universität freiburg

Algorithmen und Datenstrukturen SS 2025

Vorlesung 4: O-Notation

Dienstag, 13. Mai 2025

Prof. Dr. Hannah Bast
Professur für Algorithmen und Datenstrukturen
Institut für Informatik
Albert-Ludwigs-Universität Freiburg

Blick über die Vorlesung heute

- Organisatorisches
  - Erfahrungen mit dem Ü3 — Spaß mit I/E-Sequenzen
  - Noch mal ZeroOneSort — Doch vergleichsbasiert?
  - Vorgucker auf das Ü4 — Noch einmal Theorie
- O-Notation
  - Grundlagen der O-Notation — O, Ω, Θ, o, ω
  - Definition über Grenzwerte — $\lim_{n\to\infty} f(n)/g(n)$
  - Diskussion über den Sinn — wichtig zum Verständnis

Erfahrungen mit dem Ü3 1/3

- Auszüge aus Ihrem Feedback [halbwegs repräsentativ]

  "Vorlesung sehr hilfreich und gut rübergebracht"

  "Vorlesung war deutlich komplexer, aber dennoch verständlich"

  "Die vielen Beispiele zur I/E-Sequenz waren sehr hilfreich"

  "Ich fand die Hilfestellungen auf dem Übungsblatt sehr gut"

  "Blatt im Prinzip nicht schwer, aber veranlasst zu prokrastinieren"

  "Darstellung 'Was ist ein Beweis' hat sehr geholfen"

  "Bei Aufgabe 1 das Gefühl, zuviel geschwafelt zu haben"

  "Ich bin mir nie sicher, wann ein Beweis ausreichend ist"

  "Ich empfand das Blatt sehr schwer; mehr freiwillige Aufgaben"

  "Weniger Panikmache und mehr Motivation fände ich schön"

Erfahrungen mit dem Ü3 2/3

- Feedback zur Absolventenquote von nur 15%
  - "Informatik ist schwerer als sich die meisten das vorstellen"
  - "Viele studieren das mit den falschen Vorstellungen"
  - "Viele benutzen das Informatikstudium zum Zwischenparken"
  - "Studium zu anstrengend für das 'klassische Studentenleben'"
  - "Viele unterschätzen Mathe und können nicht mehr mithalten"
  - "Informatik hat keinen Numerus Clausus (finde ich super)"
  - "Regelmäßige disziplinierte Aufarbeitung des Stoffes nötig"
  - "Es gibt keine dummen Menschen, nur faule"
  - "Hängt damit zusammen, dass in D studieren fast nichts koscht"
  - "Die Statistik hat gezeigt, dass sich die meisten überschätzen"
  - "Skills + Sinn + Support = Überleben, fehlt eines dann 200 → 30"

Erfahrungen mit dem Ü3 3/3

- Vergleichsbasiertes ZeroOneSort

$2^n \ll n!$

  - Was wenn man den Anfang von ZeroOneSort so codiert:

```
numOnes = 0
for x in numbers:
   if x == 1: numOnes += 1
```

1, 0, 0, 1, 1, 0

I E E I I E

Nach unserer Definition wäre das "vergleichsbasiert" (jede Eingabe bekommt ihre eigene I/E-Sequenz → 2^n viele)

Bezeichnung nicht glücklich in dem Fall, weil nur mit numOnes weitergerechnet wird (nur n + 1 Möglichkeiten)

  - Widerspricht das nicht der unteren Schranke von $n \log n$?

Es gibt hier nur 2^n verschiedene Eingaben, also braucht man auch nur 2^n verschiedene Permutationen und es reichen deshalb $\log_2 2^n = n$ Verzweigungen (I/E)

Vorgucker auf das Ü4

- Nochmal Theorie, das Ü5 wird dann wieder praktisch

  A1 Einfache Aufgabe zu Logarithmus und Θ-Notation

  Hier sollen Sie die "zu Fuß" Definition benutzen

  A2 Weiterführende Aufgabe zu O, Ω, Θ, o, ω

  Hier können Sie die Definition über den Grenzwert benutzen

  A3 Ein abstrakterer mathematischer Beweis

  Wenn man es verstanden hat, ist der Beweis nicht schwer

  A4 Laufzeitbestimmung von einem gegebenen Programm

  Nette Aufgabe mit etwas Knobelcharakter

  Nutzen Sie die Gelegenheit zum Üben, es hilft Ihnen bei den weiteren Vorlesungen und sowas kommt immer in der Klausur

