Algorithmen und Datenstrukturen (ESE)
Entwurf, Analyse und Umsetzung von Algorithmen (IEMS)
WS 2012 / 2013

Vorlesung 3, Dienstag 6. November 2012
(O-Notation, Theta, Omega)

Prof. Dr. Hannah Bast
Lehrstuhl für Algorithmen und Datenstrukturen
Institut für Informatik
Universität Freiburg

UNI FREIBURG

Blick über die Vorlesung heute

- Organisatorisches
  - Ihre Erfahrungen mit dem Ü2 (Induktion + Laufzeit HeapSort)
  - Punktevergabeschema / Qualität Ihrer Programme
- O-Notation
  - Motivation
  - Klassische Definition mit C und n_0
  - Einfachere Bestimmung über den $\lim_{n \to \infty}$
  - $f = O(g)$ heißt nicht immer f ist besser
  - Übungsblatt 3: ein paar Beweise / Rechenaufgaben dazu

    So was in der Art kommt meistens auch in der Klausur !

Erfahrungen mit dem Ü2 (Beweise, iiih)

- Zusammenfassung / Auszüge Stand 6. November 13:37
  - Aufgaben 2 und 3 waren einfach / bekannt aus Mathe 1
  - Aufgabe 4 war komisch / unklar was wie getan werden sollte

    Das "im schlechtesten Fall" hatte einigen Probleme gemacht

    Ich werde die Lösung gleich mal skizzieren
  - Einheitliches Checkstyle für die ganze Fakultät

    Sie sprechen mir aus der Seele ...
  - Jetzt dann aber lieber wieder mehr Programmieren

    Sorry, Ü3 nochmal theoretisch, aber ab Ü4 dann wieder ...
  - Das Leben ist deterministisch und hat keinen höheren Sinn

    Sinn des Lebens: 404 file not found

Punktevergabe Schema

- Dazu gibt es jetzt einen Link auf dem Wiki
  - ... das schauen wir uns jetzt mal an
  - Wichtigster Punkt dabei:

    Ihr Programm muss wenigstens kompilieren !

    (mit ant compile bzw. make compile ... sonst braucht es sich Ihr Tutor / Ihre Tutorin nicht mal anzuschauen)

    Und dann natürlich lauffähig sein !

    (und nicht mit segmentation fault o.ä. abstürzen)

Ü2, Aufgabe 4 (repair Kosten $\geq C_1 \cdot n \cdot \log_2 n$)

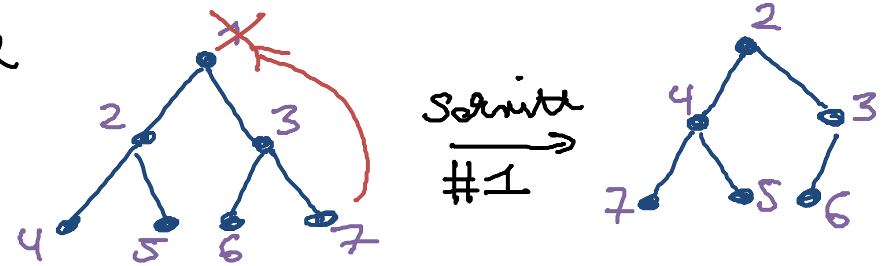
- Hier die Grundidee eines möglichen Beweises
  - Vorgehensweise: ein Beispiel konstruieren, für das $\geq C_1 \cdot n \cdot \log_2 n$ Operationen gebraucht werden
  - Idee: es reicht zu zeigen, dass die ersten $n/2$ oder so repairs $\geq C' \cdot \log_2 n$ Operationen brauchen

für beliebige n, über Level $d-1$ argumentieren. Da sind immer $\geq \frac{n}{4}$ Elemente

Erstmal Annahme $n = 2^d - 1$

und es stehen einfach die Zahlen $1, 2, 3, \ldots$ im Heap (HE gilt dann)

Beispiel für $d = 3$, $n = 7$



für $n \geq n_0$ $d-1 \geq d/2$

$c' \cdot (d-1)$

Die ersten $\underbrace{2^{d-1}}_{=\frac{n+1}{2}}$ repairs brauchen $\geq C' \cdot \underbrace{d}_{=\log_2(n+1)}$ Operationen

