	2.3
$2^n$	10	13	1.3

# Beispiel

- Exponentielle Komplexität
- Kochkurve (Helge von Koch, 1870-1924)



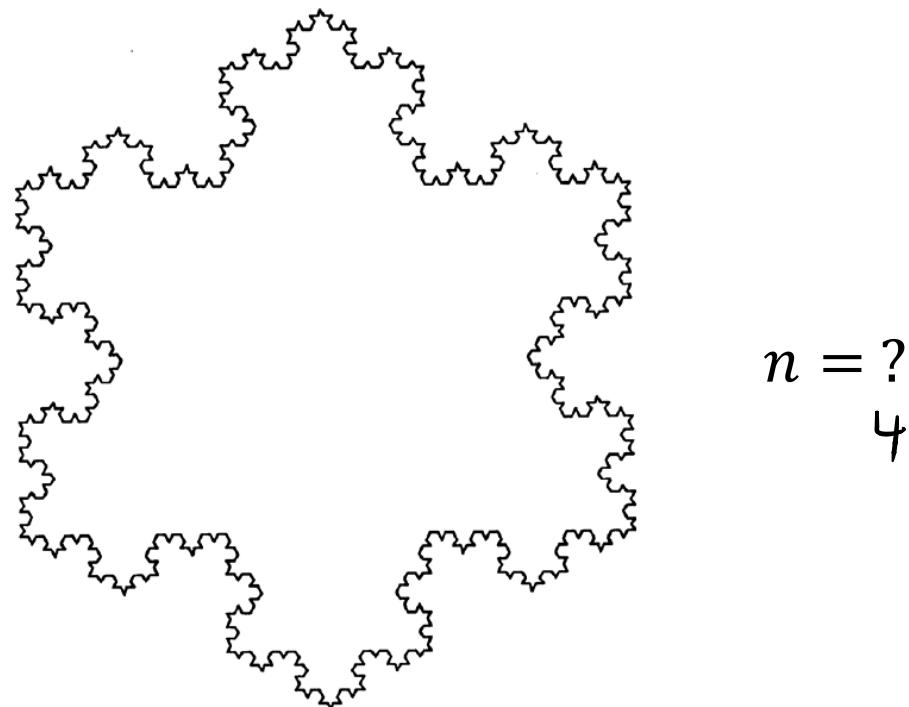
$n = 0$



$n = 1$

# Beispiel

- Exponentielle Komplexität
- Kochkurve (Helge von Koch, 1870-1924)

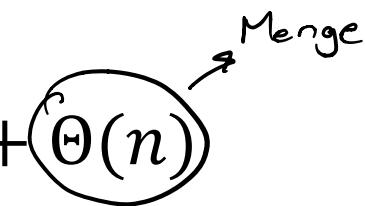


[http://en.wikipedia.org/wiki/Koch\\_snowflake](http://en.wikipedia.org/wiki/Koch_snowflake)

# Gleichungen mit Landau Notation

- Wie sollen Gleichungen dieser Form interpretiert werden?

$$2n^2 + 3n + 1 = 2n^2 + \Theta(n)$$

A diagram showing the term  $\Theta(n)$  enclosed in a circle. An arrow points from the word "Menge" to the circle.

- „Es gibt eine **anonyme** Funktion in  $\Theta(n)$ , so dass die Gleichung erfüllt ist“
  - „Anonyme Funktion in  $\Theta(n)$ “ bedeutet „irgendeine Funktion in  $\Theta(n)$ “
- Es ist erlaubt, eine Menge (d.h. Landau Symbole  $\Theta$ ,  $O$ ,  $\Omega$ ) in einer Formel durch eine beliebige Funktion aus dieser Menge zu ersetzen

# Gleichungen mit Landau Notation

- Beispiel:

$$f(n) = n^3 + O(n^2)$$

heisst

$$f(n) = n^3 + h(n), \text{ wobei } h(n) \in O(n^2)$$

zum Beispiel

$$h(n) = n^2, \text{ oder } h(n) = 3n^2 + n$$

- Anonyme Funktion  $h(n)$  wird üblicherweise nicht explizit aufgeschrieben

# Gleichungen mit Landau Notation

- „Unabhängig wie die anonyme Funktion auf der linken Seite gewählt wird, kann die anonyme Funktion auf der rechten Seite so gewählt werden, dass die Gleichung erfüllt wird“
- Beispiel:  $n^2 + O(n) = O(n^2)$  heisst

für ein beliebiges  $f(n) \in O(n)$  gilt  
 $n^2 + f(n) = h(n)$ , wobei  $h(n) \in O(n^2)$