O-Notation – Grundlagen 1/12

- Erinnerung
  - Wir haben jetzt mehrfach die Laufzeit $T(n)$ in Abhängigkeit von der Eingabegröße abgeschätzt
  - Zum Beispiel hatten wir, für $n \geq$ irgendeine Konstante n_0 :

    Laufzeit von MinSort liegt in $[C' \cdot n^2, C \cdot n^2]$

    Laufzeit von MergeSort liegt in $[C' \cdot n \cdot \log n, C \cdot n \cdot \log n]$

    Laufzeit von ZeroOneSort liegt in $[C' \cdot n, C \cdot n]$

    Laufzeit von vergleichsbasiertem Sortieren ist $\geq C \cdot n \cdot \log n$
  - Die Werte der Konstanten waren uns dabei egal und auch wenn diese Schranken erst ab einem bestimmten n gelten

    Es war aber ziemlich anstrengend, diese Konstanten (aus anderen Konstanten) auszurechnen und mitzuschleppen

O-Notation – Grundlagen 2/12

- Motivation
  - In Zukunft wollen wir es uns einfacher machen können und **formal korrekt** so etwas schreiben wie:

    Die Laufzeit von MinSort ist $\Theta(n^2)$

    Die Laufzeit von MergeSort ist $\Theta(n \cdot \log n)$

    Die Laufzeit von ZeroOneSort ist $\Theta(n)$

    Vergleichsbasiertes Sortieren hat Laufzeit $\Omega(n \cdot \log n)$

  - Wichtig für das Verständnis: was darf man in dieser Notation weglassen (z.B. die Konstanten) und was nicht und warum?

    Das sollte am Ende der Vorlesung heute klar geworden sein und werden wir in den folgenden Vorlesungen oft anwenden

O-Notation – Grundlagen 3/12

- Welcher Algorithmus ist besser?
  - Algorithmus A braucht $C_1 \cdot n \cdot \log n$ Operationen
  - Algorithmus B braucht $C_2 \cdot n^2$ Operationen
  - Die genaue Antwort: **kommt drauf an** (auf C_1, C_2 und n)

    Wir schauen uns ein Schaubild an, für $C_1 = 10$ und $C_2 = 1$

    Für egal welche C_1 und C_2 gilt: ab einem bestimmten genügend großen n ist A auf jeden Fall besser als B
  - Mathematisch betrachtet, interessiert uns das Verhalten für $n \to \infty$ … das nennt man **asymptotische** Laufzeitanalyse
  - In der Praxis sind die Konstanten aber trotzdem r  , wir kommen auf Folien 28 und 29 darauf zurück

O-Notation – Grundlagen 4/12

- Vorbetrachtung
  - Wir betrachten Funktionen $f : \mathbf{N} \to \mathbf{R}$

    N = die natürlichen Zahlen ... typisch: Eingabegröße

    R = die reellen Zahlen ... typisch: Laufzeit

    Uns reicht, wenn $f(n) > 0$ für $n \geq n_0$ ... darunter darf f negativ sein, und das kommt bei Abschätzungen auch manchmal raus
  - Beispiele

    $f(n) = 3 \cdot n + 3$

    $f(n) = 2 \cdot n \cdot (\log_2 n - 5)$

    $f(n) = n^2 / 10$

    $f(n) = n^2 + 3 \cdot n \cdot \log_2 n - 4 \cdot n$

$n < 2^5 = 32 \Rightarrow g(n) < 0$

O-Notation – Grundlagen 5/12

- Groß-O, Definition
  - Seien g und f zwei Funktionen $\mathbf{N} \to \mathbf{R}$
  - **Intuitiv:** Man sagt g ist Groß-O von f ...

    wenn g "höchstens so stark wächst wie" f
  - **Informal:** Man schreibt $g = O(f)$ ...

    wenn ab irgendeinem Wert n_0 für all $n \geq n_0$

    $g(n) \leq C \cdot f(n)$ für irgendeine Konstante C
  - **Formal:** für eine Funktion $f : \mathbf{N} \to \mathbf{R}$ ist ...

