Informatik II: Algorithmen und Datenstrukturen SS 2017

Vorlesung 3a, Dienstag, 9. Mai 2017
(O-Notation, Teil 1)

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

UNI FREIBURG

Blick über die Vorlesung heute

- Organisatorisches
  - Korrektur vom ÜB1 — Wie, wann, was
  - Fragestunde — Teil 2 der VL morgen
  - Was zählt als Plagiat — Erinnerung + Klarstellung
  - Erfahrungen mit dem ÜB2 — Laufzeitanalyse
- O-Notation
  - Motivation, Definition, Beispiele — O, Ω, Θ, o, ω
  - ÜB3: ein paar Rechenaufgaben zu O und Θ … und die Laufzeit von drei Programmen als $\Theta(\ldots)$ bestimmen

Korrektur vom ÜB1 1/2

- Das funktioniert so
  - Im SVN in einer Datei im zum ÜB gehörigen Ordner, z.B.

    xy123/blatt-01/feedback-tutor.txt
  - Machen Sie einfach **svn update** in Ihrer Arbeitskopie
  - Sie erhalten Ihre Korrektur in der Regel spätestens am Freitag nach Abgabe, allerspätestens am Wochenende

    Manchmal aber auch schon Mittwoch oder Donnerstag
  - Falls Sie Wünsche oder Abneigungen in Bezug auf die Korrektur haben, sprechen Sie einfach mit Ihrem Tutor

    Das hat in der Vergangenheit immer sehr gut funktioniert

Korrektur vom ÜB1 2/2

UNI FREIBURG

- Rückmeldung vom Tutorentreffen
  - Es haben trotz mehrfacher Warnung einige ein MergeSort mit quadratischer Laufzeit produziert
  - Variablennamen sollten selbst-dokumentierend sein

    Einbuchstabige Laufvariablen bei Schleifen OK, aber nur da
  - Einige Leute sagen, sie hätten es nicht verstanden, aber haben sich keine Hilfe auf dem Forum geholt

    Man kann seinen Tutor auch um ein Treffen bitten !
  - Dokumentation sollte nicht sagen, was der Code macht (sieht man ja am Code), sondern welches Problem er löst

    Aber nicht sowas schreiben wie: "Das steht hier, weil es sonst einen Index-Out-Of-Range Fehler gibt"

Plagiat

- Erinnerung und Klarstellung
  - Es wurde in der Vorlesung 1a besprochen, stand auf den Folien und in rot und fett auf dem 1. Übungsblatt … und trotzdem gab es schon wieder einige Plagiatsfälle
  - Deswegen hier nochmal zur Klarstellung:

    Auch das Übernehmen von **Lösungen oder Code aus dem Internet**, und sei es nur teilweise, gilt als **Plagiat**

    Sie können miteinander diskutieren und recherchieren und googeln so viel Sie möchten

    Aber den Code bzw. Ihre Lösungen müssen Sie dann zu **100% selber schreiben**

    Ausnahme: alles aus dem SVN /public dürfen Sie benutzen

Fragestunde morgen

- Nach der Vorlesung morgen
  - Die Vorlesung morgen wird nur ca. 1 Stunde dauern
  - Danach machen wir eine Frage(halbe)stunde
  - Sie können dort Fragen aller Art rund um den Vorlesungsstoff (und die Übungen dazu) stellen

    Überlegen Sie sich was !

Erfahrungen mit dem ÜB2 1/3

- Zusammenfassung / Auszüge
  - Manche haben generelle Probleme mit dem Beweisen
  - Fehler bei der Aufgabenstellung von Aufgabe 1 ($\varphi = y/x$)

    Wurde im Forum sehr schnell (Mittwoch 12:22 Uhr) geklärt!
  - Probleme mit dem Verständnis von Aufgabe 4
  - Quartische Gleichungen haben noch eine Lösungsformel

    Keine allgemeine Lösungsformel erst **ab Grad 5**
  - Einige haben bei Aufgabe 4 Beweis aus der VL wiederholt

    Bitte die Aufgaben sorgfältig lesen und verstehen und nicht einfach nur halbautomatisch Output produzieren
  - Hurra, die Fragen sind zurück + die ÜB haben wieder Sinn!

Erfahrungen mit dem ÜB2 2/3

- Lösungsskizze Aufgabe 1+2

AUFGABE 1: $\frac{x}{y} = \frac{y}{x+y}$

$\Rightarrow \frac{y}{x} = \frac{x+y}{y} = \frac{x}{y} + 1 \quad \cdot\varphi \Rightarrow$

$\varphi = \frac{y}{x}$ $\quad$ $\frac{1}{\varphi} = \frac{x}{y}$

$\varphi^2 = 1 + \varphi$

$z^2 - z - 1 = 0 \Rightarrow z_{1;2} = \frac{1}{2} \pm \sqrt{\frac{1}{4}+1} = \frac{1 \pm \sqrt{5}}{2}$

$\sqrt{\frac{1}{4}+1} = \sqrt{\frac{1}{4}+\frac{4}{4}} = \sqrt{\frac{5}{4}} = \frac{\sqrt{5}}{2}$

$x+y$, x, y

z.z. $\varphi^2 = \varphi + 1$

$\varphi > 1 \Rightarrow \varphi = \frac{1+\sqrt{5}}{2}$

andere Lösung $\psi = \frac{1-\sqrt{5}}{2} = 1 - \varphi$

AUFGABE 2: z.z. $F_n = \frac{1}{\sqrt{5}}(\varphi^n - \psi^n) \quad \forall n \in \mathbb{N}$

Induktionsanfang: $n = 1 \Rightarrow \frac{1}{\sqrt{5}}(\varphi - \psi) = \frac{1}{\sqrt{5}} \cdot \sqrt{5} = 1$

