Übung Algorithmen und Datenstrukturen

Sommersemester 2017

Patrick Schäfer, Humboldt-Universität zu Berlin

Agenda

1. **Landau-Notation (Wiederholung & Vertiefung)**
2. Pseudocode-Analyse & Algorithmenentwurf

Landau-Notation

- Definitionen:
  - $O(g) = \{f \mid \exists c > 0 \;\; \exists n_0 > 0 \;\; \forall n \geq n_0 \colon \; f(n) \leq c \cdot g(n)\}$
  - $\Omega(g) = \{f \mid \exists c > 0 \;\; \exists n_0 > 0 \;\; \forall n \geq n_0 \colon \; f(n) \geq c \cdot g(n)\}$
  - $\Theta(g) = O(g) \cap \Omega(g)$
  - $o(g) = \{f \mid \forall c > 0 \;\; \exists n_0 > 0 \;\; \forall n \geq n_0 \colon \; f(n) < c \cdot g(n)\}$
  - $\omega(g) = \{f \mid \forall c > 0 \;\; \exists n_0 > 0 \;\; \forall n \geq n_0 \colon \; f(n) > c \cdot g(n)\}$

- Nützliche Zusammenhänge:
  - $f \in O(g) \Leftrightarrow \; g \in \Omega(f)$
  - $f \in o(g) \Leftrightarrow g \in \omega(f)$
  - $f \in o(g) \Rightarrow f \in \mathrm{O}(g)$
  - $f \in o(g) \Rightarrow \; f \notin \Omega(g)$
  - $f \in \omega(g) \Rightarrow f \in \Omega(g)$
  - $f \in \omega(g) \Rightarrow f \notin O(g)$

- Hinreichende Kriterien:
  - $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)} < \infty \Rightarrow f \in \mathrm{O}(g)$
  - $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Leftrightarrow f \in o(g)$
  - $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)} > 0 \Rightarrow f \in \Omega(g)$
  - $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Leftrightarrow f \in \omega(g)$

- Satz von L'Hôpital:

  Für zwei differenzierbare Funktionen f und g, mit $\lim\limits_{n\to\infty} f(n) = \lim\limits_{n\to\infty} g(n) = \mathbf{0}$ oder $\lim\limits_{n\to\infty} f(n) = \lim\limits_{n\to\infty} g(n) = \infty$, gilt:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} \overset{L'H}{=} \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

- Logarithmengesetze:

$$(\log_2 n)' = \left(\frac{\ln n}{\ln 2}\right)' = \frac{1}{\ln 2 \cdot n}$$

- Ableitungsregeln:

$a)\; f(x) = x^n, f'(x) = nx^{n-1}$

$b)\; f(x) = g\big(h(x)\big), f'(x) = g'\big(h(x)\big) \cdot h'(x)$

$c)\; f(x) = g(x) \cdot h(x), f'(x)$
$\quad = g'(x) \cdot h(x) + g(x) \cdot h'(x)$

Wiederholung (Beweisalternativen)

- Gilt $f(n) \in O(g(n))$?
  - $f(n) = 3n^5 + 4n^3 + 15$
  - $g(n) = n^5$

a) Wir betrachten den Grenzwert:

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \frac{n^5}{3n^5 + 4n^3 + 15}$$

$$\overset{L'H}{=} \lim_{n\to\infty} \frac{(n^5)'}{(3n^5+4n^3+15)'} = \lim_{n\to\infty} \frac{5n^4}{5\cdot 3n^4 + 3\cdot 4n^2}$$

$$\overset{L'H}{=} \lim_{n\to\infty} \frac{4\cdot 5n^3}{5\cdot 3n^3 + 2\cdot 3\cdot 4n}$$

$$\overset{L'H}{=} \lim_{n\to\infty} \frac{3\cdot 4\cdot 5n^2}{3\cdot 4\cdot 5\cdot 3n^2 + 1\cdot 2\cdot 3\cdot 4}$$

$$\overset{L'H}{=} \lim_{n\to\infty} \frac{2\cdot 3\cdot 4\cdot 5n}{2\cdot 3\cdot 4\cdot 5\cdot 3n} = \lim_{n\to\infty} \frac{1}{3} = \frac{1}{3}$$

$\Rightarrow f(n) \in O(n^5)$ und $f(n) \in \Omega(n^5)$

b) Wir betrachten die Definition:

$(\exists c, n_0 > 0 \ \forall n \geq n_0: \ 3n^5 + 4n^3 + 15 \leq c \cdot n^5)$

Wähle zB $c = 3 + 4 + 15 = 22$

dann gilt $\forall n \geq 1$:

$$3n^5 + 4n^3 + 15$$
$$\leq 3n^5 + 4n^5 + 15n^5$$
$$\leq 22n^5$$

$\Rightarrow f(n) \in O(n^5)$

Vorbereitung auf Aufgabe 1.1

- Gilt $f(n) \in O(g(n))$ oder $f(n) \in \Omega(g(n))$?
  - $f(n) = \log^2 n = \log n \cdot \log n$
  - $g(n) = \sqrt{n}$

Wir betrachten den Grenzwert:

$$\lim_{n\to\infty} \frac{\log^2 n}{\sqrt{n}} = \lim_{n\to\infty} \frac{\frac{(\ln n)^2}{(\ln 2)^2}}{n^{\frac{1}{2}}}$$

$$\overset{L'H}{=} \lim_{n\to\infty} \frac{\frac{1}{(\ln 2)^2} 2(\ln n)\frac{1}{n}}{\frac{1}{2}n^{-\frac{1}{2}}} = \lim_{n\to\infty} \frac{\frac{4}{(\ln 2)^2}\ln n}{n^{\frac{1}{2}}}$$

$$\overset{L'H}{=} \lim_{n\to\infty} \frac{\frac{4}{(\ln 2)^2}\frac{1}{n}}{\frac{1}{2}n^{-\frac{1}{2}}} = \lim_{n\to\infty} \frac{\frac{8}{(\ln 2)^2}}{n^{\frac{1}{2}}} = 0$$

$\Rightarrow f(n) \in o(g(n))$ und $f(n) \notin \Omega(g(n))$

Verwendete Zusammenhänge:

- $(\log_2 n)' = \left(\frac{\ln n}{\ln 2}\right)' = \frac{1}{\ln 2 \cdot n}$
- $f(x) = g(h(x)), f'(x) = g'(h(x)) \cdot h'(x)$
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Leftrightarrow f \in o(g), f \in o(g) \Longrightarrow f \notin \Omega(g)$

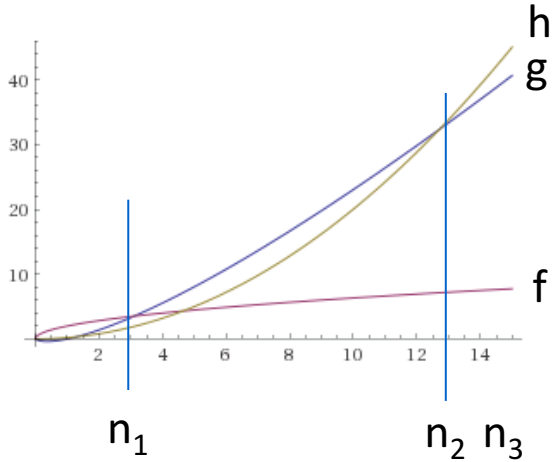
Vorbereitung auf Aufgabe 1.2

- Für beliebige Funktionen $f, g, h : \mathbb{N} \to \mathbb{N}$ gilt:

  $f \epsilon O(g) \wedge g \epsilon O(h) \Rightarrow f \epsilon O(h)$

- Gegeben:

  $\exists c_1, n_1 \colon \forall n \geq n_1 \colon f(n) \leq c_1 \cdot g(n) \quad \wedge$
  $\exists c_2, n_2 \colon \forall n \geq n_2 \colon g(n) \leq c_2 \cdot h(n)$



- Zu zeigen:

  $\exists c_3, n_3 \colon \forall n \geq n_3 \colon f(n) \leq c_3 \cdot h(n)$

- Wähle $c_3 = c_1 \cdot c_2$ und $n_3 = \max(n_1, n_2)$, dann gilt:

  $\forall n \geq n_3 \colon f(n) \leq c_1 \cdot g(n) \leq c_1 \cdot c_2 \cdot h(n) \leq c_3 \cdot h(n)$

$\Rightarrow f \epsilon O(h)$

Agenda

1. Landau-Notation (Wiederholung & Vertiefung)
2. **Pseudocode-Analyse & Algorithmenentwurf**

Pseudocode-Analyse

Algorithmus bar

Input: Array A der Länge $|A| = n$ mit $n \geq 2$
Output: Zahl x

```
x := 0;
for i := 1 to n do
  for j := i + 1 to n do
    if x < |A[i] − A[j]| then
      x := |A[i] − A[j]|;
    end if
  end for
end for
return x;
```

a) Was berechnet der Algorithmus bar?

b) Analysieren Sie die Laufzeit des Algorithmus in Abhängigkeit von n.

c) Entwerfen Sie einen bezüglich der Laufzeit effizienteren Algorithmus für das Berechnungsproblem. Notieren Sie Ihren Algorithmus als Pseudocode und analysieren Sie dessen Laufzeit.