    $O(f) = \{ g : \mathbf{N} \to \mathbf{R} \mid \exists C > 0 \; \exists n_0 \in \mathbf{N} \; \forall n \geq n_0 \;\; g(n) \leq C \cdot f(n) \}$

    dabei heißt $\exists$ = "es existiert ..." und $\forall$ = "für alle ..."

FRAGE AUS DEM CHAT:
müsste man nicht schreiben:
$O(g) = O(f)$

korrekt wäre $g \in O(f)$

O-Notation – Grundlagen 6/12

- Groß-O, Beispiel
  - Sei $g(n) = 5 \cdot n + 7$ und $f(n) = n$
  - Dann ist $g = O(f)$ bzw. man schreibt $5 \cdot n + 7 = O(n)$
  - **Intuitiv:** $5 \cdot n + 7$ wächst höchstens "linear"
  - Beweis unter Verwendung der Definition von O :

Beweis 1: $5 \cdot n + 7 \overset{\leq 7n}{\leq} 12 \cdot n \leq \ldots \cdot n$

$g(n)$ für $n \geq 1$ $g(n)$

⇒ für $n_0 = 1$, $C = 12$

Beweis 2: $5 \cdot n + 7 \leq 5 \cdot n + n \leq 6 \cdot n$

$g(n)$ $n \geq 7$ $g(n)$

⇒ für $n_0 = 7$, $C = 6$

O-Notation – Grundlagen 7/12

- Es zählt "Wachstumsrate", nicht absolute Werte
  - Für zwei Funktionen kann ohne Probleme gelten

    $g = O(f)$ g wächst **nicht stärker** als f

    $g > f$ g ist überall **echt größer** als f
  - Zum Beispiel g und f von der Folie vorher

$g(n) = 5 \cdot n + 7$

$f(n) = n$

$g = O(f)$ ... siehe vorherige Folie

ABER auch

$g(n) = 5 \cdot n + 7 > n = f(n)$

$\forall n \in \mathbb{N}$

O-Notation – Grundlagen 8/12

- Groß-Omega, Definition + Beispiel
  - **Intuitiv:** Man sagt g ist Groß-Omega von f **...**

    ... wenn g "mindestens so stark wächst wie" f

    Also wie Groß-O, nur mit "mindestens" statt "höchstens"
  - **Formal:** Für eine Funktion $f : \mathbf{N} \to \mathbf{R}$ ist

    $$\Omega(f) = \{ g : \mathbf{N} \to \mathbf{R} \mid \exists C > 0 \; \exists n_0 \in \mathbf{N} \; \forall n \geq n_0 \;\; g(n) \geq C \cdot f(n) \}$$
  - Zum Beispiel $5 \cdot n + 7 = \Omega(n)$ einziger Unterschied zu O
  - Beweis unter Verwendung der Definition von Ω :

$g(n) = 5 \cdot n + 7 \geq 1 \cdot n \geq \ldots \cdot n$ ($= g(n)$)

$\Rightarrow$ für $n_0 = 1$ und $C = 1$

O-Notation – Grundlagen 9/12

- Groß-Theta, Definition + Beispiel
  - **Intuitiv:** Man sagt g ist Theta von f **...**

    ... wenn g "genauso so stark wächst wie" f
  - **Formal:** Für eine Funktion $f : \mathbf{N} \to \mathbf{R}$ ist

    $\Theta(f) = O(f) \cap \Omega(f)$ = die Schnittmenge von $O(f)$ und $\Omega(f)$

    Wächst "höchstens so stark" **und** "mindestens so stark"
  - Zum Beispiel $5 \cdot n + 7 = \Theta(n)$
  - Beweis unter Verwendung der Definition von Θ :

Folie 12: $g = O(f)$

Folie 14: $g = \Omega(f)$

$\Longrightarrow g \in O(f) \cap \Omega(f) = \Theta(f)$

O-Notation – Grundlagen 10/12

- Es gibt auch noch o (Klein-O) und ω (Klein-Omega)
  - Die braucht man in der Informatik seltener
  - Hier die Definitionen für $f : \mathbf{N} \to \mathbf{R}$

    $o(f) = \{ g : \forall C > 0 \; \exists n_0 \in \mathbf{N} \; \forall n \geq n_0 \; g(n) \leq C \cdot f(n) \}$

