Grundbegriffe der Informatik

Einheit 18: Quantitative Aspekte von Algorithmen

Mattias Ulbrich
(basierend auf Folien von Thomas Worsch)

KIT · Institut für Theoretische Informatik

Wintersemester 2023/2024

Überblick

Karlsruher Institut für Technologie

Ressourcenverbrauch bei Berechnungen

Groß-O-Notation

- Ignorieren konstanter Faktoren
- Notation für obere und untere Schranken des Wachstums
- Eine grauenhafte Schreibweise
- Rechnen im O-Kalkül

Matrizenmultiplikation

- Rückblick auf die Schulmethode
- Algorithmus von Strassen

Asymptotisches Verhalten induktiv definierter Funktionen

- Laufzeit von Teile-und-Herrsche-Algorithmen

Wo sind wir?

Karlsruher Institut für Technologie

Ressourcenverbrauch bei Berechnungen

Groß-O-Notation

Matrizenmultiplikation

Asymptotisches Verhalten induktiv definierter Funktionen

Ressourcenverbrauch — wie detailliert?

- für jede Probleminstanz einzeln
  präzise aber oft unpraktikabel
- gröber: nur in Abhängigkeit von der «Problemgröße»
- wenn Ressourcenverbrauch
  für verschiedene Instanzen gleicher Größe
  unterschiedlich ist:
  - bester Fall? (best case)
  - Durchschnitt? (average case)
  - schlechtester Fall? (worst case)

Ressourcenverbrauch — wie detailliert?

- für jede Probleminstanz einzeln
  präzise aber oft unpraktikabel
- gröber: nur in Abhängigkeit von der «Problemgröße»
- wenn Ressourcenverbrauch
  für verschiedene Instanzen gleicher Größe
  unterschiedlich ist:
  - bester Fall? (best case)
    - oft uninteressant
  - Durchschnitt? (average case)
  - schlechtester Fall? (worst case)

Ressourcenverbrauch — wie detailliert?

- für jede Probleminstanz einzeln
  präzise aber oft unpraktikabel
- gröber: nur in Abhängigkeit von der «Problemgröße»
- wenn Ressourcenverbrauch
  für verschiedene Instanzen gleicher Größe
  unterschiedlich ist:
  - bester Fall? (best case)
    - oft uninteressant
  - Durchschnitt? (average case)
    - oft schwer zu bestimmen
  - schlechtester Fall? (worst case)

Ressourcenverbrauch — wie detailliert?

- für jede Probleminstanz einzeln
  präzise aber oft unpraktikabel
- gröber: nur in Abhängigkeit von der «Problemgröße»
- wenn Ressourcenverbrauch
  für verschiedene Instanzen gleicher Größe
  unterschiedlich ist:
  - bester Fall? (best case)
    - oft uninteressant
  - Durchschnitt? (average case)
    - oft schwer zu bestimmen
  - schlechtester Fall? (worst case)
    - oft angegeben
    - mit dem Hinweis, dass es auch besser sein kann …

Was ist wichtig

- Das sollten Sie mitnehmen:
  - Bedarf an
    - Rechenzeit und
    - Speicherplatzbedarf

    wichtige Komplexitätsmaße
  - oft Quantifizierung in Abhängigkeit von Problemgröße
    - oft schlimmster Fall (worst case)
    - gelegentlich ein «mittlerer» Fall (average case)
- Das sollten Sie üben:
  - Abschätzen/ausrechnen wie oft ein Programmstück, z. B. ein Schleifenrumpf, durchlaufen wird.

Wo sind wir?

Ressourcenverbrauch bei Berechnungen

Groß-O-Notation

Matrizenmultiplikation

Asymptotisches Verhalten induktiv definierter Funktionen

Warum keine exakten Angaben?

- **man will nicht**
  - Faulheit
  - Vergänglichkeit der genauen Werte
    - bald neuer Prozessor
  - mangelndes Interesse an genauen Werten
- **man kann nicht**
  - Unfähigkeit
  - Unkenntnis von Randbedingungen
  - Ungenauigkeiten im «Algorithmus» (??)
- **man «soll» nicht**
  - Unabhängigkeit von konkreten Probleminstanzen
  - Unabhängigkeit von Programmiersprache
  - Unabhängigkeit von Prozessor

Wie ungenau wollen wir über Funktionen reden?

- **Ignorieren konstanter Faktoren**
  - Motivation: Geschwindigkeitssteigerungen bei Prozessoren irrelevant
- **nur obere (bzw. untere) Schranken**
  - Motivation: können nur schlechtesten Fall analysieren

O, Θ, Ω — zur Notation asymptotischen Wachstums

- wichtiges Handwerkszeug zum
  - Reden über und
  - Ausrechnen von

  z. B. Laufzeiten
- insbesondere «kontrollierte Ungenauigkeiten»
  - O: «Groß-O»
    - eingeführt von Bachmann (oder früher)
    - von Landau bekannt gemacht
  - Ω, Θ
    - von Knuth zumindest verbreitet

Wo sind wir?

Karlsruher Institut für Technologie

Ressourcenverbrauch bei Berechnungen

Groß-O-Notation

- Ignorieren konstanter Faktoren
- Notation für obere und untere Schranken des Wachstums
- Eine grauenhafte Schreibweise
- Rechnen im O-Kalkül

Matrizenmultiplikation

Asymptotisches Verhalten induktiv definierter Funktionen

Zu Notation und Redeweise

- Notation:
  - $\mathbb{R}_+$: Menge der positiven reellen Zahlen (*ohne* 0)
  - $\mathbb{R}_0^+$: Menge der nichtnegativen rellen Zahlen, $\mathbb{R}_0^+ = \mathbb{R}_+ \cup \{0\}$.
  - betrachten Funktionen $f : \mathbb{N}_0 \to \mathbb{R}_0^+$.

Zu Notation und Redeweise

- Notation:
  - $\mathbb{R}_+$: Menge der positiven reellen Zahlen (*ohne* 0)
  - $\mathbb{R}_0^+$: Menge der nichtnegativen rellen Zahlen, $\mathbb{R}_0^+ = \mathbb{R}_+ \cup \{0\}$.
  - betrachten Funktionen $f : \mathbb{N}_0 \to \mathbb{R}_0^+$.
- Redeweisen:
  - *asymptotisches Wachstum* oder
  - *größenordnungsmäßiges Wachstum* von Funktionen