Schreibtischtest

- Händisches Verfahren zur Verständnisgewinnung und Überprüfung der Korrektheit eines Algorithmus

Was berechnet der Algorithmus *bar* bei Eingabe
A = [3,2,-1,4] ?

Algorithmus bar

Input: Array A der Länge $|A| = n$ mit $n \geq 2$
Output: Zahl x

```
x := 0;
for i := 1 to n do
   for j := i + 1 to n do
      if x < |A[i] − A[j]| then
         x := |A[i] − A[j]|;
      end if
   end for
end for
return x;
```

Belegungen der Variablen

i	j	x
1	2	…
1	…	…
1	n	…
2	1	…
…	…	…

Laufzeitanalyse

- “Analysieren Sie die Laufzeit”: d.h., gesucht wird die Laufzeit in O-Notation.

- Elementare Operation haben konstante Laufzeit:

```
x := 0;                 = O(1)
x := x+i;               = O(1)
a[1] = b[1];            = O(1)
```

- Bedingte Anweisung haben konstante Laufzeit:

```
if (x<0) then           O(1)
   x= -1 * x;         + O(1)
end if;                 = O(1)
```

Laufzeitanalyse

- Laufzeit von Schleifen ist abhängig von der Anzahl der Durchläufe.

```
for i:=1 to 10 do
    x += i;
end for
```

$10 \in O(1)$
$\cdot\ O(1)$
$= O(1)$

```
for i:=1 to |a| do
    x += a[i];
end for
```

$O(|a|)$
$\cdot\ O(1)$
$= O(|a|)$ $\quad |a| = n$

```
for i:=1 to 2 do
    for j:=1 to |a| do
        x += i*a[j];
    end for
end for
```

$2 \in O(1)$
$\cdot\ O(|a|)$
$= O(|a|)$ $\quad |a| = n$

```
for i:=1 to n do
    for j:=1 to m do
        x += i*j;
    end for
end for
```

$O(n)$
$\cdot\ O(m)$
$\cdot\ O(1)$
$= O(n \cdot m)$

Wichtige Komplexitätsklassen

- $O(1)$: konstant (Elementare Operation)
- $O(\log n)$: logarithmisch (Halbieren der Eingabe, Binäre Suche)
- $O(n)$: linear (Einfache Schleife, Sequentielle Suche)
- $O(n \log n)$: linear logarithmisch (Divide & Conquer, Mergesort)
- $O(n^2)$: quadratisch (Verschachtelte Schleife, Bubble Sort)
- $O(n^k)$: polynomial (Verschachtelte Schleifen)
- $O(2^n)$: exponentiell (Traveling Salesman)

- Komplexität bis zu $O(n^2)$ ist akzeptabel (zwei verschachtelte Schleifen).

Pseudocodeanalyse

Algorithmus bar

Input: Array A der Länge $|A| = n$ mit $n \geq 2$
Output: Zahl x

```
x := 0;
for i := 1 to n do
  for j := i + 1 to n do
    if x < |A[i] − A[j]| then
      x := |A[i] − A[j]|;
    end if
  end for
end for
return x;
```

b) Analysieren Sie die Laufzeit des Algorithmus in Abhängigkeit von n.

c) Entwerfen Sie einen bezüglich der Laufzeit effizienteren Algorithmus für das Berechnungsproblem. Notieren Sie Ihren Algorithmus als Pseudocode und analysieren Sie dessen Laufzeit.

Algorithmenentwurf

Gegeben sei ein Array A der Länge $|A| = n - 1$, in dem jede Zahl zwischen 1 und n bis auf eine genau einmal vorkommt. Ein Beispiel für A mit $n = 5$ wäre also $A = [2, 4, 1, 5]$. Entwerfen Sie einen Algorithmus, der als Eingabe das Array A erhält und als Ausgabe die in A fehlende Zahl zurückgibt. Die Laufzeit des zu entwerfenden Algorithmus soll in $O(n)$ liegen. Notieren Sie Ihren Algorithmus als Pseudocode.