    $\omega(f) = \{ g : \forall C > 0 \; \exists n_0 \in \mathbf{N} \; \forall n \geq n_0 \; g(n) \geq C \cdot f(n) \}$
  - Intuitiv bedeutet

    $g = o(f)$: g wächst strikt langsamer als f

    $g = \omega(f)$: g wächst strikt schneller als f

    Insbesondere ist die Schnittmenge leer: $o(f) \cap \omega(f) = \emptyset$

O-Notation – Grundlagen 11/12

- Intuitive Zusammenfassung
  - Die Operatoren O , Ω , Θ , o , ω sind auf Funktionen, was die Operatoren $\leq$, $\geq$, $=$, $<$, $>$ auf Zahlen sind:

    O entspricht $\leq$

    Ω entspricht $\geq$

    Θ entspricht $=$

    o entspricht $<$

    ω entspricht $>$

  - Es macht übrigens keinen Sinn so etwa zu sagen wie: die Laufzeit ist mindestens $O(n \cdot \log n)$

    Stattdessen: die Laufzeit ist $\Omega(n \cdot \log n)$

das wäre so wie "die Zahl ist mindestens ≤ 7"

O-Notation – Grundlagen 12/12

- Weitere Eigenschaften
  - Viele Eigenschaften von $\leq$, $\geq$, $=$, $<$, $>$ gelten auch sinngemäß genauso für O , Ω , Θ , o , ω
  - Zum Beispiel: Transitivität

    $x < y \wedge y \leq z \Rightarrow x < z$

    $f = o(g) \wedge g = O(h) \Rightarrow f = o(h)$
  - Zum Beispiel: Additivität

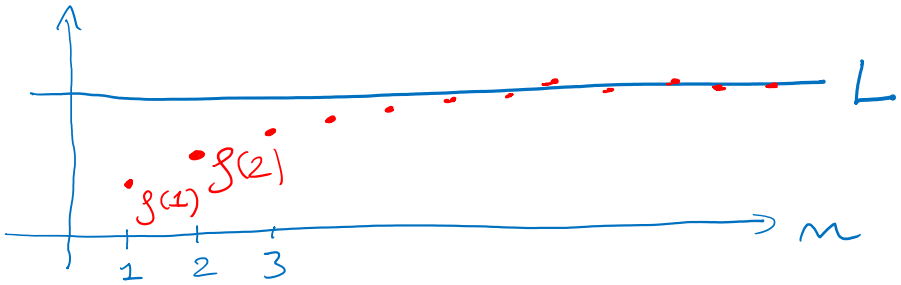
    $x_1 \leq y_1 \wedge x_2 \leq y_2 \Rightarrow x_1 + x_2 \leq y_1 + y_2$

    $f_1 = O(g_1) \wedge f_2 = O(g_2) \Rightarrow f_1 + f_2 = O(g_1 + g_2)$

Gute Zusatzaufgabe für die, die Lust auf mehr haben oder später zur Klausurvorbereitung

Bestimmung über Grenzwerte 1/8

- Grenzwertbegriff
  - Die Definitionen von O , Ω , Θ, ... erinnern sehr stark an den **Grenzwertbegriff** aus der **Analysis**
  - **Definition:** Eine unendliche Folge $f_1, f_2, f_3, \ldots$ hat einen Grenzwert L, wenn für alle $\varepsilon > 0$ ein $n_0 \in \mathbf{N}$ existiert so dass für alle $n \geq n_0$ gilt dass $|f_n - L| \leq \varepsilon$
  - In Symbolen schreibt man dann $\lim_{n\to\infty} f_n = L$
  - Eine Funktion $f : \mathbf{N} \to \mathbf{R}$ kann man genauso gut als Folge $f(1), f(2), f(3), \ldots$ auffassen und schreibt $\lim_{n\to\infty} f(n) = L$



Bestimmung über Grenzwerte 2/8

- Beispiel für einen Beweis von einem Grenzwert
  (sollten Sie eigentlich in Mathe 1 schon mal gesehen haben)
  - Es gilt: $\lim_{n\to\infty} 1/n = 0$

$\frac{1}{\infty} = 0$" ist kein Beweis

Erst mal für ein festes $\varepsilon > 0$

$\varepsilon = \frac{1}{100}$ $\quad n_0 = ?$ $\quad \forall n \geq n_0 : \left|\frac{1}{n} - 0\right| \leq \frac{1}{100}$

≥ 0, $\frac{1}{n}$

$n_0 := 100 : \forall n \geq n_0 : \frac{1}{n} \leq \frac{1}{100} = \varepsilon$

$n \geq 100$

Jetzt der Beweis:

$\varepsilon > 0$ beliebig.

