13 QUANTITATIVE ASPEKTE VON ALGORITHMEN

Hinweise für die Tutorien

13.1 RESSOURCENVERBRAUCH BEI BERECHNUNGEN

Snelting hat in seiner Vorlesung umgestellt. Den im Skript zitierten Insertionsort-Algorithmus haben die Studenten in Programmieren bis zum 9.12. wohl noch nicht gesehen. Nicht darauf aufbauen.

Im vorgezogenen Abschnitt „Rekursion" tauchen aber die Binomialkoeffizienten auf:

$$\binom{n}{k} = \begin{cases} 0 & \text{falls } k = 0 \vee k = n \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{sonst} \end{cases}$$

13.2 GROSS-O-NOTATION

13.2.1 Ignorieren konstanter Faktoren

Wir machen das ein bisschen anders als andere:

- erst wird Θ eingeführt, und danach O:
  - ich finde, dass Θ das näher liegende ist, und man kann sich erst mal drauf beschränken, dass Ignorieren konstanter Faktoren kennenzulernen
  - die Verallgemeinerungen zu O und Ω sind dann leichter (meiner Meinung nach)
- ich führe erst eine Äquivalenzrelation $\asymp$ ein, und dann $\Theta(f)$ als Äquivalenzklasse (ohne dieses Wort schon zu benutzen) von Funktionen.
- auch so ist hinterher leicht zu sehen, dass $\Theta(f) = O(f) \cap \Omega(f)$. *Das kommt vielleicht aufs Aufgabenblatt?*
- Achtung: einiges könnte man auch leicht unter Verwendung von lim, oder genauer mit lim sup argumentieren, aber die Inwis haben vermutlich noch keine Grenzwerte.

Θ und Polynome: Man versuche klar zu machen, dass immer $f \asymp g$ ist, wenn f und g Polynome gleichen Grades sind, also z. B. $42n^6 - 33n^3 + 222n^2 - 15 \asymp 66n^6 + 55555n^5$. Das kann man z. B. in Anlehnung an $n^3 + 5n^2 \asymp 3n^3 - n$ aus der Vorlesung machen.

Beispiel: Logarithmenfunktionen haben alle größenordnungsmäßig das gleiche Wachstum:

- Logarithmen sind ja wohl definitiv Schulwissen. Trotzdem darauf vorbereitet sein, dass Fragen kommen. Also: Für $a \in \mathbb{R}_+$ und $n \in \mathbb{N}_+$ ist $\log_a(n)$ die Zahl mit $a^{\log_a(n)} = n$. Beachte: $n \geq 1$, da $\log 0$ nicht definiert.
- Man zeige: $\log_2(n) \in \Theta\left(\log_8(n)\right)$

- man beginne vielleicht mit Beispielen:

n	1	8	64	512	4096
$\log_8(n)$	0	1	2	3	4
$\log_2(n)$	0	3	6	9	12

  da sollte man doch was sehen ...
- dann rechnen: $n = 8^{\log_8 n} = (2^3)^{\log_8(n)} = 2^{3\log_8(n)}$, also gilt für alle $n \geq 1$: $\log_2(n) = 3\log_8(n)$ und $\log_8(n) = \frac{1}{3}\log_2(n)$
- wenn das klar ist, dann wohl auch ...

• allgemein: $\log_b(n) \in \Theta(\log_a(n))$, denn

$$b^{\log_b(n)} = n = a^{\log_a(n)} = (b^{\log_b(a)})^{\log_a(n)} = b^{\log_b(a)\cdot\log_a(n)}$$

also liefert (Exponentiation ist injektiv) der Vergleich der Exponenten (oder anders gesagt: Logarithmieren beider Seiten): $\log_b(n) = \log_b(a) \cdot \log_a(n)$
also für $c' = c = \log_b(a)$ und alle $n \geq 1$ gilt: $c\log_a(n) \leq \log_b(n) \leq c'\log_a(n)$

• Man kann also einfach $\Theta(\log n)$ schreiben, ohne die Basis anzugeben, denn sie ist egal.

13.2.2 Notation für obere und untere Schranken des Wachstums

zum Thema $O()$:

• Damit die Studenten ein besseres Gefühl für $O(\cdot)$ bekommen, bitte noch mal genau $n^a \in O(n^b)$ falls $a \leq b$ betrachten.