# Übersicht

## Mathematische Hilfsmittel zur Komplexitätsanalyse

- Wachstum von Funktionen
- Rekursionsgleichungen

# Rekursionsgleichungen

- Komplexität von Algorithmen, die sich selbst aufrufen (rekursive Algorithmen), kann als Rekursionsgleichung dargestellt werden
- Rekursionsgleichung: Gleichung, die Funktion durch eigene Funktionswerte für kleinere Eingaben beschreibt
- Analyse von Mischen durch Sortieren aus der ersten Vorlesung

$$T(n) = f(x) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2T\left(\frac{n}{2}\right) & \text{if } n > 1 \end{cases}$$

# Rekursionsgleichungen

- Analyse der Zeitkomplexität erfordert Lösung von Rekursionsgleichung
- Lösung mit Hilfe einiger Tricks und Rechenregeln (ähnlich wie Differentiation, Integration)
- 3 Tricks
  - Substitutionsmethode (*Raten*)
  - Rekursionsbaum-Methode
  - Mastermethode

# Substitutionsmethode

- Allgemein anwendbar
1. Erraten der Form der Lösung
  2. Verifizieren durch Induktion
  3. Bestimmung der Konstanten

# Beispiel

$$T(n) = 4T(n/2) + n$$

Annahme: Induktionsbasis  $T(1) = \Theta(1)$

1. Errate:  $T(n) = O(n^3)$

2. Induktion

- Induktionsannahme:  $T(k) \leq ck^3$  für  $k < n$
- Das heisst:  $T(n/2) \leq c(n/2)^3$
- Beweise:  $T(n) \leq cn^3$

Annahme Induktionsbasis :  $T(1) = \Theta(1)$

1. Errate:  $T(n) = \Theta(n^3)$

Induktionsannahme  $T(k) \leq c \cdot k^3$  für  $k < n$

d.h.  $T\left(\frac{n}{2}\right) \leq c \cdot \left(\frac{n}{2}\right)^3$  ①

Beweise  $T(n) \leq cn^3$

$$T(n) = 4 \cdot T\left(\frac{n}{2}\right) + n$$

$$\stackrel{\textcircled{1}}{\leq} 4c \left(\frac{n}{2}\right)^3 + n$$

$$= \frac{4cn^3}{8} + n$$

$$= \frac{c}{2}n^3 + n$$

$$= cn^3 - \left(\frac{cn^3}{2} - n\right) \leq cn^3 \quad \text{falls } \frac{cn^3}{2} - n \geq 0$$

z.B.  $c=2, n \geq 1$

# Beispiel

- Einsetzen der Annahme  $T(n/2) \leq c(n/2)^3$

$$T(n) = 4T(n/2) + n$$

$$\leq 4c(n/2)^3 + n$$

$$= \frac{4cn^3}{8}$$

$$+ n$$

$$= \frac{c}{2}n^3 + n$$

$$= cn^3 - \left(\frac{c}{2}n^3 - n\right)$$

$$\leq cn^3$$

$$\text{falls } \frac{c}{2}n^3 - n \geq 0$$

$$\text{z.B. } c \geq 2, n \geq 1$$

# Beispiel (Fortsetzung)

- Müssen auch Randbedingungen berücksichtigen
  - „Verankerung“ der Induktion
  - **Induktionsanfang:** Zeige für ein fixes  $n$ , dass Ungleichung erfüllt ist
- Im Beispiel: Induktionsbasis  $T(1) = \Theta(1)$ 
  - Wähle  $n = 1$  als Induktionsanfang
  - Es gilt „ $\Theta(1)$ “  $\leq c \cdot 1^3$  wenn wir  $c$  genügend gross wählen
- Aber: müssen **nicht unbedingt** Induktionsbasis als Induktionsanfang verwenden (siehe Buch, Seiten 85f)

# Schärfere Grenze

- Beweisen, dass sogar  $T(n) = O(n^2)$
- Induktionsannahme:  $T(k) \leq ck^2$  für  $k < n$
- Induktion: substituiere  $T(n/2) \leq c(n/2)^2$

$$\begin{aligned} T(n) &= 4T(n/2) + n \\ &\leq 4c(n/2)^2 \\ &= cn^2 + n \\ &= cn^2 - (-n) \\ &\leq cn^2 \end{aligned}$$