Zu Notation und Redeweise

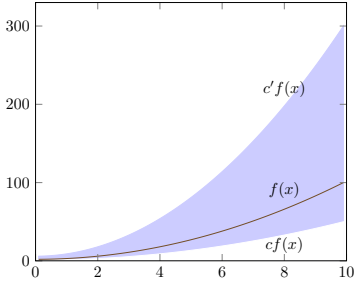
- Notation:
  - $\mathbb{R}_+$: Menge der positiven reellen Zahlen (*ohne* 0)
  - $\mathbb{R}_0^+$: Menge der nichtnegativen rellen Zahlen, $\mathbb{R}_0^+ = \mathbb{R}_+ \cup \{0\}$.
  - betrachten Funktionen $f : \mathbb{N}_0 \to \mathbb{R}_0^+$.
- Redeweisen:
  - *asymptotisches Wachstum* oder
  - *größenordnungsmäßiges Wachstum* von Funktionen
- Definition:
  - Funktion $g : \mathbb{N}_0 \to \mathbb{R}_0^+$ wächst asymptotisch genauso schnell wie Funktion $f : \mathbb{N}_0 \to \mathbb{R}_0^+$, wenn gilt:

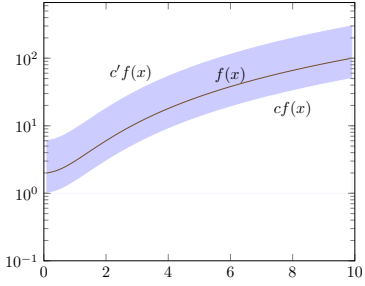
$$\exists c, c' \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : cf(n) \leq g(n) \leq c'f(n)$$

  - schreibe $f \asymp g$ oder $f(n) \asymp g(n)$

Erläuterungen zur Definition von $f \asymp g$ (1)

$$\exists c, c' \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : cf(n) \leq g(n) \leq c'f(n)$$



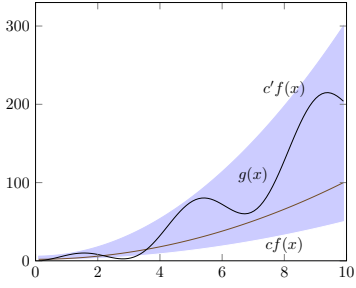


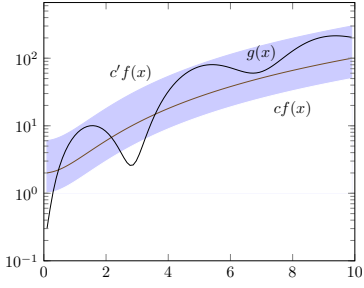
Achtung: links lineare y-Skala, rechts logarithmische y-Skala

Achtung: kontinuierliche Funktionen gezeichnet,
aber Definition für diskrete Funktionen

Erläuterungen zur Definition von $f \asymp g$ (2)

$$\exists c, c' \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : cf(n) \leq g(n) \leq c'f(n)$$





Achtung: links lineare y-Skala, rechts logarithmische y-Skala

Achtung: kontinuierliche Funktionen gezeichnet,
aber Definition für diskrete Funktionen

Beispiel

- $f\colon n \mapsto 3n^2$ und $g\colon n \mapsto 10^{-2}n^2$.
- Behauptung: $f \asymp g$

Beispiel

- $f\colon n \mapsto 3n^2$ und $g\colon n \mapsto 10^{-2}n^2$.
- Behauptung: $f \asymp g$
  - $cf(n) \leq g(n)$ gilt z. B. für $c = 10^{-3}$ und $n_0 = 0$:

$$\forall n \geq n_0 : cf(n) = 10^{-3} \cdot 3n^2 \leq 10^{-2}n^2 = g(n)$$

  - $g(n) \leq c'f(n)$ gilt z. B. für $c' = 1$ und $n_0 = 0$:

$$\forall n \geq n_0 : g(n) = 10^{-2}n^2 \leq 3n^2 = c'f(n)$$

Beispiel

- $f\colon n \mapsto 3n^2$ und $g\colon n \mapsto 10^{-2}n^2$.
- Behauptung: $f \asymp g$
  - $cf(n) \leq g(n)$ gilt z. B. für $c = 10^{-3}$ und $n_0 = 0$:

$$\forall n \geq n_0 : cf(n) = 10^{-3} \cdot 3n^2 \leq 10^{-2}n^2 = g(n)$$

  - $g(n) \leq c'f(n)$ gilt z. B. für $c' = 1$ und $n_0 = 0$:

$$\forall n \geq n_0 : g(n) = 10^{-2}n^2 \leq 3n^2 = c'f(n)$$

Rechenregel

- Für jedes $f : \mathbb{N}_0 \to \mathbb{R}_0^+$ gilt:

$$\forall a, b \in \mathbb{R}_+ : af \asymp bf$$

Nichtbeispiele für $\asymp$ (1)

- betrachte $f\colon n \mapsto n^2$ und $g\colon n \mapsto n^3$
- Behauptung: $f \not\asymp g$
- Begründung:
  - sonst insbesondere $g(n) \leq c' f(n)$ für alle n größer gleich einem n_0

Nichtbeispiele für $\asymp$ (1)

- betrachte $f\colon n \mapsto n^2$ und $g\colon n \mapsto n^3$
- Behauptung: $f \not\asymp g$
- Begründung:
  - sonst insbesondere $g(n) \leq c' f(n)$ für alle n größer gleich einem n_0
  - für $f(n) \neq 0$ äquivalent zu $g(n)/f(n) \leq c' \in \mathbb{R}_+$

Nichtbeispiele für $\asymp$ (1)

- betrachte $f\colon n \mapsto n^2$ und $g\colon n \mapsto n^3$
- Behauptung: $f \not\asymp g$
- Begründung:
  - sonst insbesondere $g(n) \leq c' f(n)$ für alle n größer gleich einem n_0
  - für $f(n) \neq 0$ äquivalent zu $g(n)/f(n) \leq c' \in \mathbb{R}_+$
  - ab einem n_0 für jedes n

Nichtbeispiele für $\asymp$ (1)

- betrachte $f\colon n \mapsto n^2$ und $g\colon n \mapsto n^3$
- Behauptung: $f \not\asymp g$
- Begründung:
  - sonst insbesondere $g(n) \leq c' f(n)$ für alle n größer gleich einem n_0
  - für $f(n) \neq 0$ äquivalent zu $g(n)/f(n) \leq c' \in \mathbb{R}_+$
  - ab einem n_0 für jedes n
  - aber $g(n)/f(n) = n$ nicht beschränkbar

Nichtbeispiele für $\asymp$ (2)

- betrachte $f\colon n \mapsto n^2$ und $g\colon n \mapsto 2^n$
- Behauptung: $f \not\asymp g$
- Begründung:
  - für $f \asymp g$ müsste insbesondere $g(n)/f(n) \leq c'$ gelten (für …)
  - Quotient $2^n/n^2$ kann durch keine Konstante beschränkt werden:
    - einfache Grenzwertbetrachtung, oder
    - betrachte die $n_i = 2^{i+2}$ und zeige durch Induktion:

$$\forall i \in \mathbb{N}_0 : 2^{n_i} \geq 4^i n_i^2$$

Äquivalenzrelation $\asymp$

Zeichen $\asymp$ erinnert an das Gleichheitszeichen.