$\frac{\log_2(n+1)}{2}$

Also dafür schon $\geq \frac{n+1}{2} \cdot c' \cdot \log_2(n+1)$ Operationen

$\geq \underbrace{\frac{c'}{2}}_{=:C_1} \cdot n \cdot \log_2 n$ $c'/4$

QED

O-Notation — Motivation

- Primär interessiert uns oft
  - das "Wachstum" einer Funktion, z.B. einer Laufzeit $T(n)$
  - Die Werte der Konstanten (z.B. A'') sind dabei oft sekundär
  - Und auch, wenn die Schranken erst ab $n \geq \ldots$ gelten
  - Zum Beispiel war beim Sortieren interessant, dass
    - die Laufzeit von MinSort "wächst wie" n^2
    - aber die Laufzeit von HeapSort "wächst wie" $n \cdot \log n$
  - Das wollen wir jetzt formaler machen, damit wir in Zukunft etwas schreiben bzw. sagen können wie:
    - Die Laufzeit des Algorithmus ist $O(n)$ "O von n"
    - Die Laufzeit des Algorithmus ist $\Omega(n)$ "Omega von n"
    - Die Laufzeit des Algorithmus ist $\Theta(n)$ "Theta von n"

O-Notation — Definition 1/5

- Vorweg
  - Wir betrachten Funktionen $f : N \rightarrow R$
    - N = die natürlichen Zahlen ... typisch: Eingabegröße
    - R = die reellen Zahlen ... typisch: Laufzeit
  - Zum Beispiel
    - $f(n) = 3 \cdot n$
    - $f(n) = 2 \cdot n \cdot \log n$
    - $f(n) = n^2 / 10$
    - $f(n) = n^2 + 3 \cdot n \cdot \log n - 4 \cdot n$

O-Notation — Definition 2/5

- Groß-O, Definition
  - Seien g und f zwei Funktionen $N \to R$
  - **Intuitiv:** Man sagt g ist Groß-O von f ...
    - ... wenn g "höchstens so stark wächst wie" f
    - es zählt die "Wachstumsrate", nicht die absoluten Werte!
  - **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 $f : N \to R$ ist ...
    - $O(f) = \{ g : N \to R \mid \exists n_0 \in N\ \exists C>0\ \forall n \geq n_0\ \ g(n) \leq C \cdot f(n) \}$
    - dabei heißt $\exists$ = "es existiert ..." und $\forall$ = "für alle ..."

Achtung z.B.

$3 \cdot n = O(n)$

also nicht $\leq$ im „wörtlichen" Sinne, sondern nur in Bezug auf Wachstum

also eigentlich müsste man schreiben $g \in O(f)$

O-Notation — Definition 3/5

- 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 :

Auch hier wieder: $g(n) \geq f(n)$ aber trotzdem $g = O(f)$

Es muss gezeigt werden: $\exists n_0, C : 5 \cdot n + 7 \leq C \cdot n \;\; \forall n \geq n_0$

Beweis: $5 \cdot n + 7 \leq 5 \cdot n + n$ (für $n \geq 7$)

$\leq 6 \cdot n$

$n_0 = 7$, $C = 6$

Alternativer Beweis: $5 \cdot n + 7 \leq 5 \cdot n + 7 \cdot n$ (für $n \geq 1$)

$\leq 12 \cdot n$

$n_0 = 1$, $C = 12$

O-Notation — Definition 4/5

- 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 $f : N \rightarrow R$ ist

    bei O: "≤"

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

$$\underbrace{5 \cdot n + 7}_{g(n)} \geq \underbrace{n}_{f(n)} \qquad n_0 = 1, \; C = 1$$

O-Notation — Definition 5/5

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

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

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

    $X \cap Y$: die Schnittmenge von X und Y
  - Zum Beispiel $5 \cdot n + 7 = \Theta(n)$
  - Beweis unter Verwendung der Definition von Θ :

approximiertes Schnittmengensymbol $\cap$ in PPT

Wir hatten O(f) und O(g) ja formal als Mengen (von Funktionen) definiert.