Term  $(-n)$  kann nicht positiv werden →  
Beweis gescheitert

Annahme Induktionsbasis :  $T(1) = \Theta(1)$

1. Errate:  $T(n) = \Theta(n^2)$

Induktionsannahme  $T(k) \leq c_1 k^2 - c_2 k$  für  $k < n$

$$T\left(\frac{n}{2}\right) \leq c_1 \left(\frac{n}{2}\right)^2 - c_2 \frac{n}{2}$$

$$\begin{aligned} T(n) &= 4 T\left(\frac{n}{2}\right) + n \\ &\leq 4 c_1 \frac{n^2}{4} + n \\ &= c_1 n^2 + n \\ &= c_1 n^2 - \underbrace{(-n)}_{\text{Residuum / wir kriegen es nicht negativ hin}} \end{aligned}$$

Residuum / wir kriegen es nicht negativ hin

$$\begin{aligned} T(n) &= \dots \\ &\leq 4 \left(c_1 \frac{n^2}{4} - c_2 \frac{n}{2}\right) + n \\ &= c_1 n^2 - 2c_2 n + n \\ &= c_1 n^2 - c_2 n - \underbrace{(c_2 n - n)}_{c_2 \geq 1} \\ &\leq c_1 n^2 - c_2 n \end{aligned}$$

# Schärfere Grenze

- Idee: stärkere Induktionsannahme

$$T(k) \leq c_1 k^2 - c_2 k \text{ für } k < n$$

$$\begin{aligned} T(n) &= 4T\left(\frac{n}{2}\right) + n \\ &\leq 4\left(c_1\left(\frac{n}{2}\right)^2 - c_2\frac{n}{2}\right) + n \\ &= c_1n^2 - 2c_2n + n \\ &= c_1n^2 - c_2n - (c_2n - n) \\ &\leq c_1n^2 - c_2n \end{aligned} \quad \begin{array}{l} \text{wenn } c_2n - n \geq 0 \\ \text{d.h. } c_2 \geq 1 \end{array}$$

Wähle  $c_1$  gross genug, um Randbedingungen zu erfüllen

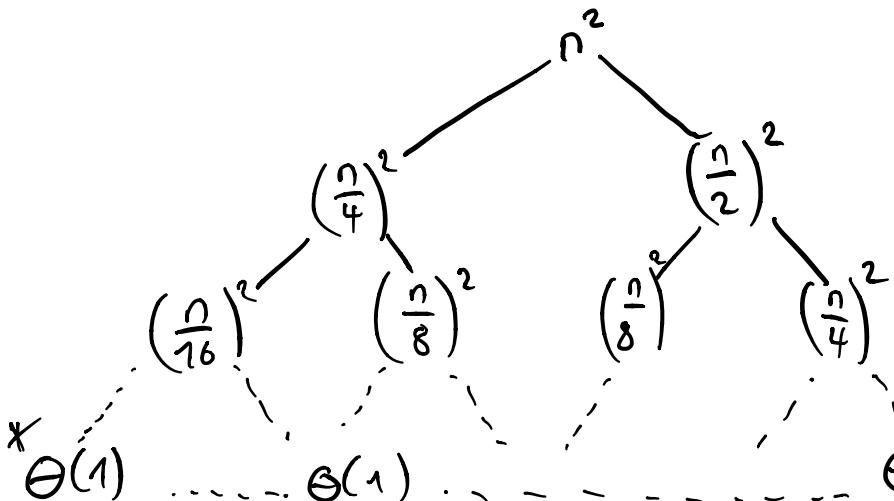
# Rekursionsbaum-Methode

- „grafisches Modell“ für Laufzeit rekursiver Algorithmen
- Gut für Intuition
- Kann als Ausgangslage für Substitutionsmethode verwendet werden
  - Erraten der Lösung

# Rekursionsbaum

- Löse  $T(n) = T(n/4) + T(n/2) + n^2$

$$T(n/16) + T(n/8) + \frac{n^2}{4} + T(n/8) + T(n/4) + \left(\frac{n}{2}\right)^2$$



Rek. Tiefe	* - Terme	$\sum$
0	$2^0 = 1$	$n^2$
1	$2^1 = 2$	$\frac{5}{16} n^2$
2	$2^2 = 4$	$\frac{25}{256} n^2$
$\vdots$	$\vdots$	$\vdots$
$2^{\lg_2(n)}$	$n$	$\Theta(n^2)$