Lemma. Die Relation $\asymp$ ist eine Äquivalenzrelation.

Erinnerung: Eine Äquivalenzrelation ist per definitionem

- reflexiv,
- symmetrisch,
- transitiv.

Relation $\asymp$ ist reflexiv und symmetrisch

- *Reflexitvität:* $f \asymp f$
  für $c = c' = 1$ und $n_0 = 0$ gilt:
  für jedes $n \geq n_0$ ist $cf(n) \leq f(n) \leq c'f(n)$

- *Symmetrie:* Wenn $f \asymp g$, dann auch $g \asymp f$
  Wenn für Konstanten $c, c' \in \mathbb{R}_+$, $n_0 \in \mathbb{N}_0$ und jedes $n \geq n_0$

  $$cf(n) \leq g(n) \leq c'f(n) \,,$$

  dann gilt für die gleichen $n \geq n_0$ und
  die Konstanten $d = 1/c$ und $d' = 1/c'$:

  $$d'g(n) \leq f(n) \leq dg(n) \,.$$

Relation $\asymp$ ist reflexiv und symmetrisch

- *Reflexitvität:* $f \asymp f$
  für $c = c' = 1$ und $n_0 = 0$ gilt:
  für jedes $n \geq n_0$ ist $cf(n) \leq f(n) \leq c'f(n)$

- *Symmetrie:* Wenn $f \asymp g$, dann auch $g \asymp f$
  Wenn für Konstanten $c, c' \in \mathbb{R}_+$, $n_0 \in \mathbb{N}_0$ und jedes $n \geq n_0$

  $$cf(n) \leq g(n) \leq c'f(n) \,,$$

  dann gilt für die gleichen $n \geq n_0$ und
  die Konstanten $d = 1/c$ und $d' = 1/c'$:

  $$d'g(n) \leq f(n) \leq dg(n) \,.$$

Relation $\asymp$ ist reflexiv und symmetrisch

- *Reflexitvität:* $f \asymp f$
  für $c = c' = 1$ und $n_0 = 0$ gilt:
  für jedes $n \geq n_0$ ist $cf(n) \leq f(n) \leq c'f(n)$

- *Symmetrie:* Wenn $f \asymp g$, dann auch $g \asymp f$
  Wenn für Konstanten $c, c' \in \mathbb{R}_+$, $n_0 \in \mathbb{N}_0$ und jedes $n \geq n_0$

  $$cf(n) \leq g(n) \leq c'f(n) \,,$$

  dann gilt für die gleichen $n \geq n_0$ und
  die Konstanten $d = 1/c$ und $d' = 1/c'$:

  $$d'g(n) \leq f(n) \leq dg(n) \,.$$

Relation $\asymp$ ist transitiv

- *Transitivität:* wenn $f \asymp g$ und $g \asymp h$, dann $f \asymp h$.
  gelte für Konstanten $c, c' \in \mathbb{R}_+$ und jedes $n \geq n_0$

$$cf(n) \leq g(n) \leq c'f(n)$$

  und für Konstanten $d, d' \in \mathbb{R}_+$ und jedes $n \geq n_1$

$$dg(n) \leq h(n) \leq d'g(n) \ .$$

  Dann gilt für jedes $n \geq \max(n_0, n_1)$

$$dcf(n) \leq dg(n) \leq h(n) \leq d'g(n) \leq d'c'f(n) \ ,$$

  wobei auch die Konstanten dc und $d'c'$ wieder positiv sind.

- $\Theta(f)$: Menge aller Funktionen, die zu f im Sinne von $\asymp$ äquivalent sind
- Also:

$$\Theta(f) = \{\, g \mid f \asymp g \,\}$$
$$= \{\, g \mid \exists c, c' \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : cf(n) \leq g(n) \leq c'f(n) \,\}$$

einfache Rechenregel für Θ

Rechenregel Für alle $f : \mathbb{N}_0 \to \mathbb{R}_0^+$ und alle Konstanten $a, b \in \mathbb{R}_+$ gilt:

$$\Theta(af) = \Theta(bf)$$

Wo sind wir?

Karlsruher Institut für Technologie

Ressourcenverbrauch bei Berechnungen

Groß-O-Notation

- Ignorieren konstanter Faktoren
- Notation für obere und untere Schranken des Wachstums
- Eine grauenhafte Schreibweise
- Rechnen im O-Kalkül

Matrizenmultiplikation

Asymptotisches Verhalten induktiv definierter Funktionen

Obere und untere Schranken

- manchmal für Funktion nur obere (bzw. untere) Schranke bekannt
- Definition:
  - $O(f) = \{ g \mid \exists c \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : g(n) \leq cf(n) \}$
  - $\Omega(f) = \{ g \mid \exists c \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : g(n) \geq cf(n) \}$
- gelegentlich auch
  - $g \preceq f$ falls $g \in O(f)$
  - $g \succeq f$ falls $g \in \Omega(f)$
- Redeweise
  - g wächst asymptotisch höchstens so schnell wie f (für $g \preceq f$)
  - g wächst asymptotisch mindestens so schnell wie f (für $g \succeq f$)

Beobachtungen

- Definition:
  - $O(f) = \{\, g \mid \exists c \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : g(n) \leq cf(n) \,\}$
- häufig hilfreich zum Nachprüfen:
  $O(f) = \{\, g \mid \exists c \in \mathbb{R}_+ : \exists n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : \frac{g(n)}{f(n)} \leq c \,\}$
- Alternative äquivalente Definition:
  $O(f) = \{\, g \mid \limsup_n \frac{g(n)}{f(n)} < \infty \,\}$ (der größte Häufungspunkt existiert)

Beispiel (1)

- Es sei $g\colon n \mapsto 10^{90}n^7$ und $f\colon n \mapsto 10^{-90}n^8$
- Behauptung: $g \in O(f)$
- Begründung:
  - wähle $c = 10^{180}$ und $n_0 = 0$
  - dann für jedes $n \geq 0$: $10^{90}n^7 \leq c \cdot 10^{-90}n^8$.
- in $O(\cdot)$ usw. können *große* Konstanten stecken

Beispiel (2)

- Was ist $O(1)$?