$n_0 := \lceil \frac{1}{\varepsilon} \rceil : \forall n \geq n_0 : \frac{1}{n} \leq \frac{1}{1/\varepsilon} = \varepsilon$

$n \geq \frac{1}{\varepsilon}$

$n \geq \lceil 1/\varepsilon \rceil \geq \frac{1}{\varepsilon}$

Bestimmung über Grenzwerte 3/8

- Satz
  - Seien $f, g : \mathbf{N} \to \mathbf{R}$ und der Grenzwert $\lim_{n\to\infty} f(n)/g(n)$ existiert (evtl. ist er ∞) ... dann gelten:

    (1) $f = O(g) \Leftrightarrow \lim_{n\to\infty} f(n)/g(n) < \infty$

    (2) $f = \Omega(g) \Leftrightarrow \lim_{n\to\infty} f(n)/g(n) > 0$

    (3) $f = \Theta(g) \Leftrightarrow \lim_{n\to\infty} f(n)/g(n) > 0$ und $< \infty$

    (4) $f = o(g) \Leftrightarrow \lim_{n\to\infty} f(n)/g(n) = 0$

    (5) $f = \omega(g) \Leftrightarrow \lim_{n\to\infty} f(n)/g(n) = \infty$
  - Wir beweisen auf der nächsten Folie Aussage (1)

    Die Beweise für die anderen Aussagen sind sehr ähnlich und ebenfalls sehr gute Übungsaufgaben für die Klausur

Bestimmung über Grenzwerte 4/8

- Beweis von: $f = O(g) \Leftrightarrow \lim_{n\to\infty} f(n)/g(n) < \infty$

"$\Rightarrow$"

$f = O(g) \Rightarrow \exists c > 0\ \exists n_0 \in \mathbb{N}\ \forall n \geq n_0\ f(n) \leq c \cdot g(n)$ (Def. von O)

$\Rightarrow \exists c > 0\ \exists n_0 \in \mathbb{N}\ \forall n \geq n_0\ \frac{f(n)}{g(n)} \leq c$

$\Rightarrow \lim_{n\to\infty} f(n)/g(n) \leq c < \infty$ ∎

"$\Leftarrow$"

$\lim_{n\to\infty} f(n)/g(n) = c$ für irgendein $c \in \mathbb{R}$

z.B. $\varepsilon = 1 \Rightarrow \exists n_0 \in \mathbb{N}\ \forall n \geq n_0\ f(n)/g(n) \leq c + 1$ (Def. v. lim)

$\Rightarrow \exists n_0 \in \mathbb{N}\ \forall n \geq n_0\ f(n) \leq (c+1) \cdot g(n)$

$\Rightarrow f = O(g)$ (Def. v. O) ∎

Bestimmung über Grenzwerte 5/8

- Was darf man ohne Beweis annehmen?
  - Gute Frage, aber da gibt es keine klare Regel

    Im Zweifelsfall lieber mehr beweisen als weniger
  - Sie müssen $\lim_{n\to\infty} f(n) = \infty$ nicht beweisen, wenn f ein Polynom, ein Logarithmus oder eine Exponentialfunktion ist

    z.B. $\lim_{n\to\infty} n^3 = \infty$ oder $\lim_{n\to\infty} 3^n = \infty$
  - Durch Kombination ergeben sich viele weitere Grenzwerte

    z.B. $\lim_{n\to\infty} n^2 \cdot \log n = \infty$ oder $\lim_{n\to\infty} 2^{-n} = 0$
  - Schwierigere Fälle lassen sich oft mit der **Regel von L'Hôpital** auf einfachere Grenzwerte zurückführen