\*  $n \cdot \Theta(1) = \Theta(n)$

$$\text{Summe} = n^2 \cdot \left(1 + \frac{5}{16} + \frac{25}{256} + \dots\right) = \Theta(n^2)$$

=> Komplexität von  $\Theta(n^2) + \Theta(n)$   
 $= \Theta(n^2)$

# Mastermethode

- „Rezept“ zur Lösung von Rekursionsgleichungen der Form

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

a Teilprobleme

Wobei

$\rightarrow$  Schwierigkeit muss immer kleiner werden

$a \geq 1, b > 1, f$  asymptotisch positiv

- Ohne Beweis hier

# Drei Fälle

- Vergleiche  $f(n)$  mit  $n^{\log_b(a)}$
1. Falls  $f(n) = O(n^{\log_b a - \varepsilon})$  für eine Konstante  $\varepsilon > 0$ 
    - $f(n)$  wächst polynomial langsamer als  $n^{\log_b a}$
    - Und zwar um den Faktor  $n^\varepsilon$
    - Zur Erinnerung:  $n^{\log_b a - \varepsilon} = \frac{n^{\log_b a}}{n^\varepsilon}$

**Lösung:**  $T(n) = \Theta(n^{\log_b a})$

# Drei Fälle

- Vergleiche  $f(n)$  mit  $n^{\log_b a}$

2. Falls  $f(n) = \Theta(n^{\log_b a})$

- $f(n)$  wächst gleich schnell wie  $n^{\log_b a}$

Lösung:  $T(n) = \Theta(n^{\log_b a} \lg n)$

Logarithmus  
zur Basis  $b$

Binärer Logarithmus  
mit Basis 2

(Achtung:  $\lg$  bezeichnet anderswo  
auch Logarithmus Basis 10)

# Drei Fälle

- Vergleiche  $f(n)$  mit  $n^{\log_b a}$
- 3. Falls  $f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right)$  für eine Konstante  $\varepsilon > 0$  und  $f(n)$  erfüllt  $a f\left(\frac{n}{b}\right) \leq c f(n)$  für eine Konstante  $c < 1$ 
  - $f(n)$  wächst polynomial schneller als  $n^{\log_b a}$
  - Und zwar um den Faktor  $n^\varepsilon$
  - Zur Erinnerung:  $n^{\log_b a + \varepsilon} = n^{\log_b a} n^\varepsilon$

Lösung:  $T(n) = \Theta(f(n))$

# Zusammenfassung

- Problem: löse  $T(n) = aT(n/b) + f(n)$
  - Vergleiche  $f(n)$  mit  $n^{\log_b a}$
1. Falls  $f(n) = O(n^{\log_b a - \varepsilon})$  für eine Konstante  $\varepsilon > 0$   
Lösung:  $T(n) = \Theta(n^{\log_b a})$
  2. Falls  $f(n) = \Theta(n^{\log_b a})$   
Lösung:  $T(n) = \Theta(n^{\log_b a} \lg n)$
  3. Falls  $f(n) = \Omega(n^{\log_b a + \varepsilon})$  für eine Konstante  $\varepsilon > 0$   
und  $f(n)$  erfüllt  $af(n/b) \leq cf(n)$  für Konstante  $c < 1$   
Lösung:  $T(n) = \Theta(f(n))$

# Beispiele

$$T(n) = 4T(n/2) + \boxed{n}$$

$$a = 4 \quad b = 2 \quad n^{\log_b a} = n^{\lg 4} = n^2$$

$$f(n) = n$$

$$n = \Theta(n^{2-\varepsilon}) \text{ falls } 0 < \varepsilon < 1 \Rightarrow T(n) = \Theta(n^2)$$

Fall 1:

$$f(n) = O(n^{2-\varepsilon}) \text{ für } 0 < \varepsilon \leq 1$$

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$$

# Beispiele

$$T(n) = 4T(n/2) + \boxed{n^2}$$

$$a = 4 \quad b = 2 \quad n^{\log_b a} = n^{\lg 4} = n^2$$

$$f(n) = n^2$$

$n^2 = \Theta(n^2)$  falls  $0 < \varepsilon < 1 \Rightarrow T(n) = \Theta(n^2 \cdot \lg(n))$   
Fall 2:

$$f(n) = \Theta(n^2)$$

$$T(n) = \Theta(n^{\log_b a} \cdot \log n) = \Theta(n^2 \cdot \log n)$$

# Beispiele

$$T(n) = 4T(n/2) + \boxed{n^3}$$

$$a = 4 \quad b = 2 \quad n^{\log_b a} = n^{\lg 4} = n^2$$

$$f(n) = n^3$$

$$n^3 = \Theta(n^{2+\varepsilon}) \text{ falls } 0 < \varepsilon < 1 \Rightarrow T(n) = \Theta(f(n)) = \Theta(n^3)$$