Beispiel (2)

- Was ist $O(1)$?
- 1 steht für die konstante Funktion $\mathbb{1}\colon n \mapsto 1$
- Definition: alle Funktionen g, für die es $c \in \mathbb{R}_+$ und $n_0 \in \mathbb{N}_0$ gibt, so dass für jedes $n \geq n_0$ gilt:

$$g(n) \leq c \cdot 1 = c$$

- alle Funktionen, die durch Konstante beschränkbar sind
  - alle konstanten Funktionen
  - aber auch Funktionen wie $3 + \sin(n)$

Beispiel (3)

- Quotient n^2/n nicht für große n durch Konstante beschränkbar
- also gilt *nicht* $n^2 \in O(n)$
- aber $n \in O\left(n^2\right)$
- Relation $\preceq$ ist also *nicht* symmetrisch
- für alle positiven reellen Konstanten $0 < a < b$ gilt

$a \neq b$!

$$n^a \preceq n^b \quad \text{aber} \quad n^b \not\preceq n^a$$

$$\text{also} \quad n^a \in O\left(n^b\right) \quad \text{aber} \quad n^b \notin O\left(n^a\right)$$

$$\text{also} \quad n^b \in \Omega\left(n^a\right) \quad \text{aber} \quad n^a \notin \Omega\left(n^b\right)$$

Beispiel (4)

- Quotient $2^n/n^2$ nicht für große n durch Konstante beschränkbar
- also gilt *nicht* $2^n \in O\left(n^2\right)$.
- aber $n^2 \in O\left(2^n\right)$.
- für alle reellen Konstanten $a > 1$ und $b > 1$, gilt

$$
\begin{array}{lrll}
 & n^a \preceq b^n & \text{aber} & b^n \not\preceq n^a \\
\text{also} & n^a \in O\left(b^n\right) & \text{aber} & b^n \notin O\left(n^a\right) \\
\text{also} & b^n \in \Omega\left(n^a\right) & \text{aber} & n^a \notin \Omega\left(b^n\right)
\end{array}
$$

Pingo-Quiz!

s.kit.edu/
gbi-pingo

Welche Aussagen über die Zahl $Z(n)$ der Aufrufe op im folgenden Programm, wenn n ein Parameter ist, stimmen?

```
for(int i = 0; i < n; i++)
  for(int j = i+1; j < 2*n-i; j++)
    op(i,j);
```

Option A $Z(n) \in O(n)$

Option B $Z(n) \in O(n^2)$

Option C $Z(n) \in O(n^3)$

Einfache Beobachtungen

- in Ungleichung $g(n) \leq cf(n)$ Konstante auf die andere Seite bringen liefert

Rechenregel Für alle Funktionen $f : \mathbb{N}_0 \to \mathbb{R}_0^+$ und $g : \mathbb{N}_0 \to \mathbb{R}_0^+$ gilt:
$g \in O(f) \longleftrightarrow f \in \Omega(g)$ also $g \preceq f \longleftrightarrow f \succeq g$

- Man kann auch zeigen:

$$\Theta(f) = O(f) \cap \Omega(f)$$

also $$g \asymp f \longleftrightarrow g \preceq f \wedge g \succeq f$$

Vorsicht!
- es gibt f und g so, dass *weder* $g \preceq f$ *noch* $f \preceq g$

Wo sind wir?

Karlsruher Institut für Technologie

Ressourcenverbrauch bei Berechnungen

Groß-O-Notation

- Ignorieren konstanter Faktoren
- Notation für obere und untere Schranken des Wachstums
- Eine grauenhafte Schreibweise
- Rechnen im O-Kalkül

Matrizenmultiplikation

Asymptotisches Verhalten induktiv definierter Funktionen

Für die Lektüre leider unverzichtbar

- sehr unschöne Variante der O-Notation, aber *sehr weit* verbreitet
- Man schreibt

$$
\begin{aligned}
g(n) &= O\left(f(n)\right) \quad \text{statt} \quad g \in O\left(f\right) \ , \\
g(n) &= \Theta\left(f(n)\right) \quad \text{statt} \quad g \in \Theta\left(f\right) \ , \\
g(n) &= \Omega\left(f(n)\right) \quad \text{statt} \quad g \in \Omega\left(f\right) \ .
\end{aligned}
$$

- Ausdrücke auf der linken Seite sind keine Gleichungen!
- Lassen Sie daher bitte immer große Vorsicht walten.

Wo sind wir?

Karlsruher Institut für Technologie

Ressourcenverbrauch bei Berechnungen

Groß-O-Notation

- Ignorieren konstanter Faktoren
- Notation für obere und untere Schranken des Wachstums
- Eine grauenhafte Schreibweise
- Rechnen im O-Kalkül

Matrizenmultiplikation

Asymptotisches Verhalten induktiv definierter Funktionen

Eine nützliche Rechenregel

- Ist $g_1 \preceq f_1$ und $g_2 \preceq f_2$, dann ist auch $g_1 + g_2 \preceq f_1 + f_2$.
- Ist umgekehrt $g \preceq f_1 + f_2$, dann kann man g in der Form $g = g_1 + g_2$ schreiben mit $g_1 \preceq f_1$ und $g_2 \preceq f_2$.
- äquivalente Formulierung:

Lemma Für alle Funktionen $f_1, f_2 : \mathbb{N}_0 \to \mathbb{R}_0^+$ gilt:

$$O(f_1) + O(f_2) = O(f_1 + f_2)$$

- Pluszeichen auf der linken Seite bedarf der Erläuterung …

Komplexoperationen

- Sind M_1 und M_2 Mengen, deren Elemente man addieren bzw. multiplizieren kann, dann sei

$$M_1 + M_2 = \{g_1 + g_2 \mid g_1 \in M_1 \wedge g_2 \in M_2\}$$
$$M_1 \cdot M_2 = \{g_1 \cdot g_2 \mid g_1 \in M_1 \wedge g_2 \in M_2\}$$

- nichts Neues: analoge Definition des Produkts formaler Sprachen

Komplexoperationen (2)

- Wenn eine der Mengen M_i einelementig ist, lässt man manchmal die Mengenklammern darum weg.
- Beispiele
  - mit Zahlenmengen

    statt $\{3\} \cdot \mathbb{N}_0 + \{1\}$ kürzer $3\mathbb{N}_0 + 1$

  - mit Funktionenmengen

    statt $\{n^3\} + O(n^2)$ kürzer $n^3 + O(n^2)$

Beweis von $O(f_1) + O(f_2) = O(f_1 + f_2)$

„$\subseteq$“: wenn für $n \geq n_1$ gilt: $g_1(n) \leq c_1 f_1(n)$ und
wenn für $n \geq n_2$ gilt: $g_2(n) \leq c_2 f_2(n)$, dann
für $n \geq n_0 = \max(n_1, n_2)$ und $c = \max(c_1, c_2)$:

$$\begin{aligned} g_1(n) + g_2(n) &\leq c_1 f_1(n) + c_2 f_2(n) \\ &\leq c f_1(n) + c f_2(n) \\ &= c(f_1(n) + f_2(n)) \end{aligned}$$

„$\supseteq$“: zu $g \in O(f_1 + f_2)$ finde g_1 und g_2 …:

Beweis von $O(f_1) + O(f_2) = O(f_1 + f_2)$

„$\subseteq$“: wenn für $n \geq n_1$ gilt: $g_1(n) \leq c_1 f_1(n)$ und
wenn für $n \geq n_2$ gilt: $g_2(n) \leq c_2 f_2(n)$, dann
für $n \geq n_0 = \max(n_1, n_2)$ und $c = \max(c_1, c_2)$:

$$\begin{aligned} g_1(n) + g_2(n) &\leq c_1 f_1(n) + c_2 f_2(n) \\ &\leq c f_1(n) + c f_2(n) \\ &= c(f_1(n) + f_2(n)) \end{aligned}$$

„$\supseteq$“: zu $g \in O(f_1 + f_2)$ finde g_1 und g_2 …:

$$\text{definiere} \qquad g_1(n) = \begin{cases} g(n) & \text{falls } g(n) \leq c f_1(n) \\ c f_1(n) & \text{falls } g(n) > c f_1(n) \end{cases}$$

$$\text{und} \qquad g_2(n) = g(n) - g_1(n)$$

Der Rest ist einfache Rechnung.