    Siehe nächste Folie

Bestimmung über Grenzwerte 6/8

- Regel von L'Hôpital
  - Seien $f, g : \mathbf{N} \to \mathbf{R}$ wie gehabt
  - Es existieren die ersten Ableitungen f' und g', sowie der Grenzwert $\lim_{n\to\infty} f'(n)/g'(n)$
  - Dann gilt:

    $\lim_{n\to\infty} f(n)/g(n) = \lim_{n\to\infty} f'(n)/g'(n)$

  - Pingel-Alert: 

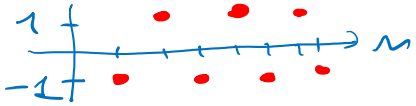
    Es ist **nicht** korrekt, $\lim_{n\to\infty} f(n)/g(n) = \lim_{n\to\infty} f'(n)/g'(n)$ hinzuschreiben und **dann erst** zu schauen, ob der Grenzwert auf der rechten Seite überhaupt existiert

    Wenn der Grenzwert auf der rechten Seite nicht existiert, stimmt auch die Gleichheit nicht

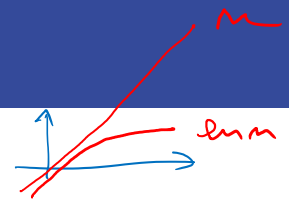
UND
$\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = 0$
ODER
$\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = \infty$

Beispiel für eine Fkt $f : \mathbb{N} \to \mathbb{R}$ wo $\lim_{n\to\infty} f(n)$ NICHT exe.

$f(n) = (-1)^n$



Bestimmung über Grenzwerte 7/8



- Beispiel: Grenzwert mit L'Hôpital
  - Was ist $\lim_{n\to\infty} (\ln n)/n$?

$\lim_{n\to\infty} \ln n = \infty$

$\lim_{n\to\infty} n = \infty$

$\frac{\infty}{\infty} = ?$

$f(n) = \ln n \Rightarrow f'(n) = \frac{1}{n}$

$g(n) = n \Rightarrow g'(n) = 1$

$$\lim_{n\to\infty} \frac{f'(n)}{g'(n)} = \lim_{n\to\infty} \frac{1/n}{1} = \lim_{n\to\infty} \frac{1}{n} = 0$$

ALSO $\lim_{n\to\infty} \frac{\ln n}{n} = 0$

klein-o

unbesondere $\ln n = o(n)$

oder $\ln = O(n)$

↖ groß-O

Bestimmung über Grenzwerte 8/8

- Hand-Waving
  - Technik 1 (Anfänger)

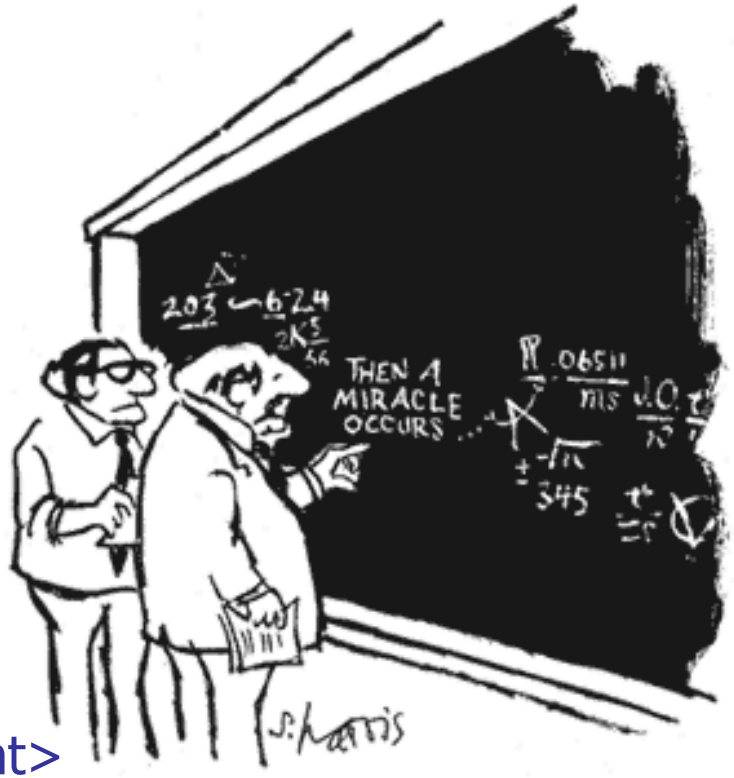
    … viele Worte …

    man schreibt hin,
    was rauskommen soll
  - Technik 2 (Profis)

    "sieht man doch"

    "trivial"

    <einschüchterndes Argument>
  - So oder so in aller Regel **falsch**



"I THINK YOU SHOULD BE MORE EXPLICIT HERE IN STEP TWO."