$n = 2 \Rightarrow \frac{1}{\sqrt{5}}(\varphi^2 - \psi^2) = \dots = 1$

Induktionsschritt: $n, n+1 \to n+2$

$F_{n+2} \overset{\text{nach Definition der } F_n}{=} F_{n+1} + F_n \overset{\text{nach I.V.}}{=} \frac{1}{\sqrt{5}}(\varphi^{n+1} - \psi^{n+1}) + \frac{1}{\sqrt{5}}(\varphi^n - \psi^n)$

$= \frac{1}{\sqrt{5}}\left(\varphi^{n+1} + \varphi^n - (\psi^{n+1} + \psi^n)\right)$

$\varphi^n + \varphi^{n+1} = \varphi^n(\varphi+1) = \varphi^n \cdot \varphi^2 = \varphi^{n+2}$

$\psi^n + \psi^{n+1} = \psi^n(\psi+1) = \psi^n \cdot \psi^2 = \psi^{n+2}$

Erfahrungen mit dem ÜB2 3/3

UNI FREIBURG

- Lösungsskizze Aufgabe 4 (untere Schranke)
  - In Vorlesung 2b wurde gezeigt: sei T(n) eine obere Schranke für die Laufzeit eines vergleichsbasierten Algorithmus für Eingabegröße n, dann $T(n) \geq \log_2 (n!)$
  - Das heißt: für **mindestens eine Eingabe** der Größe n muss die Laufzeit $\geq \log_2 (n!)$ sein
  - Das heißt **nicht**, dass **für alle Eingaben** der Größe n die Laufzeit $\geq \log_2 (n!)$ sein muss
  - Es gilt auch nicht, weil man jedem Sortieralgorithmus, einen einfachen Test voranstellen kann, der in $\leq A \cdot n$ Zeit prüft, ob die Eingabe schon sortiert ist

    Für sortierte Eingaben ist die Laufzeit dann $\leq A \cdot n$

O-Notation – Grundlagen 1/11

UNI FREIBURG

- Erinnerung
  - Wir haben jetzt mehrfach die Laufzeit $T(n)$ in Abhängigkeit von der Eingabegröße abgeschätzt
  - Die Werte der Konstanten waren dabei sekundär … und auch, wenn die Schranken erst ab $n \geq$ irgendeinem n_0 galten

    Für sehr kleine Eingaben sind Programme ja sowieso schnell
  - Zum Beispiel hatte wir, für $n \geq$ irgendeinem n_0 :

    Die Laufzeit von MinSort ist "irgendwas mal" n^2

    Die Laufzeit von MergeSort ist "irgendwas mal" $n \cdot \log n$

    Die Laufzeit von CountingSort ist "irgendwas mal" n

    Vergleichsbasiertes Sortieren dauert "irgendwas mal" $n \cdot \log n$

O-Notation – Grundlagen 2/11

UNI FREIBURG

- Motivation
  - Genau das wollen wir jetzt formaler machen, damit wir in Zukunft präzise sagen bzw. schreiben können

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

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

    Die Laufzeit von CountingSort ist $\mathbf{\Theta(n)}$

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

O-Notation – Grundlagen 3/11

UNI FREIBURG

- 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

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

    $f(n) = 2 \cdot n \cdot (\log_2 n - 5)$ für $n \leq 32 = 2^5$: $g(n) \leq 0$; für $n > 32$: $g(n) > 0$

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

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

O-Notation – Grundlagen 4/11

UNI FREIBURG

- 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 ...

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

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

eigentlich wäre korrekt: $g \in O(f)$

INTUITIV: alle Funktionen, die nicht stärker wachsen als f

O-Notation – Grundlagen 5/11

- 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 :

zu zeigen: $5 \cdot n + 7 \leq C \cdot n$ für „irgendein" C, für $n \geq$ „irgendein" n_0

Beweis 1: $5 \cdot n + 7 \leq 5 \cdot n + 7 \cdot n = \underbrace{12 \cdot n}_{=: C}$

$\leq 7 \cdot n$ für $n \geq \underbrace{1}_{=: n_0}$

Beweis 2: $5 \cdot n + 7 \leq 5 \cdot n + n = \underbrace{6 \cdot n}_{=: C}$

$\leq n$ für $n \geq \underbrace{7}_{=: n_0}$

O-Notation – Grundlagen 6/11

UNI FREIBURG

- 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

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

$\Rightarrow g(n) > f(n) \quad \forall n \geq 1 \quad$ also $\quad g > f$

TROTZDEM: $g = O(f)$

O-Notation – Grundlagen 7/11

- 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

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

zu zeigen: $5 \cdot n + 7 \geq C \cdot n$ für „irgendein" C für $n \geq$ „irgendein" n_0

Beweis: $5 \cdot n + 7 \geq 5 \cdot n$

≥ 0 sogar für alle n ; $= C$

O-Notation – Grundlagen 8/11

UNI FREIBURG

- 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 14 $\Rightarrow$ $5 \cdot n + 7 = O(n)$

    Folie 16 $\Rightarrow$ $5 \cdot n + 7 = \Omega(n)$

    $\Rightarrow$ $5 \cdot n + 7 = \Theta(n)$ ∎

O-Notation – Grundlagen 9/11

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

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

    einziger Unterschied zu $O(g)$

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

    einziger Unterschied zu $\Omega(g)$
  - Intuitiv:

    $g = o(f)$: 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 10/11

- 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	$>$

WICHTIG: es macht Sinn zu sagen, dass ein Algorithmus Laufzeit mindestens $O(n^2)$ hat.

wenn schon, dann mindestens $\Theta(n^2)$

O-Notation – Grundlagen 11/11

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

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

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

Gute Zusatzaufgabe für die, die vom ÜB unterfordert sind

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