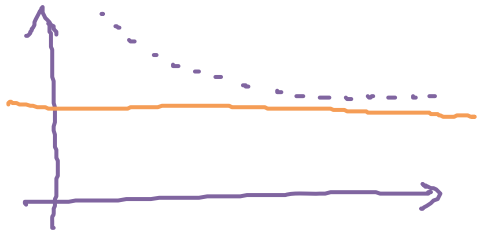
$g = O(f)$ vorletzte Folie

$g = \Omega(f)$ letzte Folie

$\Rightarrow g = \Theta(f)$

O-Notation — Grenzwerte 1/4

- In den bisherigen Beispielen ...
  - ... haben wir die Zugehörigkeit zu O(...) etc. gewissermaßen "zu Fuß" bewiesen, indem wir explizit das n_0 und das C bestimmt haben
  - Die Definitionen erinnern aber sehr 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 $n_0 \in N$ 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 $f : N \to R$ kann man genauso gut als Folge $f(1), f(2), f(3), \ldots$ auffassen und schreibt $\lim_{n\to\infty} f(n) = L$

O-Notation — Grenzwerte 1/3

- Beispiel für einen Beweis von einem Grenzwert
  (sollten Sie eigentlich in Mathe 1 schon mal gesehen haben)

Ingenieursmäßig $\frac{1}{\infty} = 0$

$$\lim_{n \to \infty} 2 + \frac{1}{n} = 2$$

zu zeigen: für jedes vorgegebene (beliebige) $\varepsilon > 0$
gibt es ein n_0, so dass $\forall n \geq n_0$ gilt:

$$\underbrace{\left| 2 + \frac{1}{n} - 2 \right|}_{1/n} \leq \varepsilon$$

z.B. für $\varepsilon = \frac{1}{10}$

$\frac{1}{n} \leq \varepsilon$ ab $n \geq 10$

allgemein $n_0 = \lceil \frac{1}{\varepsilon} \rceil$

dann für $n \geq n_0$:

$$\left| \frac{1}{n} \right| = \frac{1}{n} \leq \frac{1}{n_0} = \frac{1}{\underbrace{\lceil \frac{1}{\varepsilon} \rceil}_{\geq \frac{1}{\varepsilon}}} \leq \frac{1}{\frac{1}{\varepsilon}} = \varepsilon$$

O-Notation — Grenzwerte 2/3

- Satz
  - Seien $f, g : N \rightarrow R$ 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 $< \infty$

O-Notation — Grenzwerte 3/3

- Beweis von (1) ... die anderen Beweise gehen analog

$$(1) \quad f = O(g) \iff \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$$

Beweis von „$\Rightarrow$":

$$f = O(g) \overset{\text{Def.}}{\underset{\text{von } O}{\Longrightarrow}} \exists n_0, C \ \forall n \geq n_0 \quad f(n) \leq C \cdot g(n)$$

$$\Longrightarrow \exists n_0, C \ \forall n \geq n_0 \quad \frac{f(n)}{g(n)} \leq C$$

$$\Longrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} \leq C \quad \blacksquare$$

Beweis von „$\Leftarrow$":

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = C < \infty \overset{\text{Def.}}{\underset{\lim}{\Longrightarrow}} \text{für z.B. } \varepsilon = 1 \ \exists n_0 \ \forall n \geq n_0 \ \frac{f(n)}{g(n)} \leq C + \overset{=\varepsilon}{1}$$

$$\Longrightarrow \exists n_0 \ \forall n \geq n_0 \quad f(n) \leq \underbrace{(C+1)}_{\text{Die Konst von der O-Notation}} \cdot g(n) \quad \blacksquare$$

Rechnen mit Grenzwerten 1/2

log zur Basis e
Eulersche Zahl
2.7...