Diskussion 1/4

- Terme niedriger Ordnung
  - **Theorem:** Seien $f_1, \ldots, f_k$ und $g : \mathbf{N} \to \mathbf{R}$, dann gilt

    1. $f_1 + \ldots + f_k = O(g) \Leftrightarrow f_i = O(g)$ für alle $i = 1, \ldots, k$
    2. $f_1 + \ldots + f_k = \Theta(g) \Leftrightarrow f_i = O(g)$ für alle $i = 1, \ldots, k$ **und** $f_i = \Theta(g)$ für mindestens ein i

    Noch eine gute Übungsaufgabe für die Klausur

  - Damit können wir in Zukunft komplexe Ausdrücke leicht vereinfachen, zum Beispiel:

    $\in \Theta(n^2)$ $\in O(n^2)$ $\in O(n^2)$

    $n^2 + 7n - 5 = O(n^2)$ und auch $= \Theta(n^2)$

    $\in \Theta(n \cdot \log n)$ $\in O(n \cdot \log n)$

    $n \cdot \log n + 12n = O(n \cdot \log n)$ und auch $= \Theta(n \cdot \log n)$

Diskussion 2/4

- Vorsicht bei asymptotischer Analyse
  - Die O-Notation schaut sich das Verhalten der Funktionen an, wenn $n \to \infty$ geht (es interessieren nur die $n \geq n_0$)
  - Asymptotische Analyse sagt deswegen **nichts** über das Verhalten bei "kleinen" Eingabegrößen ($n < n_0$) aus
  - Für $n < 2$ oder $n < 10$ ist das egal, da wird schon nichts Schlimmes passieren
  - Aber das n_0 ist nicht immer so klein ... siehe nächste Folie

Diskussion 3/4

- Ein Beispiel, bei dem das n_0 nicht ganz so klein ist
  - Algorithmus A hat Laufzeit $f(n) = 80 \cdot n$
  - Algorithmus B hat Laufzeit $g(n) = 2 \cdot n \cdot \log_2 n$
  - Dann ist $f = O(g)$ und sogar $f = o(g)$

    Insbesondere für alle $n \geq$ irgendein n_0 : $f(n) \leq g(n)$

    Das heißt, A ist **asymptotisch schneller** als B
  - Allerdings gilt für $n_0 = 2^{40} \approx 10^{12} = 1$ Tera : $2^{10} \approx 1000 \approx (1000)^4$

    Deutsch: „Billionen" Englisch: „Trillion"

$\forall n < n_0 = 2^{40}$:

$$g(n) = 2 \cdot n \cdot \underbrace{\log_2 n}_{< \log_2 2^{40} = 40} < 80 \cdot n = f(n)$$

Diskussion 4/4

- Mehrere Variablen
  - Es kommt öfter mal vor, dass die Laufzeit (oder eine andere Größe) von mehr als einer Variablen abhängt
  - Zum Beispiel von der Eingabegröße n und der Anzahl m der verschiedenen Elemente in der Eingabe
  - Dann würden wir gerne so etwas schreiben wie

    $O(n + m \cdot \log m)$

  - Wir können unsere Definitionen leicht auf Funktionen $\mathbf{N}^2 \to \mathbf{R}$ verallgemeinern, zum Beispiel für Groß-O:

    $$O(f) = \{ g : \mathbf{N}^2 \to \mathbf{R} \mid \exists\, n_0 \in \mathbf{N}\ \ \exists\, C > 0\ \ \forall\, n, m \geq n_0 \quad g(n, m) \leq C \cdot f(n, m) \}$$

Literatur / Links

- O-Notation / Ω-Notation / Θ-Notation
  - In Mehlhorn/Sanders:

    2.1 Asymptotic Notation
  - In Wikipedia

    http://en.wikipedia.org/wiki/Big_O_notation

    http://de.wikipedia.org/wiki/Landau-Symbole