Fall 3:

$$f(n) = \Omega(n^{2+\varepsilon}) \text{ für } 0 < \varepsilon \leq 1 \text{ und}$$

$$4f\left(\frac{n}{2}\right) \leq cf(n) \Rightarrow 4\left(\frac{n}{2}\right)^3 \leq cn^3 \Rightarrow \frac{1}{2}n^3 \leq cn^3$$

$$\text{für } \frac{1}{2} \leq c < 1$$

$$T(n) = \Theta(f(n)) = \Theta(n^3)$$

# Beispiele (welches nicht funktioniert)

$$T(n) = 4T(n/2) + \boxed{n^2/\lg n}$$

$$a = 4 \quad b = 2 \quad n^{\log_b a} = n^{\lg 4} = n^2$$
$$f(n) = n^2/\lg n$$

$$f(n) \neq \Theta(n^2) \text{ (Fall 2)}$$

$$f(n) \neq \Omega(n^{2+\varepsilon}) \text{ (Fall 3)}$$

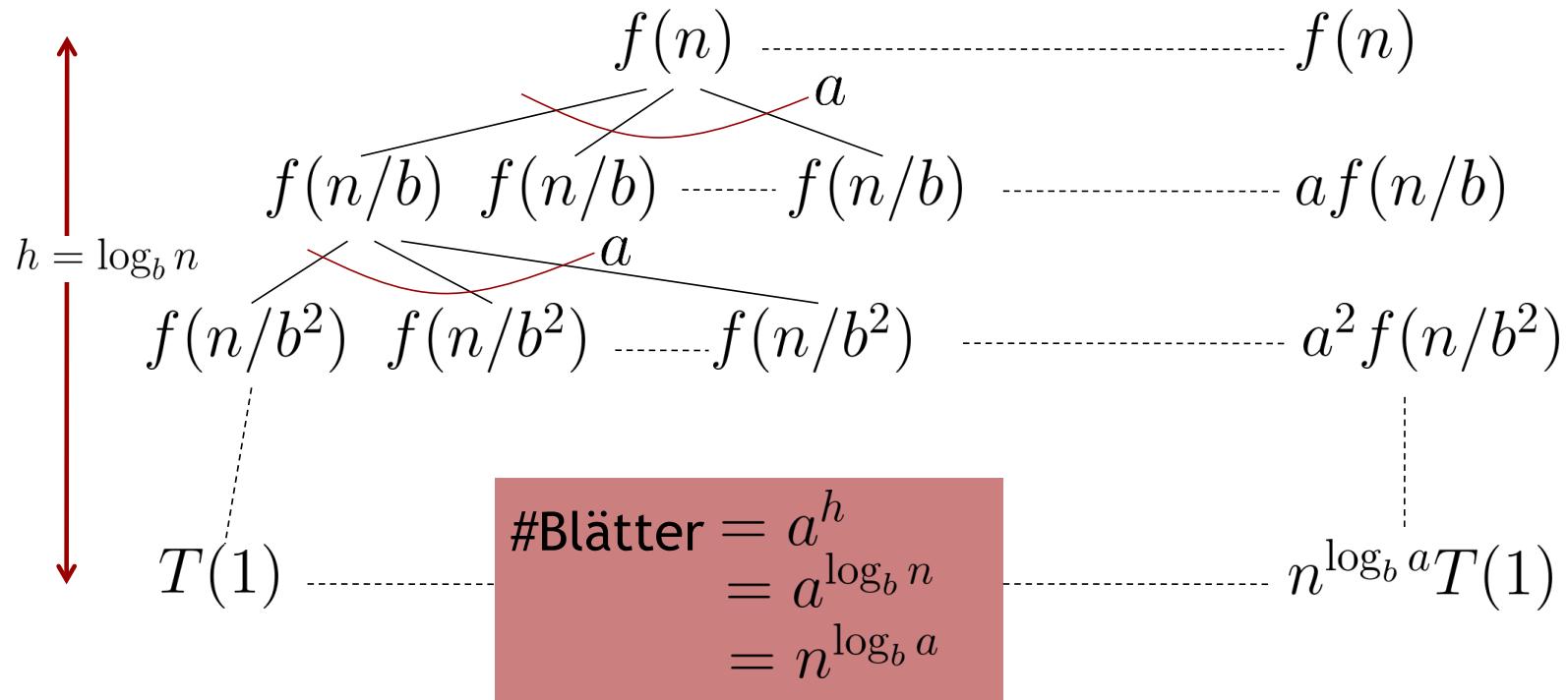
$f(n) = O(n^2)$  aber  $f(n) \neq O(n^{2-\varepsilon})$  für beliebiges  $\varepsilon > 0$