Weitere Regeln

Rechenregel Wenn $g_1 \preceq f_1$ ist, und wenn $g_1 \prec g_2$ und $f_1 \prec f_2$, dann gilt auch $g_2 \preceq f_2$.

Rechenregel Wenn $g \preceq f$ ist, also $g \in O(f)$, dann ist auch $O(g) \subseteq O(f)$ und $O(g+f) = O(f)$.

Beispiel: $O\left(n^3 + n^2\right) = O\left(n^3\right)$

Insbesondere Für Polynome gilt (mit $a_k \geq 0$):
$a_k \cdot n^k + a_{k-1} \cdot n^{k-1} + \ldots + a_1 \cdot n + a_0 \in O\left(n^k\right)$

sogar:
$a_k \cdot n^k + a_{k-1} \cdot n^{k-1} + \ldots + a_1 \cdot n + a_0 \in \Theta\left(n^k\right)$

Rechenzeiten

reale Rechenzeiten für $T(n)$ Schritte, wenn jeder Schritt 1 ns benötigt

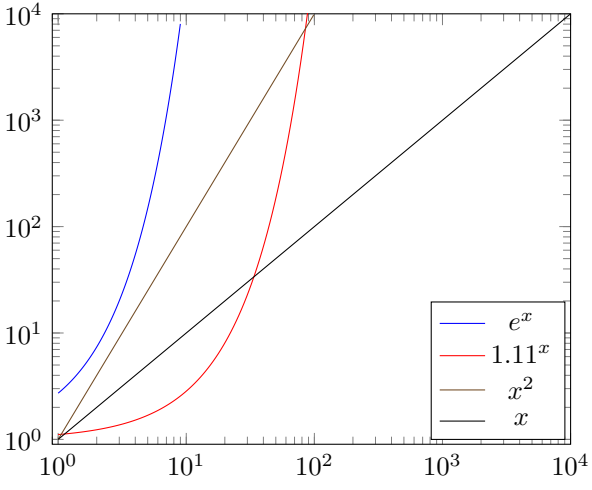
$T(n)$	n					
	10	10^2	10^3	10^4	10^5	10^6
$\log_2(n)$	3.32 ns	6.64 ns	9.97 ns	13.29 ns	16.61 ns	19.93 ns
$\sqrt{n}$	3.16 ns	10.00 ns	31.62 ns	100.00 ns	316.23 ns	1.00 µs
n	10.00 ns	100.00 ns	1.00 µs	10.00 µs	100.00 µs	1.00 ms
$n \cdot \log_2(n)$	33.22 ns	664.39 ns	9.97 µs	132.88 µs	1.66 ms	19.93 ms
n^2	100.00 ns	10.00 µs	1.00 ms	100.00 ms	10.00 s	0.27 h
n^3	1.00 µs	1.00 ms	1.00 s	0.27 h	11.57 d	31.71 a
1.01^n	1.10 ns	2.70 ns	20.96 µs			
1.1^n	2.59 ns	13.78 µs				
2^n	1.02 µs					
n^n	10 s					

Rechenzeiten

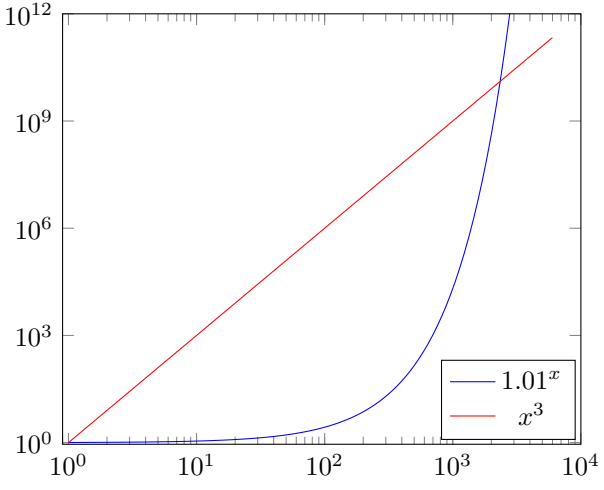
reale Rechenzeiten für $T(n)$ Schritte, wenn jeder Schritt 1 ns benötigt

$T(n)$	n					
	10	10^2	10^3	10^4	10^5	10^6
$\log_2(n)$	3.32 ns	6.64 ns	9.97 ns	13.29 ns	16.61 ns	19.93 ns
$\sqrt{n}$	3.16 ns	10.00 ns	31.62 ns	100.00 ns	316.23 ns	1.00 µs
n	10.00 ns	100.00 ns	1.00 µs	10.00 µs	100.00 µs	1.00 ms
$n \cdot \log_2(n)$	33.22 ns	664.39 ns	9.97 µs	132.88 µs	1.66 ms	19.93 ms
n^2	100.00 ns	10.00 µs	1.00 ms	100.00 ms	10.00 s	0.27 h
n^3	1.00 µs	1.00 ms	1.00 s	0.27 h	11.57 d	31.71 a
1.01^n	1.10 ns	2.70 ns	20.96 µs	$5.2 \cdot 10^{29}$ a		
1.1^n	2.59 ns	13.78 µs				
2^n	1.02 µs					
n^n	10 s					

Rechenzeiten



Rechenzeiten (2)



Was ist wichtig

- Das sollten Sie mitnehmen:
  - Definitionen von $O(f)$, $\Theta(f)$, $\Omega(f)$
  - Definitionen von $\preceq$, $\asymp$, $\succeq$
- Das sollten Sie üben:
  - Anschauung für $O(f)$, $\Theta(f)$, $\Omega(f)$
  - rechnen mit $O(f)$, $\Theta(f)$, $\Omega(f)$

Wo sind wir?

Ressourcenverbrauch bei Berechnungen

Groß-O-Notation

Matrizenmultiplikation

Asymptotisches Verhalten induktiv definierter Funktionen

Wo sind wir?