• Aber damit da kein falscher EIndruck entsteht: **Bitte beachten:** $\preceq$ und $\succeq$ sind *keine* totalen Ordnungen. Es gibt unvergleichbare Funktionen.

$$f(n) = \begin{cases} 1 & \text{falls } n \text{ gerade} \\ n & \text{falls } n \text{ ungerade} \end{cases}$$

$$g(n) = \begin{cases} n & \text{falls } n \text{ gerade} \\ 1 & \text{falls } n \text{ ungerade} \end{cases}$$

Es gilt *nicht* $g \preceq f$, es gilt *nicht* $f \preceq g$ und es gilt erst recht *nicht* $f \asymp g$.

Und das liegt auch nicht daran, dass die Funktionen so hin und her springen; für monoton wachsende Funktionen kann man so etwas auch machen; evtl. bungsaufgabe

Zu $\Omega(\cdot)$: vielleicht auch ein paar einfache Beispiele: Macht es den Studenten Probleme, sich von $n^2 \in \Omega(\log n)$ zu überzeugen?

13.2.3 Die furchtbare Schreibweise

Bitte Fragen beantworten. ABER: Ich sehe zwar einen Grund so etwas lesen zu können, aber keinen Grund diesen Unfug schreibenderweise zu üben.

13.2.4 Rechnen im O-Kalkül

Ich habe Probleme, mich in die Probleme der Studenten hineinzuversetzen. Es erscheint mir alles so banal :-(

13.3 MATRIXMULTIPLIKATION

13.3.1 Rückblick auf die Schulmethode

Matrixmultiplikation mit den Blockmatrizen:

ich werde das auführlicher erläutern als vergangenes Jahr

trotzdem: vielleicht muss man das noch ein bisschen erklären. Man nehme einfach 4×4-Matrizen und sehe sich z. B. $c_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} + a_{14}b_{41}$:

Man schreibe sich einige der Blöcke A_{11} usw. hin. Dann sieht man: Der erste Teil $a_{11}b_{11} + a_{12}b_{21}$ „kommt von"/„passt zu" $A_{11}B_{11}$ und der zweite Teil $a_{13}b_{31} + a_{14}b_{41}$ „kommt von"/„passt zu" $A_{12}B_{21}$.

13.3.2 Algorithmus von Strassen

Ich drücke mich darum, eine rekursive Prozedur hinzuschreiben.

Immerhin sollte Rekursion bei Snelting dran gewesen sein. Aber wenn man das zum ersten Mal gehört hat ...

Weitere Übungsmöglichkeit: Codeschnipsel aus Sneltings Folien von 2008 für Berechnung der Binomialkoeffizienten:

```
static int binom(int n, int k) {
  assert n >= k && k >= 0;
  if (k == 0 || k == n) {
    return 1;
  } else {
    return binom(n - 1, k - 1) + binom(n - 1, k);
  }
}
```

Diskussion: Wieviele Aufrufe von `binom` in Abhängigkeit von n werden bei der Berechnung eines $\binom{n}{k}$ gemacht? Im Detail ist das nicht ganz schön zu machen. Man überzeuge sich aber (mit Hilfe eines Beispiels?) davon, dass man mindestens 2^k Aufrufe der Form $binom(n-k)(x)$ mit $0 \leq x \leq k$ hat. Das sind im Fall $k = n/2$ also immerhin $\left(\sqrt{2}\right)^n$.

13.4 Abschätzung des Wachstums rekursiv definierter Funktionen

Zum Mastertheorem komme ich vermutlich erst am 16.12.; dann sollte in Programmieren rekursives Suchen/Sortieren dran gewesen sein. Das gibt dann noch mal Motivation für

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

Mastertheorem

- Fall 2: $f(n) \in \Theta\left(n^{\log_b a}\right)$ schlägt bei Quicksort zu
  - Formel anwenden liefert $n \log n$
  - schönes Bildchen hilft
- Fall 3: nur bei Nachfragen diskutieren ...
- statt dessen darauf hinweisen, dass einem das Mastertheorem nicht weiterhilft, wenn man eine Probleminstanz anders zerhackt, wie etwa bei $(n+1)! = (n+1) * n!$ oder

$$\binom{n}{k} = \begin{cases} 0 & \text{falls } k = 0 \lor k = n \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{sonst} \end{cases}$$

  (siehe weiter vorne)