Grund:  $\lg n$  ist asymptotisch kleiner als  $n^\varepsilon$  für  $\varepsilon > 0$   
 $\Rightarrow f(n)$  wächst nicht polynomial langsamer als  $n^2$ .

Mastertheorem ist nicht anwendbar!

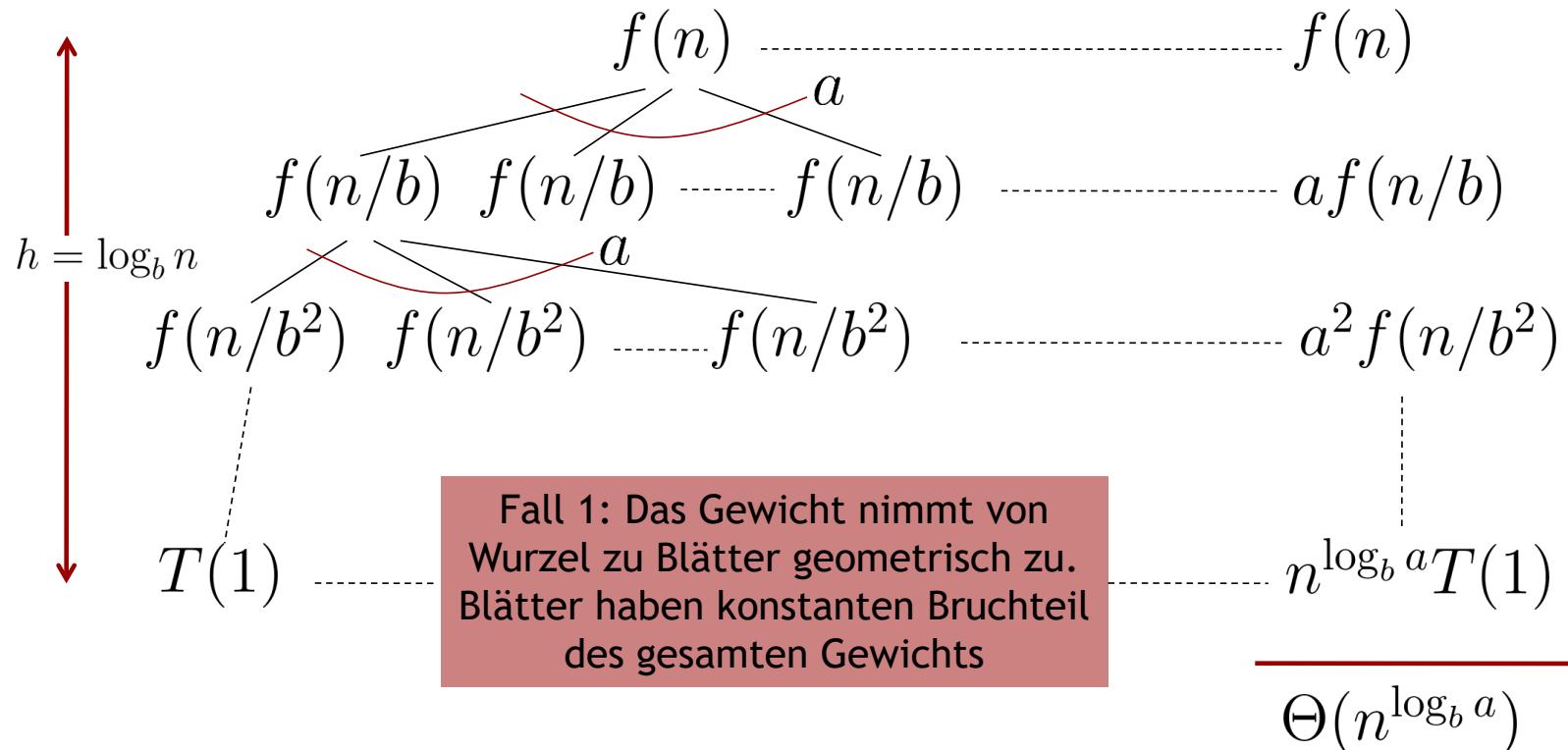
# Intuition für das Mastertheorem

- Rekursionsbaum



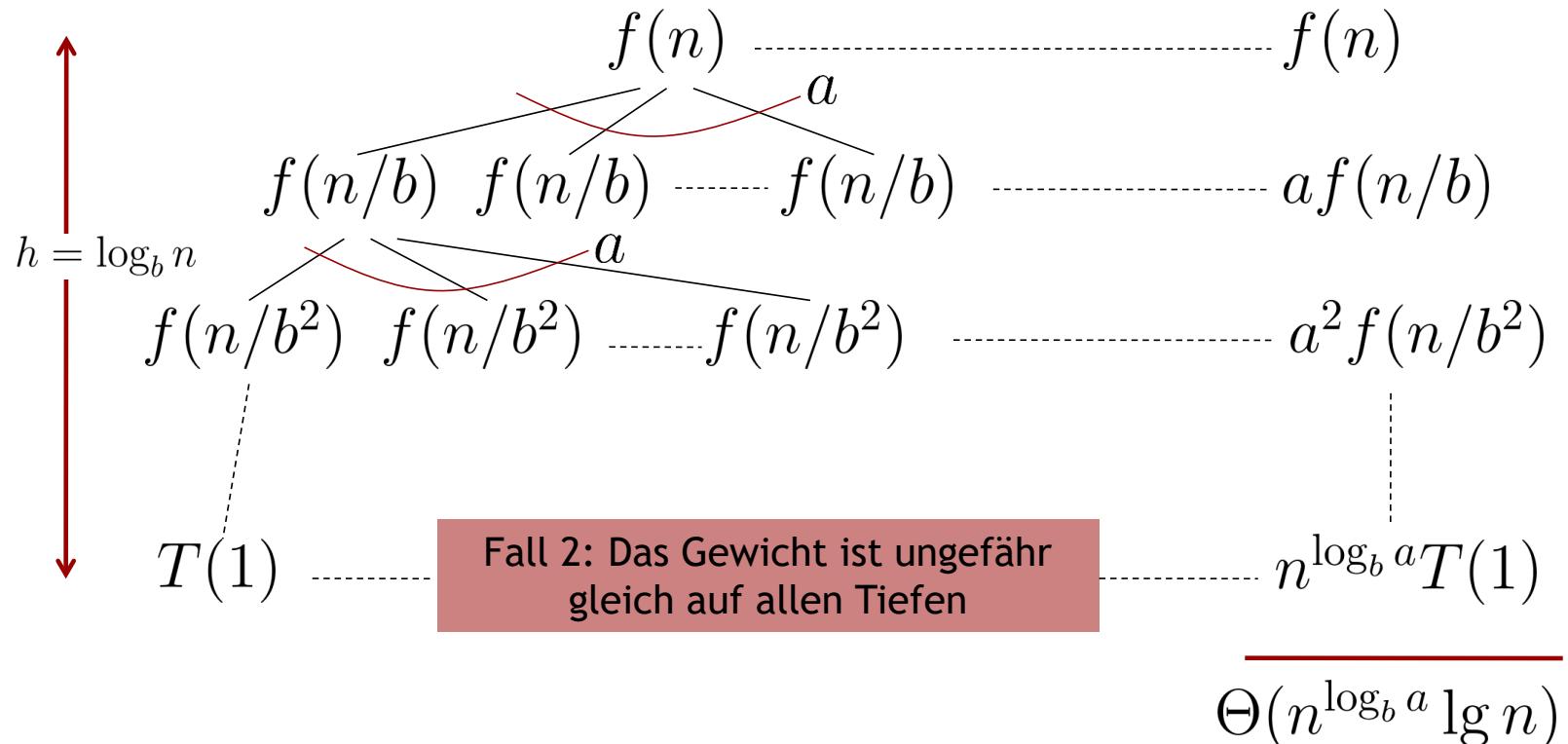