Karlsruher Institut für Technologie

Ressourcenverbrauch bei Berechnungen

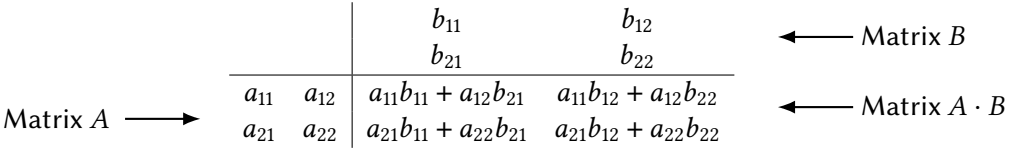
Groß-O-Notation

Matrizenmultiplikation

- Rückblick auf die Schulmethode
- Algorithmus von Strassen

Asymptotisches Verhalten induktiv definierter Funktionen

Multiplikation von 2×2-Matrizen



- Anzahl Multiplikationen: $N_{\text{mult}}(2) = 2^2 \cdot 2 = 8$
- Anzahl Additionen: $N_{\text{add}}(2) = 2^2 \cdot (2-1) = 4$.

Multiplikation von $n \times n$-Matrizen mit Blockaufteilung

- n sei gerade
- Verwendung von Blockmatrizen der Größe $n/2 \times n/2$

		B_{11}	B_{12}
		B_{21}	B_{22}
A_{11}	A_{12}	$A_{11}B_{11} + A_{12}B_{21}$	$A_{11}B_{12} + A_{12}B_{22}$
A_{21}	A_{22}	$A_{21}B_{11} + A_{22}B_{21}$	$A_{21}B_{12} + A_{22}B_{22}$

Mitteilung: Das ist auch die Produktmatrix!

- 8 Multiplikationen von Blockmatrizen und
  4 Additionen von Blockmatrizen.
- Anzahl elementarer Operationen
  - $N_{\mathsf{mult}}(n) = 8 \cdot N_{\mathsf{mult}}(n/2)$, $N_{\mathsf{mult}}(1) = 1$ und
  - $N_{\mathsf{add}}(n) = 8 \cdot N_{\mathsf{add}}(n/2) + 4 \cdot (n/2)^2 = 8 \cdot N_{\mathsf{add}}(n/2) + n^2$, $N_{\mathsf{add}}(1) = 0$.

- aus $N_{\text{mult}}(n) = 8 \cdot N_{\text{mult}}(n/2)$ und $N_{\text{mult}}(1) = 1$ folgt (induktiv)

$$\begin{aligned} N_{\text{mult}}(2^k) &= 8 \cdot N_{\text{mult}}(2^{k-1}) = 8 \cdot 8 \cdot N_{\text{mult}}(2^{k-2}) = \cdots \\ &= 8^k \cdot N_{\text{mult}}(2^0) \\ &= 8^k = (2^3)^k = (2^k)^3 = n^3 \end{aligned}$$

- Aus $N_{\text{add}}(n) = 8 \cdot N_{\text{add}}(n/2) + n^2$ und $N_{\text{add}}(1) = 0$ folgt

$$\begin{aligned} N_{\text{add}}(2^k) &= 8 \cdot N_{\text{add}}(2^{k-1}) + 4^k \\ &= 8 \cdot 8 \cdot N_{\text{add}}(2^{k-2}) + 8 \cdot 4^{k-1} + 4^k = \cdots \\ &= 8 \cdot 8 \cdot N_{\text{add}}(2^{k-2}) + 2 \cdot 4^k + 4^k = \cdots \\ &= 8^k N_{\text{add}}(2^0) + (2^{k-1} + \cdots + 1) \cdot 4^k = \\ &= 0 + 2^k \cdot 4^k - 4^k = n^3 - n^2 \end{aligned}$$

Multiplikation von $2^k \times 2^k$-Matrizen, also $n = 2^k$ ($n \neq 2^k$ «ähnlich»)

- aus $N_{\text{mult}}(n) = 8 \cdot N_{\text{mult}}(n/2)$ und $N_{\text{mult}}(1) = 1$ folgt (induktiv)

$$\begin{aligned} N_{\text{mult}}(2^k) &= 8 \cdot N_{\text{mult}}(2^{k-1}) = 8 \cdot 8 \cdot N_{\text{mult}}(2^{k-2}) = \cdots \\ &= 8^k \cdot N_{\text{mult}}(2^0) \\ &= 8^k = (2^3)^k = (2^k)^3 = n^3 \end{aligned}$$

- Aus $N_{\text{add}}(n) = 8 \cdot N_{\text{add}}(n/2) + n^2$ und $N_{\text{add}}(1) = 0$ folgt

$$\begin{aligned} N_{\text{add}}(2^k) &= 8 \cdot N_{\text{add}}(2^{k-1}) + 4^k \\ &= 8 \cdot 8 \cdot N_{\text{add}}(2^{k-2}) + 8 \cdot 4^{k-1} + 4^k = \cdots \\ &= 8 \cdot 8 \cdot N_{\text{add}}(2^{k-2}) + 2 \cdot 4^k + 4^k = \cdots \\ &= 8^k N_{\text{add}}(2^0) + (2^{k-1} + \cdots + 1) \cdot 4^k = \\ &= 0 + 2^k \cdot 4^k - 4^k = n^3 - n^2 \end{aligned}$$

$\sum_{i=0}^{k-1} 2^i = 2^k - 1$

Wo sind wir?

Die Idee von Volker Strassen (1969)

Die Idee von Volker Strassen (1969)

- $$\begin{array}{cc|cc} & & B_{11} & B_{12} \\ & & B_{21} & B_{22} \\ \hline A_{11} & A_{12} & C_{11} & C_{12} \\ A_{21} & A_{22} & C_{21} & C_{22} \end{array}$$

- Einträge C_{ij} des Produktes ergeben sich auch so:

$$C_{11} = M_1 + M_4 - M_5 + M_7 \qquad C_{12} = M_3 + M_5$$
$$C_{21} = M_2 + M_4 \qquad C_{22} = M_1 - M_2 + M_3 + M_6$$

wobei
$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22}) \qquad M_2 = (A_{21} + A_{22})B_{11}$$
$$M_3 = A_{11}(B_{12} - B_{22}) \qquad M_4 = A_{22}(B_{21} - B_{11})$$
$$M_5 = (A_{11} + A_{12})B_{22} \qquad M_6 = (A_{21} - A_{11})(B_{11} + B_{12})$$
$$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$$

Die Idee von Strassen (2)

- Blockmatrizen

$$C_{11} = M_1 + M_4 - M_5 + M_7 \qquad C_{12} = M_3 + M_5$$
$$C_{21} = M_2 + M_4 \qquad C_{22} = M_1 - M_2 + M_3 + M_6$$

wobei
$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22}) \qquad M_2 = (A_{21} + A_{22})B_{11}$$
$$M_3 = A_{11}(B_{12} - B_{22}) \qquad M_4 = A_{22}(B_{21} - B_{11})$$
$$M_5 = (A_{11} + A_{12})B_{22} \qquad M_6 = (A_{21} - A_{11})(B_{11} + B_{12})$$
$$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$$

- 18 Additionen statt 4
- 7 Multiplikationen statt 8

Die Idee von Strassen (3)

- 18 Additionen statt 4
- 7 Multiplikationen statt 8
- ja und?