z.B. $\lim_{n\to\infty} \frac{\ln n}{n} = ?$ $\quad \frac{\infty}{\infty}$

$(\ln n)' = \frac{1}{n}$ $\quad \frac{0}{0}$

$n' = 1$

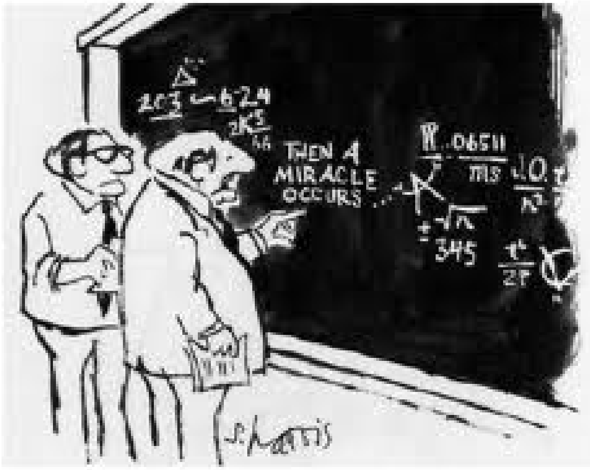
$\lim_{n\to\infty} \frac{(\ln n)'}{n'} = \lim_{n\to\infty} \frac{1}{n} = 0$

also nach H: $\lim_{n\to\infty} \frac{\ln n}{n} = 0$

- Variante 1: "zu Fuß"
  - Dafür hatten wir gerade das Beispiel

    $\lim_{n\to\infty} 1/n = 0$
- Variante 2: Regel von L'Hôpital
  - Seien $f, g : N \to R$ wie gehabt
  - Es existieren die ersten Ableitungen f' und g', sowie der Grenzwert $\lim_{n\to\infty} f'(n)/g'(n)$ ... dann gilt

    $\lim_{n\to\infty} f(n)/g(n) = \lim_{n\to\infty} f'(n)/g'(n)$
- Variante 3: göttliche Inspiration
  - Erst mit Promotion erlaubt ...



"I think you should be more explicit here in step two."

from *What's so Funny about Science?* by Sidney Harris (1977)

Rechnen mit Grenzwerten 2/2

- Was darf man ohne Beweis annehmen?
  - Gute Frage!
  - Da gibt es keine klare Regel
  - Im Zweifelsfall immer mehr beweisen als weniger !
  - **Beispiel 1:** $\lim_{n\to\infty} 1/n = 0$

    Brauchen Sie nicht mehr weiter zu beweisen
  - **Beispiel 2:** $\lim_{n\to\infty} 1/n^2 = 0$

    Einfach auf sowas wie Beispiel 1 zurückführen
  - **Beispiel 3:** $\lim_{n\to\infty} (\log n)/n = 0$

    Hier ist ein Argument angebracht, z.B. mit L'Hôpital

O-Notation — Diskussion 1/2

- Sprechweise
  - Die O-Notation schaut sich das Verhalten der Funktionen an, wenn $n \to \infty$ geht (es interessieren nur die $n \geq n_0$)
  - Wenn man Laufzeiten, Kosten, etc. als $O(...)$ oder $\Omega(...)$ oder $\Theta(...)$ ausdrückt, spricht man daher von

    asymptotischer Analyse
- Vorsicht
  - Asymptotische Analyse sagt nichts über das Laufzeit-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 ... nächste Folie

O-Notation — Diskussion 2/2

- Beispiel
  - Algorithmus A hat Laufzeit $f(n) = 80 \cdot n$
  - Algorithmus B hat Laufzeit $g(n) = 2 \cdot n \cdot \log_2 n$
  - Dann ist $f = O(g)$ aber **nicht** $f = \Theta(g)$
  - Das heißt, A ist asymptotisch schneller als B
    - d.h. ab irgendeinem n_0 ist für alle $n \geq n_0$ $f(n) \leq g(n)$
  - Allerdings:

$$n_0 = 2^{40} = (2^{10})^4 \approx 1000^4 = 10^{12} = 1 \text{ Billionen}$$

„Tera"

engl. Trillion

Dann ist für $n < n_0$:

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

Literatur / Links

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

    2.1 Asymptotic Notation
  - In Cormen/Leiserson/Rivest

    2.1 Asymptotic Notation
  - In Wikipedia

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

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