# Intuition für das Mastertheorem

- Rekursionsbaum



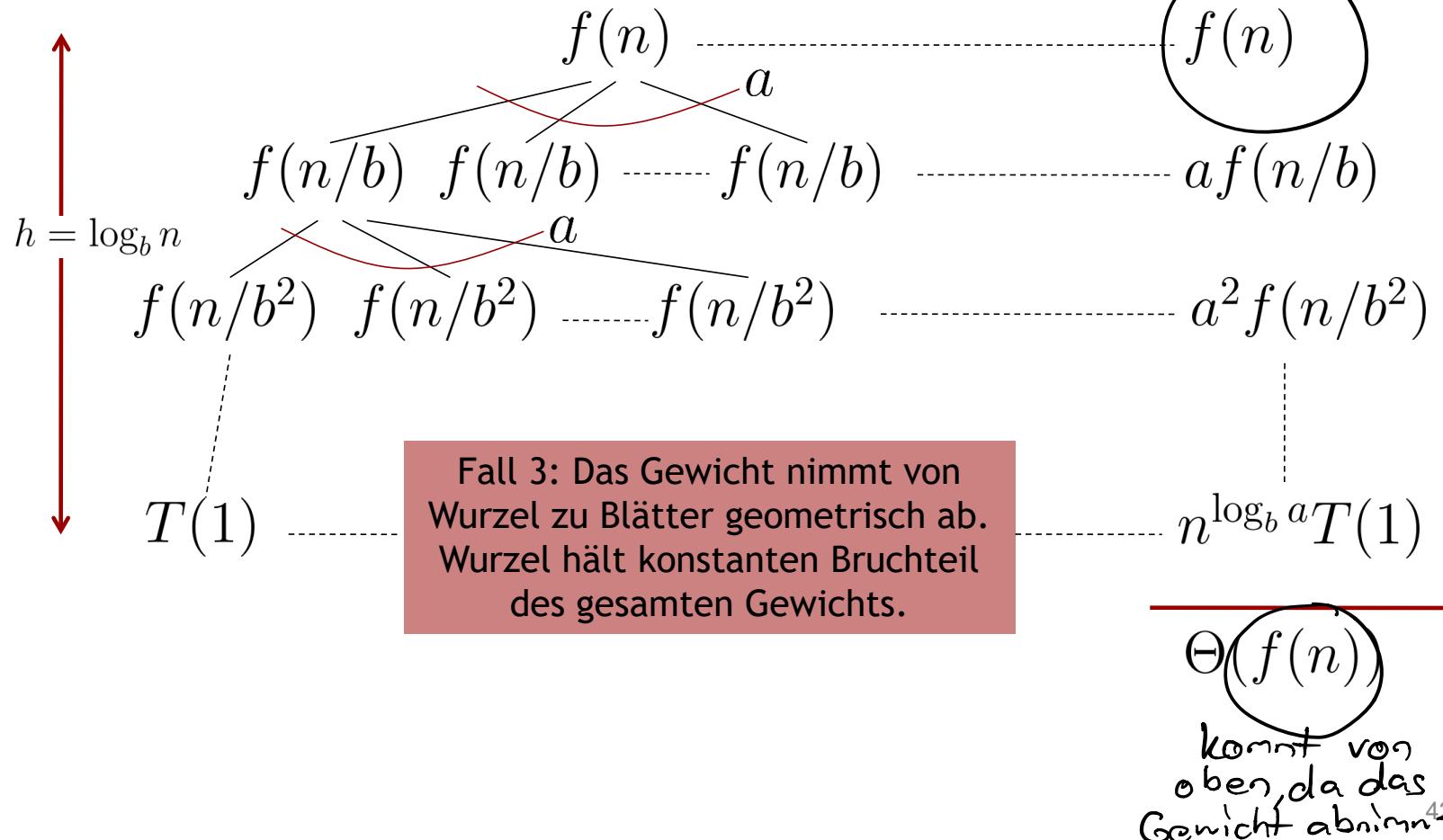
# Intuition für das Mastertheorem

- Rekursionsbaum



# Intuition für das Mastertheorem

- Rekursionsbaum



# Zusammenfassung

- Asymptotische Notationen  
 $O(g(n)), \Omega(g(n)), \Theta(g(n)), o(g(n)), \omega(g(n))$
- Komplexitätsklassen  
 $\Theta(1), \Theta(\lg n), \Theta(n), \Theta(n^m), \Theta(2^n), \Theta(3^n), \dots$
- Gleichungen mit Landau Notation
- Rekursionsgleichungen
  - Substitutionsmethode
  - Rekursionsbaum-Methode
  - Mastertheorem

# Nächste Vorlesung

- Sortieralgorithmen
- Kapitel 6 und 7