Die Idee von Strassen (3)

- 18 Additionen statt 4
- 7 Multiplikationen statt 8
- ja und?
- ganze Block*matrizen*:
  Additionen sind *viel* «billiger» als Multiplikationen

Aufwandsabschätzung für den Algorithmus von Strassen

- Anzahl elementarer Operationen:
  - $N_{\text{mult}}(n) = 7 \cdot N_{\text{mult}}(n/2)$
  - $N_{\text{add}}(n) = 7 \cdot N_{\text{add}}(n/2) + 18 \cdot (n/2)^2 = 7 \cdot N_{\text{add}}(n/2) + 4.5 \cdot n^2$
- Für den Fall $n = 2^k$ ergibt sich:

$$\begin{aligned} N_{\text{mult}}(2^k) &= 7 \cdot N_{\text{mult}}(2^{k-1}) = 7 \cdot 7 \cdot N_{\text{mult}}(2^{k-2}) = \cdots \\ &= 7^k = (2^{\log_2 7})^k = n^{\log_2 7} \approx n^{2.807\ldots} \end{aligned}$$

- analog $N_{\text{add}}(n) \in \Theta(n^{\log_2 7})$

Aufwandsabschätzung für den Algorithmus von Strassen

- Anzahl elementarer Operationen:
  - $N_{\text{mult}}(n) = 7 \cdot N_{\text{mult}}(n/2)$
  - $N_{\text{add}}(n) = 7 \cdot N_{\text{add}}(n/2) + 18 \cdot (n/2)^2 = 7 \cdot N_{\text{add}}(n/2) + 4.5 \cdot n^2$
- Für den Fall $n = 2^k$ ergibt sich:

$$\begin{aligned} N_{\text{mult}}(2^k) &= 7 \cdot N_{\text{mult}}(2^{k-1}) = 7 \cdot 7 \cdot N_{\text{mult}}(2^{k-2}) = \cdots \\ &= 7^k = (2^{\log_2 7})^k = n^{\log_2 7} \approx n^{2.807\ldots} \end{aligned}$$

- analog $N_{\text{add}}(n) \in \Theta(n^{\log_2 7})$

Aufwandsabschätzung für den Algorithmus von Strassen

- Anzahl elementarer Operationen:
  - $N_{\text{mult}}(n) = 7 \cdot N_{\text{mult}}(n/2)$
  - $N_{\text{add}}(n) = 7 \cdot N_{\text{add}}(n/2) + 18 \cdot (n/2)^2 = 7 \cdot N_{\text{add}}(n/2) + 4.5 \cdot n^2$
- Für den Fall $n = 2^k$ ergibt sich:

$$\begin{aligned} N_{\text{mult}}(2^k) &= 7 \cdot N_{\text{mult}}(2^{k-1}) = 7 \cdot 7 \cdot N_{\text{mult}}(2^{k-2}) = \cdots \\ &= 7^k = (2^{\log_2 7})^k = n^{\log_2 7} \approx n^{2.807\ldots} \end{aligned}$$

- analog $N_{\text{add}}(n) \in \Theta(n^{\log_2 7})$
- Gesamtzahl elementarer Operationen ist also in

$$\Theta(n^{\log_2 7}) + \Theta(n^{\log_2 7}) = \Theta(n^{\log_2 7}) \approx \Theta(n^{2.807\ldots})$$

besser als n^3 für große n !

Matrizenmultiplikation — geht es noch schneller?

- Coppersmith und Winograd (1990): $O\left(n^{2.376\ldots}\right)$
  - der konstante Faktor ist *groß*
- Virginia Vassilevska Williams (2012, 2020): noch ein bisschen schneller

Matrizenmultiplikation — geht es noch schneller?

- Coppersmith und Winograd (1990): $O\left(n^{2.376\ldots}\right)$
  - der konstante Faktor ist *groß*
- Virginia Vassilevska Williams (2012, 2020): noch ein bisschen schneller

Timeline of matrix multiplication exponent

Year	Bound on omega	Authors
1969	2.8074	Strassen[1]
1978	2.796	Pan[11]
1979	2.780	Bini, Capovani, Romani[12]
1981	2.522	Schönhage[13]
1981	2.517	Romani[14]
1981	2.496	Coppersmith, Winograd[15]
1986	2.479	Strassen[16]
1990	2.3755	Coppersmith, Winograd[17]
2010	2.3737	Stothers[18]
2013	2.3729	Williams[19][20]
2014	2.3728639	Le Gall[21]
2020	2.3728596	Alman, Williams[3]
2022	2.37188	Duan, Wu, Zhou[2]

(`https://en.wikipedia.org/wiki/Computational_complexity_of_matrix_multiplication`)

Matrizenmultiplikation — geht es noch schneller?

- Coppersmith und Winograd (1990): $O\left(n^{2.376\ldots}\right)$
  - der konstante Faktor ist *groß*
- Virginia Vassilevska Williams (2012, 2020): noch ein bisschen schneller

Timeline of matrix multiplication exponent

Year	Bound on omega	Authors
1969	2.8074	Strassen[1]
1978	2.796	Pan[11]
1979	2.780	Bini, Capovani, Romani[12]
1981	2.522	Schönhage[13]
1981	2.517	Romani[14]
1981	2.496	Coppersmith, Winograd[15]
1986	2.479	Strassen[16]
1990	2.3755	Coppersmith, Winograd[17]
2010	2.3737	Stothers[18]
2013	2.3729	Williams[19][20]
2014	2.3728639	Le Gall[21]
2020	2.3728596	Alman, Williams[3]
2022	2.37188	Duan, Wu, Zhou[2]

(`https://en.wikipedia.org/wiki/Computational_complexity_of_matrix_multiplication`)

- unklar: Genügen $O\left(n^2\right)$ Operationen?

Teile und herrsche — engl. divide and conquer

- «zerhacke» Probleminstanz in kleinere Teile
  - mehr oder weniger viele
- bearbeite Teile rekursiv nach gleichen Verfahren
- konstruiere aus Teilergebnissen das Resultat für die ursprüngliche Eingabe

Was ist wichtig

- Das sollten Sie mitnehmen:
  - bei algorithmischen Problemen kann man überraschende Dinge tun
  - teile und herrsche
  - Rekursionsformel für Abschätzung von (z. B.) Laufzeiten
- Das sollten Sie üben:
  - in dieser Vorlesung: rechnen mit $O(\cdot)$, $\Theta(\cdot)$, $\Omega(\cdot)$
    - spätestens in kommenden Semestern: rekursive Algorithmen

Wo sind wir?

Ressourcenverbrauch bei Berechnungen

Groß-O-Notation

Matrizenmultiplikation

Asymptotisches Verhalten induktiv definierter Funktionen

Wo sind wir?

Karlsruher Institut für Technologie

Ressourcenverbrauch bei Berechnungen

Groß-O-Notation

Matrizenmultiplikation

Asymptotisches Verhalten induktiv definierter Funktionen

- Laufzeit von Teile-und-Herrsche-Algorithmen

Laufzeit von Teile-und-Herrsche-Algorithmen

- manchmal
  - Problem der Größe n zerhackt
    - in konstante Anzahl a von
    - Teilproblemen gleicher Größe n/b
    - sinnvollerweise $a \geq 1$ und $b > 1$
  - aus Teilergebnissen Gesamtergebnis zusammengesetzt

Laufzeit von Teile-und-Herrsche-Algorithmen

- manchmal
  - Problem der Größe n zerhackt
    - in konstante Anzahl a von
    - Teilproblemen gleicher Größe n/b
    - sinnvollerweise $a \geq 1$ und $b > 1$
  - aus Teilergebnissen Gesamtergebnis zusammengesetzt
- Abschätzung (z. B.) der Laufzeit $T(n)$ liefert Zusammenang in rekursiver Form, etwa

$$T(n) = a \cdot T\left(\frac{n}{b}\right) \qquad \text{für } n \geq 1$$
$$T(1) = 1$$

Laufzeit von Teile-und-Herrsche-Algorithmen

- manchmal
  - Problem der Größe n zerhackt
    - in konstante Anzahl a von
    - Teilproblemen gleicher Größe n/b
    - sinnvollerweise $a \geq 1$ und $b > 1$
  - aus Teilergebnissen Gesamtergebnis zusammengesetzt
- Abschätzung (z. B.) der Laufzeit $T(n)$ liefert Zusammenang in rekursiver Form, etwa

$$T(n) = a \cdot T\left(\frac{n}{b}\right) \qquad \text{für } n \geq 1$$
$$T(1) = 1$$

- (genau genommen $\lfloor n/b \rfloor$ oder $\lceil n/b + c \rceil$ oder …)

Laufzeit von Teile-und-Herrsche-Algorithmen

- manchmal
  - Problem der Größe n zerhackt
    - in konstante Anzahl a von
    - Teilproblemen gleicher Größe n/b
    - sinnvollerweise $a \geq 1$ und $b > 1$
  - aus Teilergebnissen Gesamtergebnis zusammengesetzt
- Abschätzung (z. B.) der Laufzeit $T(n)$ liefert Zusammenang in rekursiver Form, etwa

$$T(n) = a \cdot T\left(\frac{n}{b}\right) \qquad \text{für } n \geq 1$$
$$T(1) = 1$$

- (genau genommen $\lfloor n/b \rfloor$ oder $\lceil n/b + c \rceil$ oder …)
- gesucht: explizite Formel für $T(n)$

Explizite Form der Rekursionsform

$$T(n) = a \cdot T\left(\frac{n}{b}\right) \quad (n \geq 1)$$

$$T(1) = 1$$

- Hier: Beschränkung auf Potenzen von b, also $n = b^k$.
  (damit die Divisionen aufgehen)
- Dann ist die explizite Lösung $T(n) = n^{\log_b a}$.

$$T(b^k) = a^k$$

Explizite Form der Rekursionsform

$$T(n) = a \cdot T\left(\frac{n}{b}\right) \quad (n \geq 1)$$

$$T(1) = 1$$

- Hier: Beschränkung auf Potenzen von b, also $n = b^k$.
  (damit die Divisionen aufgehen)
- Dann ist die explizite Lösung $T(n) = n^{\log_b a}$.
- Beispiel: $a = 7, b = 2, k = 3$: (Also $T(n) = n^{\log_2 7}$, s. Strassen)
  $T(8) = T(2^3) = 7 \cdot T(2^2) = 7^2 \cdot T(2) = 7^3 \cdot T(1) = 7^3 = (2^{\log_2 7})^3 = 8^{\log_2 7}$

$$T(b^k) = a^k$$

Explizite Form der Rekursionsform

$$T(n) = a \cdot T\left(\frac{n}{b}\right) \quad (n \geq 1)$$

$$T(1) = 1$$

- Hier: Beschränkung auf Potenzen von b, also $n = b^k$.
  (damit die Divisionen aufgehen)
- Dann ist die explizite Lösung $T(n) = n^{\log_b a}$.

$$T(b^k) = a^k$$

- Beispiel: $a = 7, b = 2, k = 3$: (Also $T(n) = n^{\log_2 7}$, s. Strassen)
  $T(8) = T(2^3) = 7 \cdot T(2^2) = 7^2 \cdot T(2) = 7^3 \cdot T(1) = 7^3 = (2^{\log_2 7})^3 = 8^{\log_2 7}$
- Beispiel: $a = 8, b = 2, k = 3$: (Also $T(n) = n^{\log_2 8} = n^3$)
  $T(8) = T(2^3) = 8 \cdot T(2^2) = \ldots = 8^3$.

Explizite Form der Rekursionsform

$$T(n) = a \cdot T\left(\frac{n}{b}\right) \quad (n \geq 1)$$

$$T(1) = 1$$

- Hier: Beschränkung auf Potenzen von b, also $n = b^k$.
  (damit die Divisionen aufgehen)
- Dann ist die explizite Lösung $T(n) = n^{\log_b a}$.

$$T(b^k) = a^k$$

- Beweis durch vollständige Induktion
  - **IA:** Für $k = 0$ ist $n^{\log_b a} = (b^0)^{\log_b a} = 1 \overset{\text{ex def.}}{=} T(b^0)$.
  - **IS:** $k \rightsquigarrow k+1$:
    IV: $T(b^k) = (b^k)^{\log_b a}$, **z.z.** $T(b^{k+1}) = (b^{k+1})^{\log_b a}$
    Es gilt $T(b^{k+1}) \overset{\text{ex def.}}{=} aT(b^k) \overset{\text{IV}}{=} a \cdot (b^k)^{\log_b a} = b^{\log_b a} \cdot (b^k)^{\log_b a} = (b \cdot (b^k))^{\log_b a} = (b^{k+1})^{\log_b a}$

Zusammenfassung

- Komplexitätsmaße
  - Laufzeitbedarf
  - Speicherplatzbedarf
- Abschätzung asymptotischen Wachstums
  - «bis auf konstante Faktoren»
  - nach oben mit $O(\cdot)$
  - nach oben und unten mit $\Theta(\cdot)$
  - nach unten mit $\Omega(\cdot)$
- Teile und Herrsche (divide and conquer)
  - am Beispiel Matrixmultiplikation
- Explizite Form für Aufwand von Teile-und-Herrsche